Deep Learning for Poets (Part II)

Amir H. Payberah
payberah@kth.se
19/12/2018

TensorFlow

Linear and Logistic regression

Deep Feedforward Networks

CNN, RNN, Autoencoders

TensorFlow

Linear and Logistic regression

Deep Feedforward Networks

CNN, RNN, Autoencoders

Linear Algebra Review

Vector

- A vector is an array of numbers.

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$

Vector

- A vector is an array of numbers.
- Notation:
  - Denoted by **bold** lowercase letters, e.g., $\mathbf{x}$.
  - x_i denotes the ith entry.

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$

Matrix and Tensor

- A matrix is a 2-D array of numbers.

$$\mathbf{A} = \begin{bmatrix} a_{1,1} & a_{1,2} & a_{1,3} & \cdots & a_{1,n} \\ a_{2,1} & a_{2,2} & a_{2,3} & \cdots & a_{2,n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{m,1} & a_{m,2} & a_{m,3} & \cdots & a_{m,n} \end{bmatrix}$$

Matrix and Tensor

- A matrix is a 2-D array of numbers.
- A tensor is an array with more than two axes.

$$\mathbf{A} = \begin{bmatrix} a_{1,1} & a_{1,2} & a_{1,3} & \dots & a_{1,n} \\ a_{2,1} & a_{2,2} & a_{2,3} & \dots & a_{2,n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{m,1} & a_{m,2} & a_{m,3} & \dots & a_{m,n} \end{bmatrix}$$

Matrix and Tensor

- A matrix is a 2-D array of numbers.
- A tensor is an array with more than two axes.
- Notation:
  - Denoted by **bold** uppercase letters, e.g., $\mathbf{A}$.
  - a_{ij} denotes the entry in ith row and jth column.
  - If $\mathbf{A}$ is $m \times n$, it has m rows and n columns.

$$\mathbf{A} = \begin{bmatrix} a_{1,1} & a_{1,2} & a_{1,3} & \dots & a_{1,n} \\ a_{2,1} & a_{2,2} & a_{2,3} & \dots & a_{2,n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{m,1} & a_{m,2} & a_{m,3} & \dots & a_{m,n} \end{bmatrix}$$

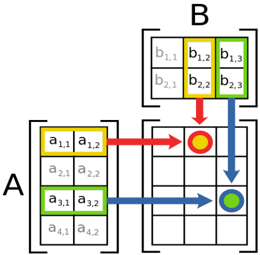
Matrix Addition and Subtraction

- The matrices must have the same dimensions.

$$\mathbf{A} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} + \begin{bmatrix} e & f \\ g & h \end{bmatrix} = \begin{bmatrix} a+e & b+f \\ c+g & d+h \end{bmatrix}$$

Matrix Product

- The matrix product: $\mathbf{C} = \mathbf{AB}$.

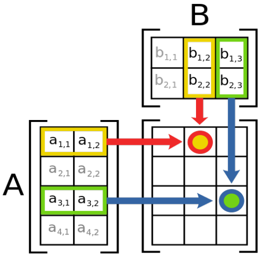


[https://en.wikipedia.org/wiki/Matrix_multiplication]

Matrix Product

- The matrix product: $\mathbf{C} = \mathbf{AB}$.
- If $\mathbf{A}$ is of shape $\mathtt{m} \times \mathtt{n}$ and $\mathbf{B}$ is of shape $\mathtt{n} \times \mathtt{p}$, then $\mathbf{C}$ is of shape $\mathtt{m} \times \mathtt{p}$.

$$c_{ij} = \sum_k a_{ik} b_{kj}$$



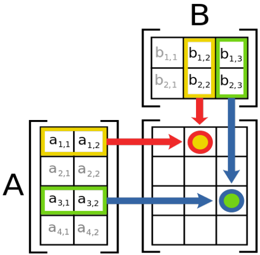
[https://en.wikipedia.org/wiki/Matrix_multiplication]

Matrix Product

- The matrix product: $\mathbf{C} = \mathbf{AB}$.
- If $\mathbf{A}$ is of shape $\mathtt{m} \times \mathtt{n}$ and $\mathbf{B}$ is of shape $\mathtt{n} \times \mathtt{p}$, then $\mathbf{C}$ is of shape $\mathtt{m} \times \mathtt{p}$.

$$c_{ij} = \sum_{k} a_{ik} b_{kj}$$

- Properties
  - Associative: $(\mathbf{AB})\mathbf{C} = \mathbf{A}(\mathbf{BC})$
  - Not commutative: $\mathbf{AB} \neq \mathbf{BA}$



[https://en.wikipedia.org/wiki/Matrix_multiplication]

Matrix Transpose

- Swap the rows and columns of a matrix.

$$\mathbf{A} = \begin{bmatrix} \mathtt{a} & \mathtt{b} \\ \mathtt{c} & \mathtt{d} \\ \mathtt{e} & \mathtt{f} \end{bmatrix} \Rightarrow \mathbf{A}^\intercal = \begin{bmatrix} \mathtt{a} & \mathtt{c} & \mathtt{e} \\ \mathtt{b} & \mathtt{d} & \mathtt{f} \end{bmatrix}$$

Matrix Transpose

- Swap the rows and columns of a matrix.

$$\mathbf{A} = \begin{bmatrix} a & b \\ c & d \\ e & f \end{bmatrix} \Rightarrow \mathbf{A}^\intercal = \begin{bmatrix} a & c & e \\ b & d & f \end{bmatrix}$$

- Properties
  - $\mathbf{A}_{ij} = \mathbf{A}^\intercal_{ji}$
  - If $\mathbf{A}$ is $m \times n$, then $\mathbf{A}^\intercal$ is $n \times m$
  - $(\mathbf{A} + \mathbf{B})^\intercal = \mathbf{A}^\intercal + \mathbf{B}^\intercal$
  - $(\mathbf{AB})^\intercal = \mathbf{B}^\intercal \mathbf{A}^\intercal$

Inverse of a Matrix

- If $\mathbf{A}$ is a square matrix, its inverse is called $\mathbf{A}^{-1}$.

$$\mathbf{A}\mathbf{A}^{-1} = \mathbf{A}^{-1}\mathbf{A} = \mathbf{I}$$

Inverse of a Matrix

- If $\mathbf{A}$ is a square matrix, its inverse is called $\mathbf{A}^{-1}$.

$$\mathbf{A}\mathbf{A}^{-1} = \mathbf{A}^{-1}\mathbf{A} = \mathbf{I}$$

- Where $\mathbf{I}$, the identity matrix, is a diagonal matrix with all 1's on the diagonal.

$$\mathbf{I}_2 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \qquad \mathbf{I}_3 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

L^p Norm for Vectors

- We can measure the size of vectors using a norm function.

L^p Norm for Vectors

- We can measure the size of vectors using a norm function.
- Norms are functions mapping vectors to non-negative values.

L^p Norm for Vectors

- We can measure the size of vectors using a norm function.
- Norms are functions mapping vectors to non-negative values.
- L^1 norm

$$||\mathbf{x}||_1 = \sum_i |x_i|$$

L^p Norm for Vectors

- We can measure the size of vectors using a norm function.
- Norms are functions mapping vectors to non-negative values.
- L^1 norm

$$||\mathbf{x}||_1 = \sum_i |x_i|$$

- L^2 norm

$$||\mathbf{x}||_2 = (\sum_i |x_i|^2)^{\frac{1}{2}} = \sqrt{x_1^2 + x_2^2 + \cdots + x_n^2}$$

L^p Norm for Vectors

- We can measure the size of vectors using a norm function.
- Norms are functions mapping vectors to non-negative values.
- L^1 norm

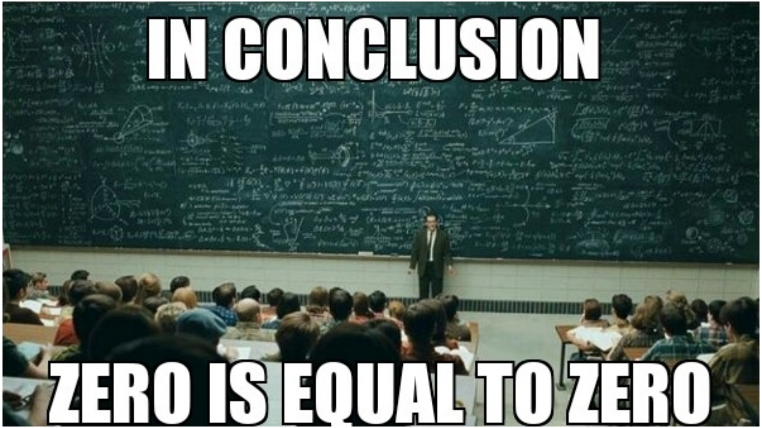
$$||\mathbf{x}||_1 = \sum_i |x_i|$$

- L^2 norm

$$||\mathbf{x}||_2 = (\sum_i |x_i|^2)^{\frac{1}{2}} = \sqrt{x_1^2 + x_2^2 + \cdots + x_n^2}$$

- L^p norm

$$||\mathbf{x}||_p = (\sum_i |x_i|^p)^{\frac{1}{p}}$$


IN CONCLUSION
ZERO IS EQUAL TO ZERO

Probability Review

Random Variables

- Random variable: a variable that can take on different values randomly.

Random Variables

- Random variable: a variable that can take on different values randomly.
- Random variables may be discrete or continuous.

Random Variables

- Random variable: a variable that can take on different values randomly.
- Random variables may be discrete or continuous.
- Notation:
  - Denoted by an upper case letter, e.g., X
  - Values of a random variable X are denoted by lower case letters, e.g., x and y.

Probability Distributions

- Probability distribution: how likely a random variable is to take on each of its possible states.

Probability Distributions

- Probability distribution: how likely a random variable is to take on each of its possible states.
- E.g., the random variable X denotes the outcome of a coin toss.

Probability Distributions

- Probability distribution: how likely a random variable is to take on each of its possible states.
- E.g., the random variable $\mathtt{X}$ denotes the outcome of a coin toss.
  - The probability distribution of $\mathtt{X}$ would take the value 0.5 for $\mathtt{X} = \mathtt{head}$, and 0.5 for $\mathtt{Y} = \mathtt{tail}$ (assuming the coin is fair).

Probability Distributions

- Probability distribution: how likely a random variable is to take on each of its possible states.
- E.g., the random variable $\mathtt{X}$ denotes the outcome of a coin toss.
  - The probability distribution of $\mathtt{X}$ would take the value 0.5 for $\mathtt{X} = \mathtt{head}$, and 0.5 for $\mathtt{Y} = \mathtt{tail}$ (assuming the coin is fair).
- The way we describe probability distributions depends on whether the variables are discrete or continuous.

Discrete Variables

- Probability mass function (PMF): the probability distribution of a discrete random variable `X`.

Discrete Variables

- Probability mass function (PMF): the probability distribution of a discrete random variable $\mathtt{X}$.

- Notation: denoted by a lowercase $\mathtt{p}$.
  - E.g., $\mathtt{p}(\mathtt{x}) = 1$ indicates that $\mathtt{X} = \mathtt{x}$ is certain
  - E.g., $\mathtt{p}(\mathtt{x}) = 0$ indicates that $\mathtt{X} = \mathtt{x}$ is impossible

Discrete Variables

- Probability mass function (PMF): the probability distribution of a discrete random variable X.

- Notation: denoted by a lowercase p.
  - E.g., $p(x) = 1$ indicates that $X = x$ is certain
  - E.g., $p(x) = 0$ indicates that $X = x$ is impossible

- Properties:
  - The domain D of p must be the set of all possible states of X
  - $\forall x \in D(X), 0 \leq p(x) \leq 1$
  - $\sum_{x \in D(X)} p(x) = 1$

Independence

- Two random variables X and Y are independent, if their probability distribution can be expressed as their products.

$$\forall x \in D(X), y \in D(Y), p(X = x, Y = y) = p(X = x)p(Y = y)$$

Independence

- Two random variables X and Y are independent, if their probability distribution can be expressed as their products.

$$\forall x \in D(X), y \in D(Y), p(X = x, Y = y) = p(X = x)p(Y = y)$$

- E.g., if a coin is tossed and a single 6-sided die is rolled, then the probability of landing on the head side of the coin and rolling a 3 on the die is:

Independence

- Two random variables $\mathtt{X}$ and $\mathtt{Y}$ are independent, if their probability distribution can be expressed as their products.

$$\forall \mathtt{x} \in \mathtt{D}(\mathtt{X}), \mathtt{y} \in \mathtt{D}(\mathtt{Y}), \mathtt{p}(\mathtt{X} = \mathtt{x}, \mathtt{Y} = \mathtt{y}) = \mathtt{p}(\mathtt{X} = \mathtt{x})\mathtt{p}(\mathtt{Y} = \mathtt{y})$$

- E.g., if a coin is tossed and a single 6-sided die is rolled, then the probability of landing on the head side of the coin and rolling a 3 on the die is:

$$\mathtt{p}(\mathtt{X} = \mathtt{head}, \mathtt{Y} = 3) = \mathtt{p}(\mathtt{X} = \mathtt{head})\mathtt{p}(\mathtt{Y} = 3) = \frac{1}{2} \times \frac{1}{6} = \frac{1}{12}$$

Conditional Probability

- Conditional probability: the probability of an event given that another event has occurred.

$$p(Y = y \mid X = x) = \frac{p(Y = y, X = x)}{p(X = x)}$$

Conditional Probability

- Conditional probability: the probability of an event given that another event has occurred.

$$p(Y = y \mid X = x) = \frac{p(Y = y, X = x)}{p(X = x)}$$

- E.g., if 60% of the class passed both labs and 80% of the class passed the first labs, then what percent of those who passed the first lab also passed the second lab?

Conditional Probability

- Conditional probability: the probability of an event given that another event has occurred.

$$p(Y = y \mid X = x) = \frac{p(Y = y, X = x)}{p(X = x)}$$

- E.g., if 60% of the class passed both labs and 80% of the class passed the first labs, then what percent of those who passed the first lab also passed the second lab?
  - E.g., X and Y random variables for the first and the second labs, respectively.

$$p(Y = \texttt{lab2} \mid X = \texttt{lab1}) = \frac{p(Y = \texttt{lab2}, X = \texttt{lab1})}{p(X = \texttt{lab1})} = \frac{0.6}{0.8} = \frac{3}{4}$$

Expectation

- The expected value of a random variable X with respect to a probability distribution $p(X)$ is the average value that X takes on when it is drawn from $p(X)$.

$$\mathbb{E}_{x \sim p}[X] = \sum_{x} p(x) x$$

Expectation

- The expected value of a random variable X with respect to a probability distribution $p(X)$ is the average value that X takes on when it is drawn from $p(X)$.

$$E_{x \sim p}[X] = \sum_{x} p(x)x$$

- E.g., If $X : \{1, 2, 3\}$, and $p(X = 1) = 0.3$, $p(X = 2) = 0.5$, $p(X = 3) = 0.2$

Expectation

- The expected value of a random variable $\mathtt{X}$ with respect to a probability distribution $\mathtt{p(X)}$ is the average value that $\mathtt{X}$ takes on when it is drawn from $\mathtt{p(X)}$.

$$\mathtt{E}_{\mathtt{x}\sim\mathtt{p}}[\mathtt{X}] = \sum_{\mathtt{x}} \mathtt{p(x)x}$$

- E.g., If $\mathtt{X} : \{1, 2, 3\}$, and $\mathtt{p(X} = 1) = 0.3$, $\mathtt{p(X} = 2) = 0.5$, $\mathtt{p(X} = 3) = 0.2$
  - $\mathtt{E[X]} = 0.3 \times 1 + 0.5 \times 2 + 0.2 \times 3 = 1.9$

Variance and Standard Deviation

- The variance gives a measure of how much the values of a random variable $\mathtt{X}$ vary as we sample it from its probability distribution $\mathtt{p}(\mathtt{X})$.

$$\mathtt{Var}(\mathtt{X}) = \mathtt{E}[(\mathtt{X} - \mathtt{E}[\mathtt{X}])^2]$$

$$\mathtt{Var}(\mathtt{X}) = \sum_{\mathtt{x}} \mathtt{p}(\mathtt{x})(\mathtt{x} - \mathtt{E}[\mathtt{X}])^2$$

Variance and Standard Deviation

- The variance gives a measure of how much the values of a random variable $\mathtt{X}$ vary as we sample it from its probability distribution $\mathtt{p}(\mathtt{X})$.

$$\mathtt{Var}(\mathtt{X}) = \mathtt{E}[(\mathtt{X} - \mathtt{E}[\mathtt{X}])^2]$$
$$\mathtt{Var}(\mathtt{X}) = \sum_{\mathtt{x}} \mathtt{p}(\mathtt{x})(\mathtt{x} - \mathtt{E}[\mathtt{X}])^2$$

- E.g., If $\mathtt{X} : \{1, 2, 3\}$, and $\mathtt{p}(\mathtt{X} = 1) = 0.3$, $\mathtt{p}(\mathtt{X} = 2) = 0.5$, $\mathtt{p}(\mathtt{X} = 3) = 0.2$

Variance and Standard Deviation

- The variance gives a measure of how much the values of a random variable X vary as we sample it from its probability distribution $p(X)$.

$$\text{Var}(X) = E[(X - E[X])^2]$$
$$\text{Var}(X) = \sum_x p(x)(x - E[X])^2$$

- E.g., If $X : \{1, 2, 3\}$, and $p(X = 1) = 0.3$, $p(X = 2) = 0.5$, $p(X = 3) = 0.2$
  - $E[X] = 0.3 \times 1 + 0.5 \times 2 + 0.2 \times 3 = 1.9$
  - $\text{Var}(X) = 0.3(1 - 1.9)^2 + 0.5(2 - 1.9)^2 + 0.2(3 - 1.9)^2 = 0.49$

Variance and Standard Deviation

- The variance gives a measure of how much the values of a random variable $\mathtt{X}$ vary as we sample it from its probability distribution $\mathtt{p(X)}$.

$$\mathtt{Var(X)} = \mathtt{E}[(\mathtt{X} - \mathtt{E[X]})^2]$$
$$\mathtt{Var(X)} = \sum_{\mathtt{x}} \mathtt{p(x)}(\mathtt{x} - \mathtt{E[X]})^2$$

- E.g., If $\mathtt{X} : \{1, 2, 3\}$, and $\mathtt{p}(\mathtt{X} = 1) = 0.3$, $\mathtt{p}(\mathtt{X} = 2) = 0.5$, $\mathtt{p}(\mathtt{X} = 3) = 0.2$
  - $\mathtt{E[X]} = 0.3 \times 1 + 0.5 \times 2 + 0.2 \times 3 = 1.9$
  - $\mathtt{Var(X)} = 0.3(1 - 1.9)^2 + 0.5(2 - 1.9)^2 + 0.2(3 - 1.9)^2 = 0.49$

- The standard deviation, shown by σ, is the square root of the variance.

Probability and Likelihood (1/2)

- Let $\mathtt{X} : \{\mathtt{x}^{(1)}, \mathtt{x}^{(2)}, \cdots, \mathtt{x}^{(m)}\}$ be a discrete random variable drawn independently from a distribution probability $\mathtt{p}$ depending on a parameter θ.

Probability and Likelihood (1/2)

- Let $\mathtt{X} : \{\mathtt{x}^{(1)}, \mathtt{x}^{(2)}, \cdots, \mathtt{x}^{(m)}\}$ be a discrete random variable drawn independently from a distribution probability $\mathtt{p}$ depending on a parameter θ.
  - For six tosses of a coin, $\mathtt{X} : \{\mathtt{h}, \mathtt{t}, \mathtt{t}, \mathtt{t}, \mathtt{h}, \mathtt{t}\}$, $\mathtt{h}$: head, and $\mathtt{t}$: tail.
  - Suppose you have a coin with probability θ to land heads and $(1 - \theta)$ to land tails.

Probability and Likelihood (1/2)

- Let $\mathtt{X} : \{\mathtt{x}^{(1)}, \mathtt{x}^{(2)}, \cdots, \mathtt{x}^{(m)}\}$ be a discrete random variable drawn independently from a distribution probability $\mathtt{p}$ depending on a parameter θ.
  - For six tosses of a coin, $\mathtt{X} : \{\mathtt{h}, \mathtt{t}, \mathtt{t}, \mathtt{t}, \mathtt{h}, \mathtt{t}\}$, $\mathtt{h}$: head, and $\mathtt{t}$: tail.
  - Suppose you have a coin with probability θ to land heads and $(1 - \theta)$ to land tails.

- $\mathtt{p}(\mathtt{X} \mid \theta = \frac{2}{3})$ is the probability of $\mathtt{X}$ given $\theta = \frac{2}{3}$.

Probability and Likelihood (1/2)

- Let $\mathtt{X} : \{\mathtt{x}^{(1)}, \mathtt{x}^{(2)}, \cdots, \mathtt{x}^{(m)}\}$ be a discrete random variable drawn independently from a distribution probability $\mathtt{p}$ depending on a parameter θ.
  - For six tosses of a coin, $\mathtt{X} : \{\mathtt{h}, \mathtt{t}, \mathtt{t}, \mathtt{t}, \mathtt{h}, \mathtt{t}\}$, $\mathtt{h}$: head, and $\mathtt{t}$: tail.
  - Suppose you have a coin with probability θ to land heads and $(1 - \theta)$ to land tails.

- $\mathtt{p}(\mathtt{X} \mid \theta = \frac{2}{3})$ is the probability of $\mathtt{X}$ given $\theta = \frac{2}{3}$.

- $\mathtt{p}(\mathtt{X} = \mathtt{h} \mid \theta)$ is the likelihood of θ given $\mathtt{X} = \mathtt{h}$.

Probability and Likelihood (1/2)

- Let $\mathtt{X} : \{\mathtt{x}^{(1)}, \mathtt{x}^{(2)}, \cdots, \mathtt{x}^{(m)}\}$ be a discrete random variable drawn independently from a distribution probability $\mathtt{p}$ depending on a parameter θ.
  - For six tosses of a coin, $\mathtt{X} : \{\mathtt{h}, \mathtt{t}, \mathtt{t}, \mathtt{t}, \mathtt{h}, \mathtt{t}\}$, $\mathtt{h}$: head, and $\mathtt{t}$: tail.
  - Suppose you have a coin with probability θ to land heads and $(1 - \theta)$ to land tails.

- $\mathtt{p}(\mathtt{X} \mid \theta = \frac{2}{3})$ is the probability of $\mathtt{X}$ given $\theta = \frac{2}{3}$.

- $\mathtt{p}(\mathtt{X} = \mathtt{h} \mid \theta)$ is the likelihood of θ given $\mathtt{X} = \mathtt{h}$.

- Likelihood ($\mathtt{L}$): a function of the parameters (θ) of a probability model, given specific observed data, e.g., $\mathtt{X} = \mathtt{h}$.

$$\mathtt{L}(\theta \mid \mathtt{X}) = \mathtt{p}(\mathtt{X} \mid \theta)$$

Probability and Likelihood (2/2)

- The likelihood differs from that of a probability.

Probability and Likelihood (2/2)

- The likelihood differs from that of a probability.
- A probability $p(X \mid \theta)$ refers to the occurrence of future events.

Probability and Likelihood (2/2)

- The likelihood differs from that of a probability.
- A probability $p(X \mid \theta)$ refers to the occurrence of future events.
- A likelihood $L(\theta \mid X)$ refers to past events with known outcomes.

Likelihood and Log-Likelihood (1/2)

- If samples in $\mathtt{X}$ are independent we have:

$$
\begin{aligned}
\mathtt{L}(\theta \mid \mathtt{X}) = \mathtt{p}(\mathtt{X} \mid \theta) &= \mathtt{p}(\mathtt{x}^{(1)}, \mathtt{x}^{(2)}, \cdots, \mathtt{x}^{(m)} \mid \theta) \\
&= \mathtt{p}(\mathtt{x}^{(1)} \mid \theta)\mathtt{p}(\mathtt{x}^{(2)} \mid \theta) \cdots \mathtt{p}(\mathtt{x}^{(m)} \mid \theta) = \prod_{\mathtt{i}=1}^{\mathtt{m}} \mathtt{p}(\mathtt{x}^{(\mathtt{i})} \mid \theta)
\end{aligned}
$$

Likelihood and Log-Likelihood (1/2)

- If samples in $\mathtt{X}$ are independent we have:

$$\mathtt{L}(\theta \mid \mathtt{X}) = \mathtt{p}(\mathtt{X} \mid \theta) = \mathtt{p}(\mathtt{x}^{(1)}, \mathtt{x}^{(2)}, \cdots, \mathtt{x}^{(m)} \mid \theta)$$
$$= \mathtt{p}(\mathtt{x}^{(1)} \mid \theta)\mathtt{p}(\mathtt{x}^{(2)} \mid \theta) \cdots \mathtt{p}(\mathtt{x}^{(m)} \mid \theta) = \prod_{\mathtt{i}=1}^{\mathtt{m}} \mathtt{p}(\mathtt{x}^{(\mathtt{i})} \mid \theta)$$

- E.g., six tosses of a coin, with the following model:
  - Data: $\mathtt{X} : \{\mathtt{h}, \mathtt{t}, \mathtt{t}, \mathtt{t}, \mathtt{h}, \mathtt{t}\}$
  - Possible outcomes: h with probability of θ, and t with probability $(1 - \theta)$.

Likelihood and Log-Likelihood (1/2)

- If samples in $\mathtt{X}$ are independent we have:

$$\begin{aligned} \mathtt{L}(\theta \mid \mathtt{X}) = \mathtt{p}(\mathtt{X} \mid \theta) &= \mathtt{p}(\mathtt{x}^{(1)}, \mathtt{x}^{(2)}, \cdots, \mathtt{x}^{(m)} \mid \theta) \\ &= \mathtt{p}(\mathtt{x}^{(1)} \mid \theta)\mathtt{p}(\mathtt{x}^{(2)} \mid \theta) \cdots \mathtt{p}(\mathtt{x}^{(m)} \mid \theta) = \prod_{\mathtt{i}=1}^{\mathtt{m}} \mathtt{p}(\mathtt{x}^{(\mathtt{i})} \mid \theta) \end{aligned}$$

- E.g., six tosses of a coin, with the following model:
  - Data: $\mathtt{X} : \{\mathtt{h}, \mathtt{t}, \mathtt{t}, \mathtt{t}, \mathtt{h}, \mathtt{t}\}$
  - Possible outcomes: h with probability of θ, and t with probability $(1 - \theta)$.

$$\begin{aligned} \mathtt{L}(\theta \mid \mathtt{X}) &= \mathtt{p}(\mathtt{X} \mid \theta) \\ &= \mathtt{p}(\mathtt{X} = \mathtt{h} \mid \theta)\mathtt{p}(\mathtt{X} = \mathtt{t} \mid \theta)\mathtt{p}(\mathtt{X} = \mathtt{t} \mid \theta)\mathtt{p}(\mathtt{X} = \mathtt{t} \mid \theta)\mathtt{p}(\mathtt{X} = \mathtt{h} \mid \theta)\mathtt{p}(\mathtt{X} = \mathtt{t} \mid \theta) \\ &= \theta(1 - \theta)(1 - \theta)(1 - \theta)\theta(1 - \theta) \\ &= \theta^2(1 - \theta)^4 \end{aligned}$$

Likelihood and Log-Likelihood (2/2)

- The probability product is prone to numerical underflow.

$$\mathtt{L}(\theta \mid \mathtt{X}) = \mathtt{p}(\mathtt{X} \mid \theta) = \prod_{\mathtt{i}=1}^{\mathtt{m}} \mathtt{p}(\mathtt{x}^{(\mathtt{i})} \mid \theta)$$

Likelihood and Log-Likelihood (2/2)

- The probability product is prone to numerical underflow.

$$\mathtt{L}(\theta \mid \mathtt{X}) = \mathtt{p}(\mathtt{X} \mid \theta) = \prod_{\mathtt{i}=1}^{\mathtt{m}} \mathtt{p}(\mathtt{x}^{(\mathtt{i})} \mid \theta)$$

- To overcome this problem we can use the logarithm of the likelihood.

$$\mathtt{logL}(\theta \mid \mathtt{X}) = \mathtt{log} \prod_{\mathtt{i}=1}^{\mathtt{m}} \mathtt{p}(\mathtt{x}^{(\mathtt{i})} \mid \theta) = \sum_{\mathtt{i}=1}^{\mathtt{m}} \mathtt{logp}(\mathtt{x}^{(\mathtt{i})} \mid \theta)$$

Negative Log-Likelihood

- Likelihood: $\mathtt{L}(\theta \mid \mathtt{X}) = \prod_{\mathtt{i}=1}^{\mathtt{m}} \mathtt{p}(\mathtt{x}^{(\mathtt{i})} \mid \theta)$

Negative Log-Likelihood

- Likelihood: $\mathtt{L}(\theta \mid \mathtt{X}) = \prod_{\mathtt{i}=1}^{\mathtt{m}} \mathtt{p}(\mathtt{x}^{(\mathtt{i})} \mid \theta)$
- Log-Likelihood: $\mathtt{logL}(\theta \mid \mathtt{X}) = \mathtt{log} \prod_{\mathtt{i}=1}^{\mathtt{m}} \mathtt{p}(\mathtt{x}^{(\mathtt{i})} \mid \theta) = \sum_{\mathtt{i}=1}^{\mathtt{m}} \mathtt{logp}(\mathtt{x}^{(\mathtt{i})} \mid \theta)$

Negative Log-Likelihood

- Likelihood: $\mathtt{L}(\theta \mid \mathtt{X}) = \prod_{\mathtt{i}=1}^{\mathtt{m}} \mathtt{p}(\mathtt{x}^{(\mathtt{i})} \mid \theta)$
- Log-Likelihood: $\mathtt{logL}(\theta \mid \mathtt{X}) = \mathtt{log} \prod_{\mathtt{i}=1}^{\mathtt{m}} \mathtt{p}(\mathtt{x}^{(\mathtt{i})} \mid \theta) = \sum_{\mathtt{i}=1}^{\mathtt{m}} \mathtt{logp}(\mathtt{x}^{(\mathtt{i})} \mid \theta)$
- Negative Log-Likelihood: $-\mathtt{logL}(\theta \mid \mathtt{X}) = -\sum_{\mathtt{i}=1}^{\mathtt{m}} \mathtt{logp}(\mathtt{x}^{(\mathtt{i})} \mid \theta)$

Negative Log-Likelihood

- Likelihood: $\mathtt{L}(\theta \mid \mathtt{X}) = \prod_{\mathtt{i}=1}^{\mathtt{m}} \mathtt{p}(\mathtt{x}^{(\mathtt{i})} \mid \theta)$
- Log-Likelihood: $\mathtt{log}\mathtt{L}(\theta \mid \mathtt{X}) = \mathtt{log} \prod_{\mathtt{i}=1}^{\mathtt{m}} \mathtt{p}(\mathtt{x}^{(\mathtt{i})} \mid \theta) = \sum_{\mathtt{i}=1}^{\mathtt{m}} \mathtt{log}\mathtt{p}(\mathtt{x}^{(\mathtt{i})} \mid \theta)$
- Negative Log-Likelihood: $-\mathtt{log}\mathtt{L}(\theta \mid \mathtt{X}) = -\sum_{\mathtt{i}=1}^{\mathtt{m}} \mathtt{log}\mathtt{p}(\mathtt{x}^{(\mathtt{i})} \mid \theta)$
- Negative log-likelihood is also called the cross-entropy

Cross-Entropy

- Coss-entropy: quantify the difference (error) between two probability distributions.

- How close is the predicted distribution to the true distribution?

$$H(p, q) = -\sum_{x} p(x) \log(q(x))$$

- Where p is the true distribution, and q the predicted distribution.

Cross-Entropy - Example

- Six tosses of a coin: $\mathtt{X} : \{\mathtt{h}, \mathtt{t}, \mathtt{t}, \mathtt{t}, \mathtt{h}, \mathtt{t}\}$
- The true distribution $\mathtt{p}$: $\mathtt{p}(\mathtt{h}) = \frac{2}{6}$ and $\mathtt{p}(\mathtt{t}) = \frac{4}{6}$
- The predicted distribution $\mathtt{q}$: h with probability of θ, and t with probability $(1 - \theta)$.

Cross-Entropy - Example

- Six tosses of a coin: $\mathtt{X} : \{\mathtt{h}, \mathtt{t}, \mathtt{t}, \mathtt{t}, \mathtt{h}, \mathtt{t}\}$
- The true distribution $\mathtt{p}$: $\mathtt{p}(\mathtt{h}) = \frac{2}{6}$ and $\mathtt{p}(\mathtt{t}) = \frac{4}{6}$
- The predicted distribution $\mathtt{q}$: h with probability of θ, and t with probability $(1 - \theta)$.
- Cross entropy: $\mathtt{H}(\mathtt{p}, \mathtt{q}) = -\sum_{\mathtt{x}} \mathtt{p}(\mathtt{x})\mathtt{log}(\mathtt{q}(\mathtt{x}))$
  $= -\mathtt{p}(\mathtt{h})\mathtt{log}(\mathtt{q}(\mathtt{h})) - \mathtt{p}(\mathtt{t})\mathtt{log}(\mathtt{q}(\mathtt{t})) = -\frac{2}{6}\mathtt{log}(\theta) - \frac{4}{6}\mathtt{log}(1 - \theta)$

Cross-Entropy - Example

- Six tosses of a coin: $\mathtt{X} : \{\mathtt{h}, \mathtt{t}, \mathtt{t}, \mathtt{t}, \mathtt{h}, \mathtt{t}\}$
- The true distribution $\mathtt{p}$: $\mathtt{p}(\mathtt{h}) = \frac{2}{6}$ and $\mathtt{p}(\mathtt{t}) = \frac{4}{6}$
- The predicted distribution $\mathtt{q}$: h with probability of θ, and t with probability $(1 - \theta)$.
- Cross entropy: $\mathtt{H}(\mathtt{p}, \mathtt{q}) = -\sum_{\mathtt{x}} \mathtt{p}(\mathtt{x})\mathtt{log}(\mathtt{q}(\mathtt{x}))$
  $= -\mathtt{p}(\mathtt{h})\mathtt{log}(\mathtt{q}(\mathtt{h})) - \mathtt{p}(\mathtt{t})\mathtt{log}(\mathtt{q}(\mathtt{t})) = -\frac{2}{6}\mathtt{log}(\theta) - \frac{4}{6}\mathtt{log}(1 - \theta)$
- Likelihood: $\theta^2(1 - \theta)^4$

Cross-Entropy - Example

- Six tosses of a coin: $\mathtt{X} : \{\mathtt{h}, \mathtt{t}, \mathtt{t}, \mathtt{t}, \mathtt{h}, \mathtt{t}\}$
- The true distribution $\mathtt{p}$: $\mathtt{p}(\mathtt{h}) = \frac{2}{6}$ and $\mathtt{p}(\mathtt{t}) = \frac{4}{6}$
- The predicted distribution $\mathtt{q}$: h with probability of θ, and t with probability $(1 - \theta)$.
- Cross entropy: $\mathtt{H}(\mathtt{p}, \mathtt{q}) = -\sum_{\mathtt{x}} \mathtt{p}(\mathtt{x})\mathtt{log}(\mathtt{q}(\mathtt{x}))$
  $= -\mathtt{p}(\mathtt{h})\mathtt{log}(\mathtt{q}(\mathtt{h})) - \mathtt{p}(\mathtt{t})\mathtt{log}(\mathtt{q}(\mathtt{t})) = -\frac{2}{6}\mathtt{log}(\theta) - \frac{4}{6}\mathtt{log}(1 - \theta)$
- Likelihood: $\theta^2(1 - \theta)^4$
- Negative log likelihood: $-\mathtt{log}(\theta^2(1 - \theta)^4) = -2\mathtt{log}(\theta) - 4\mathtt{log}(1 - \theta)$

Linear Regression

Let's Start with an Example

The Housing Price Example (1/3)

- Given the dataset of `m` houses.

Living area	No. of bedrooms	Price
2104	3	400
1600	3	330
2400	3	369
⋮	⋮	⋮

The Housing Price Example (1/3)

- Given the dataset of `m` houses.

Living area	No. of bedrooms	Price
2104	3	400
1600	3	330
2400	3	369
⋮	⋮	⋮

- Predict the prices of other houses, as a function of the size of living area and number of bedrooms?

The Housing Price Example (2/3)

Living area	No. of bedrooms	Price
2104	3	400
1600	3	330
2400	3	369
⋮	⋮	⋮

The Housing Price Example (2/3)

Living area	No. of bedrooms	Price
2104	3	400
1600	3	330
2400	3	369
⋮	⋮	⋮

$$\mathbf{x}^{(1)} = \begin{bmatrix} 2104 \\ 3 \end{bmatrix} \quad y^{(1)} = 400 \qquad \mathbf{x}^{(2)} = \begin{bmatrix} 1600 \\ 3 \end{bmatrix} \quad y^{(2)} = 330 \qquad \mathbf{x}^{(3)} = \begin{bmatrix} 2400 \\ 3 \end{bmatrix} \quad y^{(3)} = 369$$

The Housing Price Example (2/3)

Living area	No. of bedrooms	Price
2104	3	400
1600	3	330
2400	3	369
$\vdots$	$\vdots$	$\vdots$

$$\mathbf{x}^{(1)} = \begin{bmatrix} 2104 \\ 3 \end{bmatrix} \quad y^{(1)} = 400 \qquad \mathbf{x}^{(2)} = \begin{bmatrix} 1600 \\ 3 \end{bmatrix} \quad y^{(2)} = 330 \qquad \mathbf{x}^{(3)} = \begin{bmatrix} 2400 \\ 3 \end{bmatrix} \quad y^{(3)} = 369$$

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}^{(1)\intercal} \\ \mathbf{x}^{(2)\intercal} \\ \mathbf{x}^{(3)\intercal} \\ \vdots \end{bmatrix} = \begin{bmatrix} 2104 & 3 \\ 1600 & 3 \\ 2400 & 3 \\ \vdots & \vdots \end{bmatrix} \qquad \mathbf{y} = \begin{bmatrix} 400 \\ 330 \\ 369 \\ \vdots \end{bmatrix}$$

The Housing Price Example (2/3)

Living area	No. of bedrooms	Price
2104	3	400
1600	3	330
2400	3	369
$\vdots$	$\vdots$	$\vdots$

$$\mathbf{x}^{(1)} = \begin{bmatrix} 2104 \\ 3 \end{bmatrix} \quad y^{(1)} = 400 \qquad \mathbf{x}^{(2)} = \begin{bmatrix} 1600 \\ 3 \end{bmatrix} \quad y^{(2)} = 330 \qquad \mathbf{x}^{(3)} = \begin{bmatrix} 2400 \\ 3 \end{bmatrix} \quad y^{(3)} = 369$$

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}^{(1)\intercal} \\ \mathbf{x}^{(2)\intercal} \\ \mathbf{x}^{(3)\intercal} \\ \vdots \end{bmatrix} = \begin{bmatrix} 2104 & 3 \\ 1600 & 3 \\ 2400 & 3 \\ \vdots & \vdots \end{bmatrix} \qquad \mathbf{y} = \begin{bmatrix} 400 \\ 330 \\ 369 \\ \vdots \end{bmatrix}$$

- $\mathbf{x}^{(i)} \in \mathbb{R}^2$: $x_1^{(i)}$ is the living area, and $x_2^{(i)}$ is the number of bedrooms of the ith house in the training set.

The Housing Price Example (3/3)

Living area	No. of bedrooms	Price
2104	3	400
1600	3	330
2400	3	369
⋮	⋮	⋮

- Predict the prices of other houses $\hat{y}$ as a function of the size of their living areas x_1, and number of bedrooms x_2, i.e., $\hat{y} = f(x_1, x_2)$
- E.g., what is $\hat{y}$, if $x_1 = 4000$ and $x_2 = 4$?

The Housing Price Example (3/3)

Living area	No. of bedrooms	Price
2104	3	400
1600	3	330
2400	3	369
⋮	⋮	⋮

- Predict the prices of other houses $\hat{y}$ as a function of the size of their living areas x_1, and number of bedrooms x_2, i.e., $\hat{y} = f(x_1, x_2)$
- E.g., what is $\hat{y}$, if $x_1 = 4000$ and $x_2 = 4$?
- As an initial choice: $\hat{y} = f_w(\mathbf{x}) = w_1 x_1 + w_2 x_2$

Linear Regression (1/2)

- Our goal: to build a system that takes input $\mathbf{x} \in \mathbb{R}^\mathtt{n}$ and predicts output $\hat{\mathtt{y}} \in \mathbb{R}$.

Linear Regression (1/2)

- Our goal: to build a system that takes input $\mathbf{x} \in \mathbb{R}^n$ and predicts output $\hat{y} \in \mathbb{R}$.
- In linear regression, the output $\hat{y}$ is a linear function of the input $\mathbf{x}$.

$$\hat{y} = f_w(\mathbf{x}) = w_1 x_1 + w_2 x_2 + \cdots + w_n x_n$$
$$\hat{y} = \mathbf{w}^\intercal \mathbf{x}$$

Linear Regression (1/2)

- Our goal: to build a system that takes input $\mathbf{x} \in \mathbb{R}^n$ and predicts output $\hat{y} \in \mathbb{R}$.
- In linear regression, the output $\hat{y}$ is a linear function of the input $\mathbf{x}$.

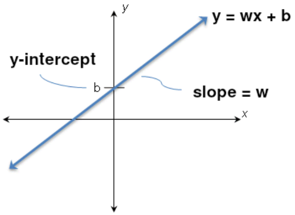
$$\hat{y} = f_w(\mathbf{x}) = w_1 x_1 + w_2 x_2 + \cdots + w_n x_n$$
$$\hat{y} = \mathbf{w}^\intercal \mathbf{x}$$

  - $\hat{y}$: the predicted value
  - x_i: the ith feature value
  - w_j: the jth model parameter ($\mathbf{w} \in \mathbb{R}^n$)
  - n: the number of features

Linear Regression (2/2)

- Linear regression often has one additional parameter, called intercept b:

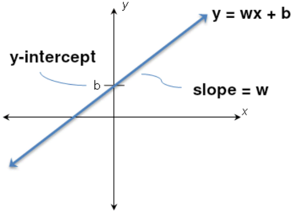
$$\hat{y} = \mathbf{w}^\mathsf{T}\mathbf{x} + b$$



Linear Regression (2/2)

- Linear regression often has one additional parameter, called intercept b:

$$\hat{y} = \mathbf{w}^\mathsf{T}\mathbf{x} + b$$



- Instead of adding the bias parameter b, we can augment $\mathbf{x}$ with an extra entry that is always set to 1.

$$\hat{y} = f_w(\mathbf{x}) = w_0 x_0 + w_1 x_1 + w_2 x_2 + \cdots + w_n x_n, \text{ where } x_0 = 1$$

Linear Regression - Model Parameters

- Parameters $\mathbf{w} \in \mathbb{R}^n$ are values that control the behavior of the model.

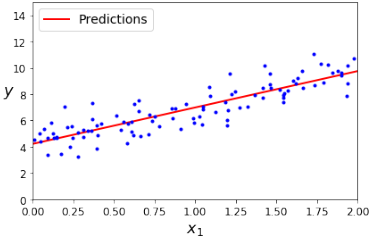
Linear Regression - Model Parameters

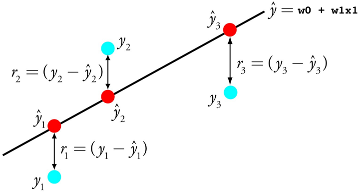
- Parameters $\mathbf{w} \in \mathbb{R}^{\mathbf{n}}$ are values that control the behavior of the model.
- $\mathbf{w}$ are a set of weights that determine how each feature affects the prediction.

How to Learn Model Parameters $\mathbf{w}$?


LEARNING?
I LOVE LEARNING

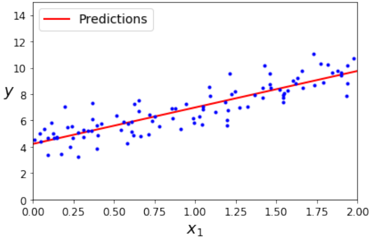
Linear Regression - Cost Function (1/2)

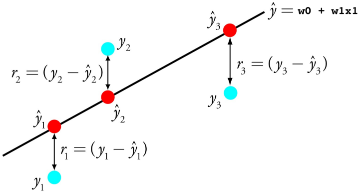




- One reasonable model should make $\hat{y}$ close to y, at least for the training dataset.

Linear Regression - Cost Function (1/2)

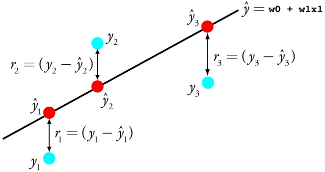




- One reasonable model should make $\hat{y}$ close to y, at least for the training dataset.
- Residual: the difference between the dependent variable y and the predicted value $\hat{y}$.

$$r^{(i)} = y^{(i)} - \hat{y}^{(i)}$$

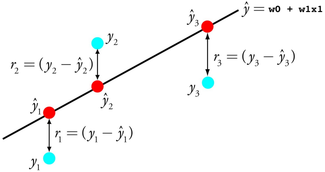
Linear Regression - Cost Function (2/2)



- Cost function $\mathtt{J}(\mathbf{w})$
  - For each value of the $\mathbf{w}$, it measures how close the $\hat{\mathtt{y}}^{(\mathtt{i})}$ is to the corresponding $\mathtt{y}^{(\mathtt{i})}$.
  - We can define $\mathtt{J}(\mathbf{w})$ as the mean squared error (MSE):

$$\mathtt{J}(\mathbf{w}) = \mathtt{MSE}(\mathbf{w}) = \frac{1}{\mathtt{m}} \sum_{\mathtt{i}}^{\mathtt{m}} (\hat{\mathtt{y}}^{(\mathtt{i})} - \mathtt{y}^{(\mathtt{i})})^2$$

Linear Regression - Cost Function (2/2)



- Cost function $\mathtt{J}(\mathbf{w})$
  - For each value of the $\mathbf{w}$, it measures how close the $\hat{\mathtt{y}}^{(\mathtt{i})}$ is to the corresponding $\mathtt{y}^{(\mathtt{i})}$.
  - We can define $\mathtt{J}(\mathbf{w})$ as the mean squared error (MSE):

$$\mathtt{J}(\mathbf{w}) = \mathtt{MSE}(\mathbf{w}) = \frac{1}{\mathtt{m}} \sum_{\mathtt{i}}^{\mathtt{m}} (\hat{\mathtt{y}}^{(\mathtt{i})} - \mathtt{y}^{(\mathtt{i})})^2$$

$$= \mathtt{E}[(\hat{\mathtt{y}} - \mathtt{y})^2] = \frac{1}{\mathtt{m}} ||\hat{\mathbf{y}} - \mathbf{y}||_2^2$$

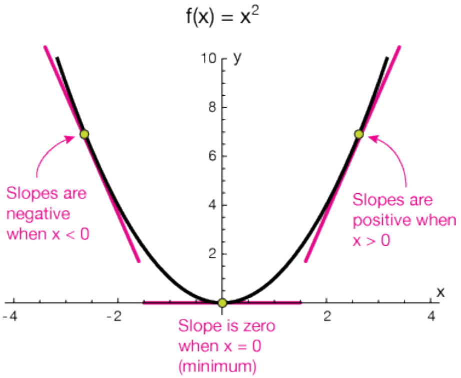
How to Learn Model Parameters?

- We want to choose $\mathbf{w}$ so as to minimize $J(\mathbf{w})$.
- Two approaches to find $\mathbf{w}$:
  - Normal equation
  - Gradient descent

Normal Equation

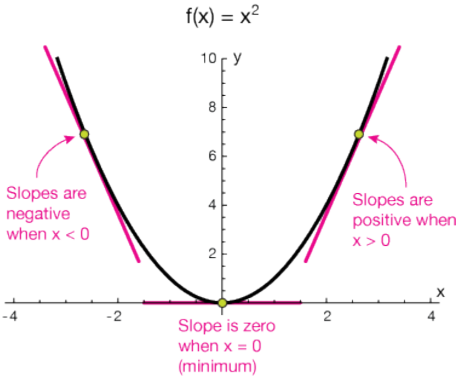
Derivatives and Gradient (1/3)

- The first derivative of $f(x)$, shown as $f'(x)$, shows the slope of the tangent line to the function at the poa x.



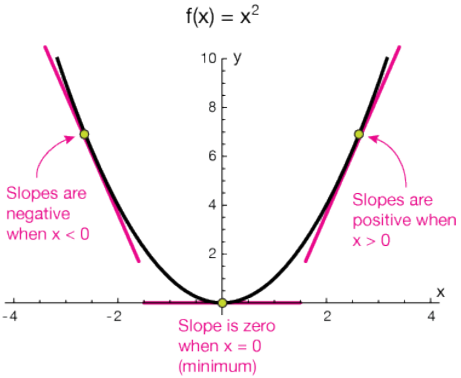
Derivatives and Gradient (1/3)

- The first derivative of $f(x)$, shown as $f'(x)$, shows the slope of the tangent line to the function at the poa x.
- $f(x) = x^2 \Rightarrow f'(x) = 2x$



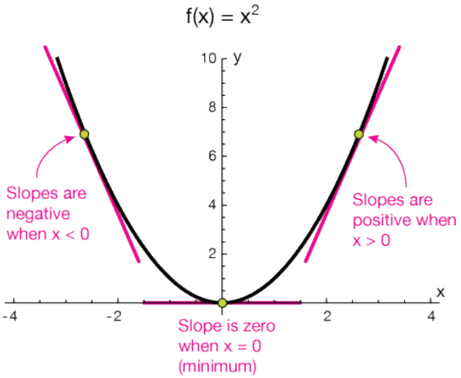
Derivatives and Gradient (1/3)

- The first derivative of $f(x)$, shown as $f'(x)$, shows the slope of the tangent line to the function at the poa x.
- $f(x) = x^2 \Rightarrow f'(x) = 2x$
- If $f(x)$ is increasing, then $f'(x) > 0$



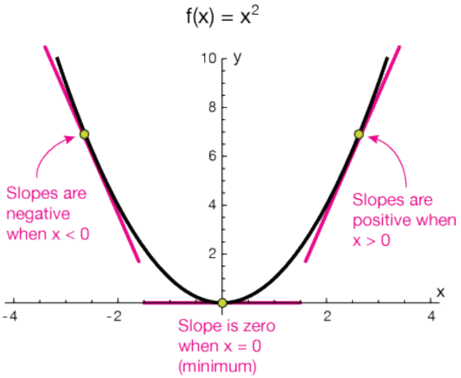
Derivatives and Gradient (1/3)

- The first derivative of $f(x)$, shown as $f'(x)$, shows the slope of the tangent line to the function at the poa x.
- $f(x) = x^2 \Rightarrow f'(x) = 2x$
- If $f(x)$ is increasing, then $f'(x) > 0$
- If $f(x)$ is decreasing, then $f'(x) < 0$



Derivatives and Gradient (1/3)

- The first derivative of $f(x)$, shown as $f'(x)$, shows the slope of the tangent line to the function at the poa x.
- $f(x) = x^2 \Rightarrow f'(x) = 2x$
- If $f(x)$ is increasing, then $f'(x) > 0$
- If $f(x)$ is decreasing, then $f'(x) < 0$
- If $f(x)$ is at local minimum/maximum, then $f'(x) = 0$



Derivatives and Gradient (2/3)

- What if a function has multiple arguments, e.g., $f(x_1, x_2, \cdots, x_n)$

Derivatives and Gradient (2/3)

- What if a function has multiple arguments, e.g., $f(x_1, x_2, \cdots, x_n)$

- Partial derivatives: the derivative with respect to a particular argument.
  - $\frac{\partial f}{\partial x_1}$, the derivative with respect to x_1
  - $\frac{\partial f}{\partial x_2}$, the derivative with respect to x_2

Derivatives and Gradient (2/3)

- What if a function has multiple arguments, e.g., $f(x_1, x_2, \cdots, x_n)$

- Partial derivatives: the derivative with respect to a particular argument.
  - $\frac{\partial f}{\partial x_1}$, the derivative with respect to x_1
  - $\frac{\partial f}{\partial x_2}$, the derivative with respect to x_2

- $\frac{\partial f}{\partial x_i}$: shows how much the function f will change, if we change x_i.

Derivatives and Gradient (2/3)

- What if a function has multiple arguments, e.g., $f(x_1, x_2, \cdots, x_n)$

- Partial derivatives: the derivative with respect to a particular argument.
  - $\frac{\partial f}{\partial x_1}$, the derivative with respect to x_1
  - $\frac{\partial f}{\partial x_2}$, the derivative with respect to x_2

- $\frac{\partial f}{\partial x_i}$: shows how much the function f will change, if we change x_i.

- Gradient: the vector of all partial derivatives for a function f.

$$\nabla_{\mathbf{x}} f(\mathbf{x}) = \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \\ \vdots \\ \frac{\partial f}{\partial x_n} \end{bmatrix}$$

Derivatives and Gradient (3/3)

- What is the gradient of $f(x_1, x_2, x_3) = x_1 - x_1 x_2 + x_3^2$?

Derivatives and Gradient (3/3)

- What is the gradient of $f(x_1, x_2, x_3) = x_1 - x_1 x_2 + x_3^2$?

$$\nabla_{\mathbf{x}} f(\mathbf{x}) = \begin{bmatrix} \frac{\partial}{\partial x_1}(x_1 - x_1 x_2 + x_3^2) \\ \frac{\partial}{\partial x_2}(x_1 - x_1 x_2 + x_3^2) \\ \frac{\partial}{\partial x_3}(x_1 - x_1 x_2 + x_3^2) \end{bmatrix} = \begin{bmatrix} 1 - x_2 \\ -x_1 \\ 2x_3 \end{bmatrix}$$

Normal Equation (1/2)

- To minimize $J(\mathbf{w})$, we can simply solve for where its gradient is 0: $\nabla_{\mathbf{w}} J(\mathbf{w}) = 0$

$$\hat{y} = \mathbf{w}^\intercal \mathbf{x}$$

Normal Equation (1/2)

- To minimize $J(\mathbf{w})$, we can simply solve for where its gradient is 0: $\nabla_{\mathbf{w}} J(\mathbf{w}) = 0$

$$\hat{y} = \mathbf{w}^\top \mathbf{x}$$

$$\mathbf{X} = \begin{bmatrix} [x_1^{(1)}, x_2^{(1)}, \cdots, x_n^{(1)}] \\ [x_1^{(2)}, x_2^{(2)}, \cdots, x_n^{(2)}] \\ \vdots \\ [x_1^{(m)}, x_2^{(m)}, \cdots, x_n^{(m)}] \end{bmatrix} = \begin{bmatrix} \mathbf{x}^{(1)\top} \\ \mathbf{x}^{(2)\top} \\ \vdots \\ \mathbf{x}^{(m)\top} \end{bmatrix} \qquad \hat{\mathbf{y}} = \begin{bmatrix} \hat{y}^{(1)} \\ \hat{y}^{(2)} \\ \vdots \\ \hat{y}^{(m)} \end{bmatrix}$$

Normal Equation (1/2)

- To minimize $\mathtt{J}(\mathbf{w})$, we can simply solve for where its gradient is 0: $\nabla_{\mathbf{w}}\mathtt{J}(\mathbf{w}) = 0$

$$\hat{\mathtt{y}} = \mathbf{w}^\intercal \mathbf{x}$$

$$\mathbf{X} = \begin{bmatrix} [\mathtt{x}_1^{(1)}, \mathtt{x}_2^{(1)}, \cdots, \mathtt{x}_n^{(1)}] \\ [\mathtt{x}_1^{(2)}, \mathtt{x}_2^{(2)}, \cdots, \mathtt{x}_n^{(2)}] \\ \vdots \\ [\mathtt{x}_1^{(m)}, \mathtt{x}_2^{(m)}, \cdots, \mathtt{x}_n^{(m)}] \end{bmatrix} = \begin{bmatrix} \mathbf{x}^{(1)\intercal} \\ \mathbf{x}^{(2)\intercal} \\ \vdots \\ \mathbf{x}^{(m)\intercal} \end{bmatrix} \qquad \hat{\mathbf{y}} = \begin{bmatrix} \hat{\mathtt{y}}^{(1)} \\ \hat{\mathtt{y}}^{(2)} \\ \vdots \\ \hat{\mathtt{y}}^{(m)} \end{bmatrix}$$

$$\hat{\mathbf{y}} = \mathbf{w}^\intercal \mathbf{X}^\intercal \text{ or } \hat{\mathbf{y}} = \mathbf{X}\mathbf{w}$$

Normal Equation (2/2)

- To minimize $\mathtt{J}(\mathbf{w})$, we can simply solve for where its gradient is 0: $\nabla_{\mathbf{w}}\mathtt{J}(\mathbf{w}) = 0$

Normal Equation (2/2)

- To minimize $J(\mathbf{w})$, we can simply solve for where its gradient is 0: $\nabla_{\mathbf{w}} J(\mathbf{w}) = 0$

$$J(\mathbf{w}) = \frac{1}{m}||\hat{\mathbf{y}} - \mathbf{y}||_2^2, \nabla_{\mathbf{w}} J(\mathbf{w}) = 0$$

Normal Equation (2/2)

- To minimize $J(\mathbf{w})$, we can simply solve for where its gradient is 0: $\nabla_{\mathbf{w}} J(\mathbf{w}) = 0$

$$J(\mathbf{w}) = \frac{1}{m}||\hat{\mathbf{y}} - \mathbf{y}||_2^2, \nabla_{\mathbf{w}} J(\mathbf{w}) = 0$$

$$\Rightarrow \nabla_{\mathbf{w}} \frac{1}{m}||\hat{\mathbf{y}} - \mathbf{y}||_2^2 = 0$$

Normal Equation (2/2)

- To minimize $J(\mathbf{w})$, we can simply solve for where its gradient is 0: $\nabla_{\mathbf{w}} J(\mathbf{w}) = 0$

$$J(\mathbf{w}) = \frac{1}{m}||\hat{\mathbf{y}} - \mathbf{y}||_2^2, \nabla_{\mathbf{w}} J(\mathbf{w}) = 0$$
$$\Rightarrow \nabla_{\mathbf{w}} \frac{1}{m}||\hat{\mathbf{y}} - \mathbf{y}||_2^2 = 0$$
$$\Rightarrow \nabla_{\mathbf{w}} \frac{1}{m}||\mathbf{X}\mathbf{w} - \mathbf{y}||_2^2 = 0$$

Normal Equation (2/2)

- To minimize $J(\mathbf{w})$, we can simply solve for where its gradient is 0: $\nabla_{\mathbf{w}} J(\mathbf{w}) = 0$

$$J(\mathbf{w}) = \frac{1}{m}||\hat{\mathbf{y}} - \mathbf{y}||_2^2, \nabla_{\mathbf{w}} J(\mathbf{w}) = 0$$
$$\Rightarrow \nabla_{\mathbf{w}} \frac{1}{m}||\hat{\mathbf{y}} - \mathbf{y}||_2^2 = 0$$
$$\Rightarrow \nabla_{\mathbf{w}} \frac{1}{m}||\mathbf{X}\mathbf{w} - \mathbf{y}||_2^2 = 0$$
$$\Rightarrow \nabla_{\mathbf{w}} (\mathbf{X}\mathbf{w} - \mathbf{y})^\intercal (\mathbf{X}\mathbf{w} - \mathbf{y}) = 0$$

Normal Equation (2/2)

- To minimize $\mathtt{J}(\mathbf{w})$, we can simply solve for where its gradient is 0: $\nabla_{\mathbf{w}}\mathtt{J}(\mathbf{w}) = 0$

$$\mathtt{J}(\mathbf{w}) = \frac{1}{\mathtt{m}}||\hat{\mathbf{y}} - \mathbf{y}||_2^2, \nabla_{\mathbf{w}}\mathtt{J}(\mathbf{w}) = 0$$

$$\Rightarrow \nabla_{\mathbf{w}}\frac{1}{\mathtt{m}}||\hat{\mathbf{y}} - \mathbf{y}||_2^2 = 0$$

$$\Rightarrow \nabla_{\mathbf{w}}\frac{1}{\mathtt{m}}||\mathbf{X}\mathbf{w} - \mathbf{y}||_2^2 = 0$$

$$\Rightarrow \nabla_{\mathbf{w}}(\mathbf{X}\mathbf{w} - \mathbf{y})^\intercal(\mathbf{X}\mathbf{w} - \mathbf{y}) = 0$$

$$\Rightarrow \nabla_{\mathbf{w}}(\mathbf{w}^\intercal\mathbf{X}^\intercal\mathbf{X}\mathbf{w} - 2\mathbf{w}^\intercal\mathbf{X}^\intercal\mathbf{y} + \mathbf{y}^\intercal\mathbf{y}) = 0$$

Normal Equation (2/2)

- To minimize $J(\mathbf{w})$, we can simply solve for where its gradient is 0: $\nabla_{\mathbf{w}} J(\mathbf{w}) = 0$

$$J(\mathbf{w}) = \frac{1}{m}||\hat{\mathbf{y}} - \mathbf{y}||_2^2, \nabla_{\mathbf{w}} J(\mathbf{w}) = 0$$

$$\Rightarrow \nabla_{\mathbf{w}} \frac{1}{m}||\hat{\mathbf{y}} - \mathbf{y}||_2^2 = 0$$

$$\Rightarrow \nabla_{\mathbf{w}} \frac{1}{m}||\mathbf{X}\mathbf{w} - \mathbf{y}||_2^2 = 0$$

$$\Rightarrow \nabla_{\mathbf{w}} (\mathbf{X}\mathbf{w} - \mathbf{y})^\intercal (\mathbf{X}\mathbf{w} - \mathbf{y}) = 0$$

$$\Rightarrow \nabla_{\mathbf{w}} (\mathbf{w}^\intercal \mathbf{X}^\intercal \mathbf{X}\mathbf{w} - 2\mathbf{w}^\intercal \mathbf{X}^\intercal \mathbf{y} + \mathbf{y}^\intercal \mathbf{y}) = 0$$

$$\Rightarrow 2\mathbf{X}^\intercal \mathbf{X}\mathbf{w} - 2\mathbf{X}^\intercal \mathbf{y} = 0$$

Normal Equation (2/2)

- To minimize $J(\mathbf{w})$, we can simply solve for where its gradient is 0: $\nabla_{\mathbf{w}} J(\mathbf{w}) = 0$

$$J(\mathbf{w}) = \frac{1}{m}||\hat{\mathbf{y}} - \mathbf{y}||_2^2, \nabla_{\mathbf{w}} J(\mathbf{w}) = 0$$

$$\Rightarrow \nabla_{\mathbf{w}} \frac{1}{m}||\hat{\mathbf{y}} - \mathbf{y}||_2^2 = 0$$

$$\Rightarrow \nabla_{\mathbf{w}} \frac{1}{m}||\mathbf{X}\mathbf{w} - \mathbf{y}||_2^2 = 0$$

$$\Rightarrow \nabla_{\mathbf{w}} (\mathbf{X}\mathbf{w} - \mathbf{y})^\intercal (\mathbf{X}\mathbf{w} - \mathbf{y}) = 0$$

$$\Rightarrow \nabla_{\mathbf{w}} (\mathbf{w}^\intercal \mathbf{X}^\intercal \mathbf{X}\mathbf{w} - 2\mathbf{w}^\intercal \mathbf{X}^\intercal \mathbf{y} + \mathbf{y}^\intercal \mathbf{y}) = 0$$

$$\Rightarrow 2\mathbf{X}^\intercal \mathbf{X}\mathbf{w} - 2\mathbf{X}^\intercal \mathbf{y} = 0$$

$$\Rightarrow \mathbf{w} = (\mathbf{X}^\intercal \mathbf{X})^{-1} \mathbf{X}^\intercal \mathbf{y}$$

Normal Equation - Example (1/4)

Living area	No. of bedrooms	Price
2104	3	400
1600	3	330
2400	3	369
1416	2	232
3000	4	540

- Predict the value of $\hat{y}$, when $x_1 = 4000$ and $x_2 = 4$.

Normal Equation - Example (1/4)

Living area	No. of bedrooms	Price
2104	3	400
1600	3	330
2400	3	369
1416	2	232
3000	4	540

- Predict the value of $\hat{y}$, when $x_1 = 4000$ and $x_2 = 4$.
- We should find w_0, w_1, and w_2 in $\hat{y} = w_0 + w_1 x_1 + w_2 x_2$.
- $\mathbf{w} = (\mathbf{X}^\intercal \mathbf{X})^{-1} \mathbf{X}^\intercal \mathbf{y}$.

Normal Equation - Example (2/4)

Living area	No. of bedrooms	Price
2104	3	400
1600	3	330
2400	3	369
1416	2	232
3000	4	540

$$\mathbf{X} = \begin{bmatrix} 1 & 2104 & 3 \\ 1 & 1600 & 3 \\ 1 & 2400 & 3 \\ 1 & 1416 & 2 \\ 1 & 3000 & 4 \end{bmatrix} \qquad \mathbf{y} = \begin{bmatrix} 400 \\ 330 \\ 369 \\ 232 \\ 540 \end{bmatrix}$$

Normal Equation - Example (3/4)

$$\mathbf{w} = (\mathbf{X}^\intercal \mathbf{X})^{-1} \mathbf{X}^\intercal \mathrm{y}$$

Normal Equation - Example (3/4)

$$\mathbf{w} = (\mathbf{X}^\intercal \mathbf{X})^{-1} \mathbf{X}^\intercal \mathrm{y}$$

$$\mathbf{X}^\intercal \mathbf{X} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 2104 & 1600 & 2400 & 1416 & 3000 \\ 3 & 3 & 3 & 2 & 4 \end{bmatrix} \begin{bmatrix} 1 & 2104 & 3 \\ 1 & 1600 & 3 \\ 1 & 2400 & 3 \\ 1 & 1416 & 2 \\ 1 & 3000 & 4 \end{bmatrix} = \begin{bmatrix} 5 & 10520 & 15 \\ 10520 & 23751872 & 33144 \\ 15 & 33144 & 47 \end{bmatrix}$$

Normal Equation - Example (3/4)

$$\mathbf{w} = (\mathbf{X}^\intercal \mathbf{X})^{-1} \mathbf{X}^\intercal \mathrm{y}$$

$$\mathbf{X}^\intercal \mathbf{X} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 2104 & 1600 & 2400 & 1416 & 3000 \\ 3 & 3 & 3 & 2 & 4 \end{bmatrix} \begin{bmatrix} 1 & 2104 & 3 \\ 1 & 1600 & 3 \\ 1 & 2400 & 3 \\ 1 & 1416 & 2 \\ 1 & 3000 & 4 \end{bmatrix} = \begin{bmatrix} 5 & 10520 & 15 \\ 10520 & 23751872 & 33144 \\ 15 & 33144 & 47 \end{bmatrix}$$

$$(\mathbf{X}^\intercal \mathbf{X})^{-1} = \begin{bmatrix} 4.90366455e+00 & 7.48766737e-04 & -2.09302326e+00 \\ 7.48766737e-04 & 2.75281889e-06 & -2.18023256e-03 \\ -2.09302326e+00 & -2.18023256e-03 & 2.22674419e+00 \end{bmatrix}$$

Normal Equation - Example (3/4)

$$\mathbf{w} = (\mathbf{X}^\intercal\mathbf{X})^{-1}\mathbf{X}^\intercal \mathrm{y}$$

$$\mathbf{X}^\intercal\mathbf{X} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 2104 & 1600 & 2400 & 1416 & 3000 \\ 3 & 3 & 3 & 2 & 4 \end{bmatrix} \begin{bmatrix} 1 & 2104 & 3 \\ 1 & 1600 & 3 \\ 1 & 2400 & 3 \\ 1 & 1416 & 2 \\ 1 & 3000 & 4 \end{bmatrix} = \begin{bmatrix} 5 & 10520 & 15 \\ 10520 & 23751872 & 33144 \\ 15 & 33144 & 47 \end{bmatrix}$$

$$(\mathbf{X}^\intercal\mathbf{X})^{-1} = \begin{bmatrix} 4.90366455e+00 & 7.48766737e-04 & -2.09302326e+00 \\ 7.48766737e-04 & 2.75281889e-06 & -2.18023256e-03 \\ -2.09302326e+00 & -2.18023256e-03 & 2.22674419e+00 \end{bmatrix}$$

$$\mathbf{X}^\intercal\mathbf{y} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 2104 & 1600 & 2400 & 1416 & 3000 \\ 3 & 3 & 3 & 2 & 4 \end{bmatrix} \begin{bmatrix} 400 \\ 330 \\ 369 \\ 232 \\ 540 \end{bmatrix} = \begin{bmatrix} 1871 \\ 4203712 \\ 5921 \end{bmatrix}$$

Normal Equation - Example (4/4)

$$\mathbf{w} = (\mathbf{X}^\intercal\mathbf{X})^{-1}\mathbf{X}^\intercal\mathrm{y} = \begin{bmatrix} 4.90366455\mathrm{e}+00 & 7.48766737\mathrm{e}-04 & -2.09302326\mathrm{e}+00 \\ 7.48766737\mathrm{e}-04 & 2.75281889\mathrm{e}-06 & -2.18023256\mathrm{e}-03 \\ -2.09302326\mathrm{e}+00 & -2.18023256\mathrm{e}-03 & 2.22674419\mathrm{e}+00 \end{bmatrix} \begin{bmatrix} 1871 \\ 4203712 \\ 5921 \end{bmatrix}$$

$$= \begin{bmatrix} -7.04346018e+01 \\ 6.38433756e-02 \\ 1.03436047e+02 \end{bmatrix}$$

Normal Equation - Example (4/4)

$$\mathbf{w} = (\mathbf{X}^\intercal\mathbf{X})^{-1}\mathbf{X}^\intercal\mathrm{y} = \begin{bmatrix} 4.90366455e+00 & 7.48766737e-04 & -2.09302326e+00 \\ 7.48766737e-04 & 2.75281889e-06 & -2.18023256e-03 \\ -2.09302326e+00 & -2.18023256e-03 & 2.22674419e+00 \end{bmatrix} \begin{bmatrix} 1871 \\ 4203712 \\ 5921 \end{bmatrix}$$

$$= \begin{bmatrix} -7.04346018e+01 \\ 6.38433756e-02 \\ 1.03436047e+02 \end{bmatrix}$$

- Predict the value of y, when $\mathrm{x}_1 = 4000$ and $\mathrm{x}_2 = 4$.

$$\hat{\mathrm{y}} = -7.04346018e+01 + 6.38433756e-02 \times 4000 + 1.03436047e+02 \times 4 \approx 599$$

Normal Equation in TensorFlow (1/2)

```
import numpy as np
import tensorflow as tf
from sklearn.datasets import fetch_california_housing
```

Normal Equation in TensorFlow (1/2)

```
import numpy as np
import tensorflow as tf
from sklearn.datasets import fetch_california_housing
```

```
housing = fetch_california_housing()

X_train = housing.data
y_train = housing.target.reshape(-1, 1) # reshaping is done to convert y from vector to matrix
```

Normal Equation in TensorFlow (1/2)

```
import numpy as np
import tensorflow as tf
from sklearn.datasets import fetch_california_housing
```

```
housing = fetch_california_housing()

X_train = housing.data
y_train = housing.target.reshape(-1, 1) # reshaping is done to convert y from vector to matrix
```

```
# add the bias input feature i.e. a column of 1's

m = len(y_train)
X_train = np.c_[np.ones(m), X_train]
```

Normal Equation in TensorFlow (2/2)

```
# create TensorFlow Constants to store data

X = tf.constant(X_train, tf.float32, name="X")
y = tf.constant(y_train, tf.float32, name="y")
```

Normal Equation in TensorFlow (2/2)

```
# create TensorFlow Constants to store data

X = tf.constant(X_train, tf.float32, name="X")
y = tf.constant(y_train, tf.float32, name="y")
```

```
# use Normal Equation, i.e., w = (X^T.X)^-1.X.y

X_T = tf.transpose(X)
temp = tf.matrix_inverse(tf.matmul(X_T, X))
w = tf.matmul(tf.matmul(temp, X_T), y)
```

Normal Equation in TensorFlow (2/2)

```
# create TensorFlow Constants to store data

X = tf.constant(X_train, tf.float32, name="X")
y = tf.constant(y_train, tf.float32, name="y")
```

```
# use Normal Equation, i.e., w = (X^T.X)^-1.X.y

X_T = tf.transpose(X)
temp = tf.matrix_inverse(tf.matmul(X_T, X))
w = tf.matmul(tf.matmul(temp, X_T), y)
```

```
# create TensorFlow Session

with tf.Session() as sess:
    weights = w.eval()
print(weights)
```

Normal Equation - Computational Complexity

- The computational complexity of inverting $\mathbf{X}^\intercal\mathbf{X}$ is $O(n^3)$.
  - For an $m \times n$ matrix (where n is the number of features).

Normal Equation - Computational Complexity

- The computational complexity of inverting $\mathbf{X}^\intercal\mathbf{X}$ is $\mathtt{O}(\mathtt{n}^3)$.
  - For an $\mathtt{m} \times \mathtt{n}$ matrix (where $\mathtt{n}$ is the number of features).
- But, this equation is linear with regards to the number of instances in the training set (it is $\mathtt{O}(\mathtt{m})$).
  - It handles large training sets efficiently, provided they can fit in memory.

Gradient Descent


GRADIENTS,
GRADIENTS EVERYWHERE
quickmeme.com

Gradient Descent (1/2)

- Gradient descent is a generic optimization algorithm capable of finding optimal solutions to a wide range of problems.

Gradient Descent (1/2)

- Gradient descent is a generic optimization algorithm capable of finding optimal solutions to a wide range of problems.

- To tweak parameters $\mathbf{w}$ iteratively in order to minimize a cost function $\mathtt{J}(\mathbf{w})$.

Gradient Descent (2/2)

- Suppose you are lost in the mountains in a dense fog.

Gradient Descent (2/2)

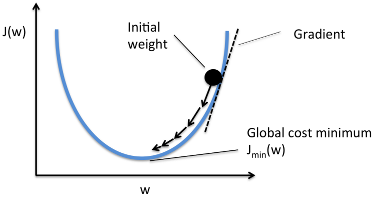
- Suppose you are lost in the mountains in a dense fog.
- You can only feel the slope of the ground below your feet.

Gradient Descent (2/2)

- Suppose you are lost in the mountains in a dense fog.
- You can only feel the slope of the ground below your feet.
- A strategy to get to the bottom of the valley is to go downhill in the direction of the steepest slope.

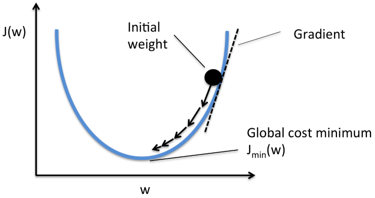
Gradient Descent - Iterative Optimization Algorithm

- Choose a starting point, e.g., filling **w** with random values.



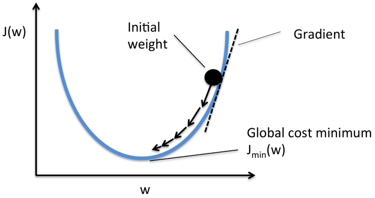
Gradient Descent - Iterative Optimization Algorithm

- Choose a starting point, e.g., filling **w** with random values.
- If the stopping criterion is true return the current solution, otherwise continue.



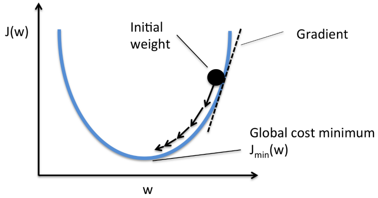
Gradient Descent - Iterative Optimization Algorithm

- Choose a starting point, e.g., filling **w** with random values.
- If the stopping criterion is true return the current solution, otherwise continue.
- Find a descent direction, a direction in which the function value decreases near the current point.



Gradient Descent - Iterative Optimization Algorithm

- Choose a starting point, e.g., filling **w** with random values.
- If the stopping criterion is true return the current solution, otherwise continue.
- Find a descent direction, a direction in which the function value decreases near the current point.
- Determine the step size, the length of a step in the given direction.



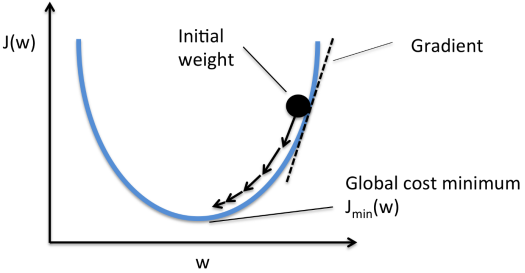
Gradient Descent - Key Points

- Stopping criterion
- Descent direction
- Step size (learning rate)

Gradient Descent - Stopping Criterion

- The cost function minimum property: the gradient has to be zero.

$$\nabla_{\mathbf{w}} J(\mathbf{w}) = 0$$



Gradient Descent - Descent Direction (1/2)

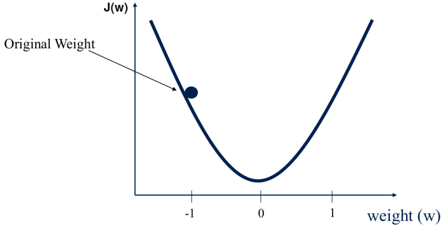
- Direction in which the function value decreases near the current point.
- Find the direction of descent (slope).

Gradient Descent - Descent Direction (1/2)

- Direction in which the function value decreases near the current point.
- Find the direction of descent (slope).
- Example:

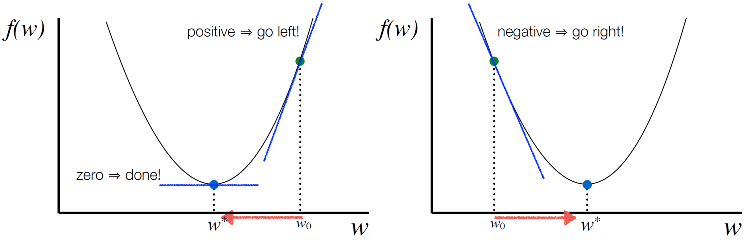
$$J(w) = w^2$$

$$\frac{\partial J(w)}{\partial w} = 2w = -2 \text{ at } w = -1$$



Gradient Descent - Descent Direction (2/2)

- Follow the opposite direction of the slope.

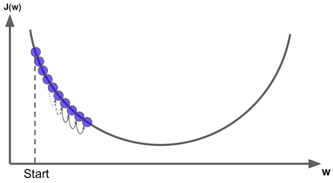


Gradient Descent - Learning Rate

- Learning rate: the length of steps.

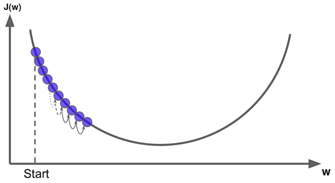
Gradient Descent - Learning Rate

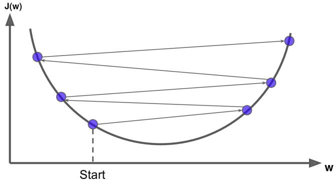
- Learning rate: the length of steps.
- If it is too small: many iterations to converge.



Gradient Descent - Learning Rate

- Learning rate: the length of steps.
- If it is too small: many iterations to converge.
- If it is too high: the algorithm might diverge.





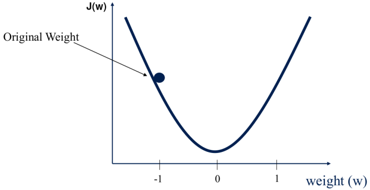
Gradient Descent - How to Learn Model Parameters **w**?

- Goal: find $\mathbf{w}$ that minimizes $J(\mathbf{w}) = \sum_{i=1}^{m}(\mathbf{w}^\intercal \mathbf{x}^{(i)} - y^{(i)})^2$.

Gradient Descent - How to Learn Model Parameters **w**?

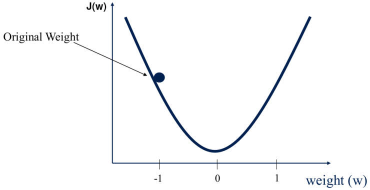
- Goal: find $\mathbf{w}$ that minimizes $J(\mathbf{w}) = \sum_{i=1}^{m}(\mathbf{w}^\intercal \mathbf{x}^{(i)} - y^{(i)})^2$.

- Start at a random point, and repeat the following steps, until the stopping criterion is satisfied:



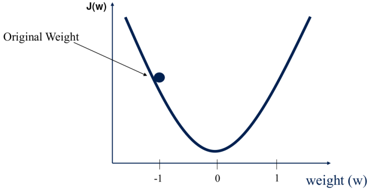
Gradient Descent - How to Learn Model Parameters **w**?

- Goal: find **w** that minimizes $J(\mathbf{w}) = \sum_{i=1}^{m}(\mathbf{w}^\intercal \mathbf{x}^{(i)} - y^{(i)})^2$.
- Start at a random point, and repeat the following steps, until the stopping criterion is satisfied:
  1. Determine a descent direction $\frac{\partial J(\mathbf{w})}{\partial w}$



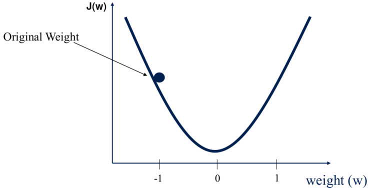
Gradient Descent - How to Learn Model Parameters $\mathbf{w}$?

- Goal: find $\mathbf{w}$ that minimizes $J(\mathbf{w}) = \sum_{i=1}^{m}(\mathbf{w}^\intercal \mathbf{x}^{(i)} - y^{(i)})^2$.
- Start at a random point, and repeat the following steps, until the stopping criterion is satisfied:
  1. Determine a descent direction $\frac{\partial J(\mathbf{w})}{\partial w}$
  2. Choose a step size η



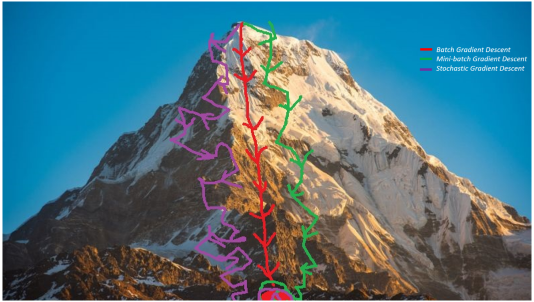
Gradient Descent - How to Learn Model Parameters $\mathbf{w}$?

- Goal: find $\mathbf{w}$ that minimizes $J(\mathbf{w}) = \sum_{i=1}^{m}(\mathbf{w}^\top \mathbf{x}^{(i)} - y^{(i)})^2$.
- Start at a random point, and repeat the following steps, until the stopping criterion is satisfied:
  1. Determine a descent direction $\frac{\partial J(\mathbf{w})}{\partial w}$
  2. Choose a step size η
  3. Update the parameters: $w^{(next)} = w - \eta \frac{\partial J(\mathbf{w})}{\partial w}$ (should be done for all parameters simultanously)



Gradient Descent - Different Algorithms

- Batch gradient descent
- Stochastic gradient descent
- Mini-batch gradient descent



[https://towardsdatascience.com/gradient-descent-algorithm-and-its-variants-10f652806a3]

Batch Gradient Descent

Batch Gradient Descent (1/2)

- Repeat the following steps, until the stopping criterion is satisfied:

1. Determine a descent direction $\frac{\partial J(\mathbf{w})}{\partial \mathbf{w}}$ for all parameters $\mathbf{w}$.

$$J(\mathbf{w}) = \sum_{i=1}^{m}(\mathbf{w}^\intercal \mathbf{x}^{(i)} - y^{(i)})^2$$

Batch Gradient Descent (1/2)

- Repeat the following steps, until the stopping criterion is satisfied:
  1. Determine a descent direction $\frac{\partial J(\mathbf{w})}{\partial w}$ for all parameters $\mathbf{w}$.

$$J(\mathbf{w}) = \sum_{i=1}^{m}(\mathbf{w}^\intercal \mathbf{x}^{(i)} - y^{(i)})^2$$

$$\frac{\partial J(\mathbf{w})}{\partial w_j} = \frac{2}{m}\sum_{i=1}^{m}(\mathbf{w}^\intercal \mathbf{x}^{(i)} - y^{(i)})x_j^{(i)}2$$

Batch Gradient Descent (1/2)

- Repeat the following steps, until the stopping criterion is satisfied:

1. Determine a descent direction $\frac{\partial J(\mathbf{w})}{\partial w}$ for all parameters $\mathbf{w}$.

$$J(\mathbf{w}) = \sum_{i=1}^{m}(\mathbf{w}^\intercal \mathbf{x}^{(i)} - y^{(i)})^2$$

$$\frac{\partial J(\mathbf{w})}{\partial w_j} = \frac{2}{m}\sum_{i=1}^{m}(\mathbf{w}^\intercal \mathbf{x}^{(i)} - y^{(i)})x_j^{(i)}2$$

2. Choose a step size η

Batch Gradient Descent (1/2)

- Repeat the following steps, until the stopping criterion is satisfied:

1. Determine a descent direction $\frac{\partial J(\mathbf{w})}{\partial w}$ for all parameters $\mathbf{w}$.

$$J(\mathbf{w}) = \sum_{i=1}^{m} (\mathbf{w}^\intercal \mathbf{x}^{(i)} - y^{(i)})^2$$

$$\frac{\partial J(\mathbf{w})}{\partial w_j} = \frac{2}{m} \sum_{i=1}^{m} (\mathbf{w}^\intercal \mathbf{x}^{(i)} - y^{(i)}) x_j^{(i)} 2$$

2. Choose a step size η
3. Update the parameters: $w_j^{(next)} = w_j - \eta \frac{\partial J(\mathbf{w})}{\partial w_j}$

Batch Gradient Descent (2/2)

- Batch Gradient Descent: at each step the calculation is over the full training set $\mathbf{X}$.

$$J(\mathbf{w}) = \sum_{i=1}^{m}(\mathbf{w}^\intercal \mathbf{x}^{(i)} - y^{(i)})^2$$

Batch Gradient Descent (2/2)

- Batch Gradient Descent: at each step the calculation is over the full training set $\mathbf{X}$.

$$J(\mathbf{w}) = \sum_{i=1}^{m} (\mathbf{w}^\intercal \mathbf{x}^{(i)} - y^{(i)})^2$$

- As a result it is slow on very large training sets, i.e., large m.

Batch Gradient Descent (2/2)

- Batch Gradient Descent: at each step the calculation is over the full training set $\mathbf{X}$.

$$J(\mathbf{w}) = \sum_{i=1}^{m} (\mathbf{w}^\intercal \mathbf{x}^{(i)} - y^{(i)})^2$$

- As a result it is slow on very large training sets, i.e., large m.
- But, it scales well with the number of features n.

Batch Gradient Descent - Example (1/5)

Living area	No. of bedrooms	Price
2104	3	400
1600	3	330
2400	3	369
1416	2	232
3000	4	540

$$\hat{y} = w_0 + w_1 x_1 + w_2 x_2$$

$$\mathbf{X} = \left[\begin{array}{c|cc} 1 & 2104 & 3 \\ 1 & 1600 & 3 \\ 1 & 2400 & 3 \\ 1 & 1416 & 2 \\ 1 & 3000 & 4 \end{array}\right] \qquad \mathbf{y} = \begin{bmatrix} 400 \\ 330 \\ 369 \\ 232 \\ 540 \end{bmatrix}$$

Batch Gradient Descent - Example (2/5)

$$\mathbf{X} = \left[\begin{array}{c|cc} 1 & 2104 & 3 \\ 1 & 1600 & 3 \\ 1 & 2400 & 3 \\ 1 & 1416 & 2 \\ 1 & 3000 & 4 \end{array}\right] \qquad \mathbf{y} = \begin{bmatrix} 400 \\ 330 \\ 369 \\ 232 \\ 540 \end{bmatrix}$$

$$\frac{\partial J(\mathbf{w})}{\partial w_0} = \frac{2}{m}\sum_{i=1}^{m}(\mathbf{w}^\intercal \mathbf{x}^{(i)} - y^{(i)})x_0^{(i)}$$

$$= \frac{2}{5}[(w_0 + 2104w_1 + 3w_2 - 400) + (w_0 + 1600w_1 + 3w_2 - 330) +$$

$$(w_0 + 2400w_1 + 3w_2 - 369) + (w_0 + 1416w_1 + 2w_2 - 232) + (w_0 + 3000w_1 + 4w_2 - 540)]$$

Batch Gradient Descent - Example (3/5)

$$\mathbf{X} = \left[\begin{array}{c|cc} 1 & 2104 & 3 \\ 1 & 1600 & 3 \\ 1 & 2400 & 3 \\ 1 & 1416 & 2 \\ 1 & 3000 & 4 \end{array}\right] \qquad \mathbf{y} = \begin{bmatrix} 400 \\ 330 \\ 369 \\ 232 \\ 540 \end{bmatrix}$$

$$\frac{\partial J(\mathbf{w})}{\partial w_1} = \frac{2}{m}\sum_{i=1}^{m}(\mathbf{w}^\intercal \mathbf{x}^{(i)} - y^{(i)})x_1^{(i)}$$

$$= \frac{2}{5}[2104(w_0 + 2104w_1 + 3w_2 - 400) + 1600(w_0 + 1600w_1 + 3w_2 - 330)+$$

$$2400(w_0 + 2400w_1 + 3w_2 - 369) + 1416(w_0 + 1416w_1 + 2w_2 - 232) + 3000(w_0 + 3000w_1 + 4w_2 - 540)]$$

Batch Gradient Descent - Example (4/5)

$$\mathbf{X} = \begin{bmatrix} 1 & 2104 & 3 \\ 1 & 1600 & 3 \\ 1 & 2400 & 3 \\ 1 & 1416 & 2 \\ 1 & 3000 & 4 \end{bmatrix} \qquad \mathbf{y} = \begin{bmatrix} 400 \\ 330 \\ 369 \\ 232 \\ 540 \end{bmatrix}$$

$$\frac{\partial J(\mathbf{w})}{\partial w_2} = \frac{2}{m}\sum_{i=1}^{m}(\mathbf{w}^\intercal \mathbf{x}^{(i)} - y^{(i)})x_2^{(i)}$$

$$= \frac{2}{5}[3(w_0 + 2104w_1 + 3w_2 - 400) + 3(w_0 + 1600w_1 + 3w_2 - 330) +$$

$$3(w_0 + 2400w_1 + 3w_2 - 369) + 2(w_0 + 1416w_1 + 2w_2 - 232) + 4(w_0 + 3000w_1 + 4w_2 - 540)]$$

Batch Gradient Descent - Example (5/5)

$$\mathtt{w}_0^{(\mathtt{next})} = \mathtt{w}_0 - \eta \frac{\partial \mathtt{J}(\mathbf{w})}{\partial \mathtt{w}_0}$$

$$\mathtt{w}_1^{(\mathtt{next})} = \mathtt{w}_1 - \eta \frac{\partial \mathtt{J}(\mathbf{w})}{\partial \mathtt{w}_1}$$

$$\mathtt{w}_2^{(\mathtt{next})} = \mathtt{w}_2 - \eta \frac{\partial \mathtt{J}(\mathbf{w})}{\partial \mathtt{w}_2}$$

Stochastic Gradient Descent

Stochastic Gradient Descent

- Batch gradient descent problem: it's slow, because it uses the whole training set to compute the gradients at every step.

Stochastic Gradient Descent

- Batch gradient descent problem: it's slow, because it uses the whole training set to compute the gradients at every step.

- Stochastic gradient descent computes the gradients based on only a single instance.
  - It picks a random instance in the training set at every step.

Stochastic Gradient Descent

- Batch gradient descent problem: it's slow, because it uses the whole training set to compute the gradients at every step.

- Stochastic gradient descent computes the gradients based on only a single instance.
  - It picks a random instance in the training set at every step.

- The algorithm is much faster, but less regular than batch gradient descent.

Stochastic Gradient Descent - Example (1/3)

Living area	No. of bedrooms	Price
2104	3	400
1600	3	330
2400	3	369
1416	2	232
3000	4	540

$$\hat{y} = w_0 + w_1 x_1 + w_2 x_2$$

$$\mathbf{X} = \left[\begin{array}{c|cc} 1 & 2104 & 3 \\ 1 & 1600 & 3 \\ 1 & 2400 & 3 \\ 1 & 1416 & 2 \\ 1 & 3000 & 4 \end{array}\right] \qquad \mathbf{y} = \begin{bmatrix} 400 \\ 330 \\ 369 \\ 232 \\ 540 \end{bmatrix}$$

Stochastic Gradient Descent - Example (2/3)

$$\mathbf{X} = \left[\begin{array}{c|cc} 1 & 2104 & 3 \\ 1 & 1600 & 3 \\ 1 & 2400 & 3 \\ 1 & 1416 & 2 \\ 1 & 3000 & 4 \end{array}\right] \qquad \mathbf{y} = \begin{bmatrix} 400 \\ 330 \\ 369 \\ 232 \\ 540 \end{bmatrix}$$

$$\frac{\partial J(\mathbf{w})}{\partial w_0} = \frac{2}{m}(\mathbf{w}^\intercal \mathbf{x}^{(i)} - y^{(i)})x_0^{(i)} = \frac{2}{5}[(w_0 + 1600w_1 + 3w_2 - 330)]$$

Stochastic Gradient Descent - Example (2/3)

$$\mathbf{X} = \left[\begin{array}{c|cc} 1 & 2104 & 3 \\ 1 & 1600 & 3 \\ 1 & 2400 & 3 \\ 1 & 1416 & 2 \\ 1 & 3000 & 4 \end{array}\right] \qquad \mathbf{y} = \begin{bmatrix} 400 \\ 330 \\ 369 \\ 232 \\ 540 \end{bmatrix}$$

$$\frac{\partial J(\mathbf{w})}{\partial w_0} = \frac{2}{m}(\mathbf{w}^\mathsf{T}\mathbf{x}^{(i)} - y^{(i)})x_0^{(i)} = \frac{2}{5}[(w_0 + 1600w_1 + 3w_2 - 330)]$$

$$\frac{\partial J(\mathbf{w})}{\partial w_1} = \frac{2}{m}(\mathbf{w}^\mathsf{T}\mathbf{x}^{(i)} - y^{(i)})x_1^{(i)} = \frac{2}{5}[1416(w_0 + 1416w_1 + 2w_2 - 232)]$$

Stochastic Gradient Descent - Example (2/3)

$$\mathbf{X} = \begin{bmatrix} 1 & 2104 & 3 \\ 1 & 1600 & 3 \\ 1 & 2400 & 3 \\ 1 & 1416 & 2 \\ 1 & 3000 & 4 \end{bmatrix} \qquad \mathbf{y} = \begin{bmatrix} 400 \\ 330 \\ 369 \\ 232 \\ 540 \end{bmatrix}$$

$$\frac{\partial J(\mathbf{w})}{\partial w_0} = \frac{2}{m}(\mathbf{w}^\intercal \mathbf{x}^{(i)} - y^{(i)})x_0^{(i)} = \frac{2}{5}[(w_0 + 1600w_1 + 3w_2 - 330)]$$

$$\frac{\partial J(\mathbf{w})}{\partial w_1} = \frac{2}{m}(\mathbf{w}^\intercal \mathbf{x}^{(i)} - y^{(i)})x_1^{(i)} = \frac{2}{5}[1416(w_0 + 1416w_1 + 2w_2 - 232)]$$

$$\frac{\partial J(\mathbf{w})}{\partial w_2} = \frac{2}{m}(\mathbf{w}^\intercal \mathbf{x}^{(i)} - y^{(i)})x_2^{(i)} = \frac{2}{5}[3(w_0 + 2104w_1 + 3w_2 - 400)]$$

Stochastic Gradient Descent - Example (3/3)

$$
\begin{aligned}
w_0^{(next)} &= w_0 - \eta \frac{\partial J(\mathbf{w})}{\partial w_0} \\
w_1^{(next)} &= w_1 - \eta \frac{\partial J(\mathbf{w})}{\partial w_1} \\
w_2^{(next)} &= w_2 - \eta \frac{\partial J(\mathbf{w})}{\partial w_2}
\end{aligned}
$$

Mini-Batch Gradient Descent

Mini-Batch Gradient Descent

- Batch gradient descent: at each step, it computes the gradients based on the full training set.

Mini-Batch Gradient Descent

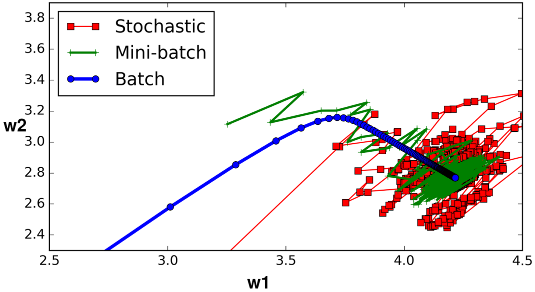
- Batch gradient descent: at each step, it computes the gradients based on the full training set.
- Stochastic gradient descent: at each step, it computes the gradients based on just one instance.

Mini-Batch Gradient Descent

- Batch gradient descent: at each step, it computes the gradients based on the full training set.
- Stochastic gradient descent: at each step, it computes the gradients based on just one instance.
- Mini-batch gradient descent: at each step, it computes the gradients based on small random sets of instances called mini-batches.

Comparison of Algorithms for Linear Regression

Algorithm	Large m	Large n
Normal Equation	Fast	Slow
Batch GD	Slow	Fast
Stochastic GD	Fast	Fast
Mini-batch GD	Fast	Fast



Gradient Descent in TensorFlow - First Implementation

```
x_train = [1, 2, 3]
y_train = [1, 2, 3]

X = tf.placeholder(tf.float32)
y_true = tf.placeholder(tf.float32)

w = tf.Variable(5.)
b = tf.Variable(5.)
```

Gradient Descent in TensorFlow - First Implementation

```
x_train = [1, 2, 3]
y_train = [1, 2, 3]

X = tf.placeholder(tf.float32)
y_true = tf.placeholder(tf.float32)

w = tf.Variable(5.)
b = tf.Variable(5.)
```

```
y_hat = tf.matmul(w, tf.transpose(x)) + b
cost = tf.reduce_mean(tf.square(y_hat - y_true))
```

Gradient Descent in TensorFlow - First Implementation

```
x_train = [1, 2, 3]
y_train = [1, 2, 3]

X = tf.placeholder(tf.float32)
y_true = tf.placeholder(tf.float32)

w = tf.Variable(5.)
b = tf.Variable(5.)
```

```
y_hat = tf.matmul(w, tf.transpose(x)) + b
cost = tf.reduce_mean(tf.square(y_hat - y_true))
```

```
learning_rate = 0.1
w_gradient = tf.reduce_mean((y_hat - y_true) * X) * 2
w_descent = w - learning_rate * w_gradient
w_update = tf.assign(w, w_descent)
```

Gradient Descent in TensorFlow - First Implementation

```
x_train = [1, 2, 3]
y_train = [1, 2, 3]

X = tf.placeholder(tf.float32)
y_true = tf.placeholder(tf.float32)

w = tf.Variable(5.)
b = tf.Variable(5.)
```

```
y_hat = tf.matmul(w, tf.transpose(x)) + b
cost = tf.reduce_mean(tf.square(y_hat - y_true))
```

```
learning_rate = 0.1
w_gradient = tf.reduce_mean((y_hat - y_true) * X) * 2
w_descent = w - learning_rate * w_gradient
w_update = tf.assign(w, w_descent)
```

```
b_gradient = tf.reduce_mean(y_hat - y_true) * 2
b_descent = b - learning_rate * b_gradient
b_update = tf.assign(b, b_descent)
```

Gradient Descent in TensorFlow - Second Implementation

```
x_train = [1, 2, 3]
y_train = [1, 2, 3]

X = tf.placeholder(tf.float32)
y_true = tf.placeholder(tf.float32)

w = tf.Variable(5.)
b = tf.Variable(5.)
```

Gradient Descent in TensorFlow - Second Implementation

```
x_train = [1, 2, 3]
y_train = [1, 2, 3]

X = tf.placeholder(tf.float32)
y_true = tf.placeholder(tf.float32)

w = tf.Variable(5.)
b = tf.Variable(5.)
```

```
y_hat = tf.matmul(w, tf.transpose(x)) + b
cost = tf.reduce_mean(tf.square(y_hat - Y))
```

Gradient Descent in TensorFlow - Second Implementation

```
x_train = [1, 2, 3]
y_train = [1, 2, 3]

X = tf.placeholder(tf.float32)
y_true = tf.placeholder(tf.float32)

w = tf.Variable(5.)
b = tf.Variable(5.)
```

```
y_hat = tf.matmul(w, tf.transpose(x)) + b
cost = tf.reduce_mean(tf.square(y_hat - Y))
```

```
learning_rate = 0.1
optimizer = tf.train.GradientDescentOptimizer(learning_rate=learning_rate)
gvs = optimizer.compute_gradients(cost, [w, b])
apply_gradients = optimizer.apply_gradients(gvs)
```

Gradient Descent in TensorFlow - Third Implementation

```
x_train = [1, 2, 3]
y_train = [1, 2, 3]

X = tf.placeholder(tf.float32)
y_true = tf.placeholder(tf.float32)

w = tf.Variable(5.)
b = tf.Variable(5.)
```

Gradient Descent in TensorFlow - Third Implementation

```
x_train = [1, 2, 3]
y_train = [1, 2, 3]

X = tf.placeholder(tf.float32)
y_true = tf.placeholder(tf.float32)

w = tf.Variable(5.)
b = tf.Variable(5.)
```

```
y_hat = tf.matmul(w, tf.transpose(x)) + b
cost = tf.reduce_mean(tf.square(y_hat - y_true))
```

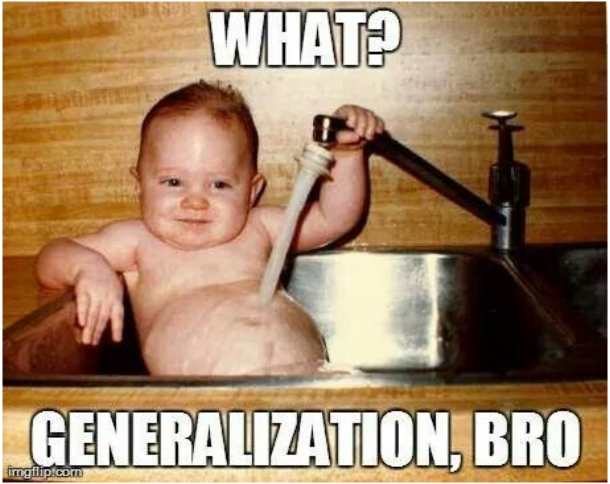
Gradient Descent in TensorFlow - Third Implementation

```
x_train = [1, 2, 3]
y_train = [1, 2, 3]

X = tf.placeholder(tf.float32)
y_true = tf.placeholder(tf.float32)

w = tf.Variable(5.)
b = tf.Variable(5.)
```

```
y_hat = tf.matmul(w, tf.transpose(x)) + b
cost = tf.reduce_mean(tf.square(y_hat - y_true))
```

```
learning_rate = 0.1
optimizer = tf.train.GradientDescentOptimizer(learning_rate=learning_rate)
op = optimizer.minimize(cost)
```

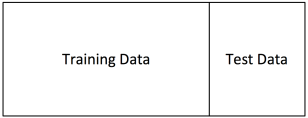
Generalization


WHAT?
GENERALIZATION, BRO
imgflip.com

Training Data and Test Data

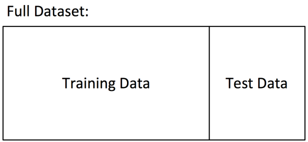
- Split data into a training set and a test set.

Full Dataset:



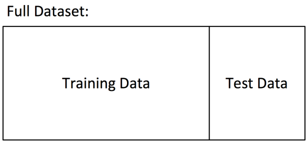
Training Data and Test Data

- Split data into a training set and a test set.
- Use training set when training a machine learning model.
  - Try to reduce this training error.



Training Data and Test Data

- Split data into a training set and a test set.
- Use training set when training a machine learning model.
  - Try to reduce this training error.
- Use test set to measure the accuracy of the model.
  - Test error is the error when you run the trained model on test data (new data).



Generalization

- Generalization: make a model that performs well on test data.

Generalization

- Generalization: make a model that performs well on test data.
  - Have a small test error.

Generalization

- Generalization: make a model that performs well on test data.
  - Have a small test error.

- Challenges
  1. Make the training error small.
  2. Make the gap between training and test error small.

More About The Test Error

- The test error is computed as the MSE of k test instances.

$$\mathrm{MSE}_{\mathrm{test}} = \frac{1}{k}\sum_{i}^{k}(\hat{y}^{(i)}_{\mathrm{test}} - y^{(i)}_{\mathrm{test}})^2 = \mathrm{E}[(\hat{y}_{\mathrm{test}} - y_{\mathrm{test}})^2]$$

More About The Test Error

- The test error is computed as the MSE of $\mathtt{k}$ test instances.

$$\mathtt{MSE}_{\mathtt{test}} = \frac{1}{\mathtt{k}} \sum_{\mathtt{i}}^{\mathtt{k}} (\hat{\mathtt{y}}_{\mathtt{test}}^{(\mathtt{i})} - \mathtt{y}_{\mathtt{test}}^{(\mathtt{i})})^2 = \mathtt{E}[(\hat{\mathtt{y}}_{\mathtt{test}} - \mathtt{y}_{\mathtt{test}})^2]$$

- A model's test error can be expressed as the sum of bias and variance.

$$\mathtt{E}[(\hat{\mathtt{y}}_{\mathtt{test}} - \mathtt{y}_{\mathtt{test}})^2] = \mathtt{Bias}[\hat{\mathtt{y}}_{\mathtt{test}}, \mathtt{y}_{\mathtt{test}}]^2 + \mathtt{Var}[\hat{\mathtt{y}}_{\mathtt{test}}] + \varepsilon^2$$

Bias and Underfitting

- Bias: the expected deviation from the true value of the function.

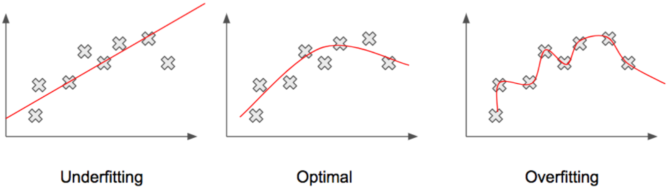
$$\mathtt{Bias}[\hat{\mathtt{y}}_{\mathtt{test}}, \mathtt{y}_{\mathtt{test}}] = \mathtt{E}[\hat{\mathtt{y}}_{\mathtt{test}}] - \mathtt{y}_{\mathtt{test}}$$

Bias and Underfitting

- Bias: the expected deviation from the true value of the function.

$$\texttt{Bias}[\hat{y}_{\texttt{test}}, y_{\texttt{test}}] = \texttt{E}[\hat{y}_{\texttt{test}}] - y_{\texttt{test}}$$

- A high-bias model is most likely to underfit the training data.
  - High error value on the training set.

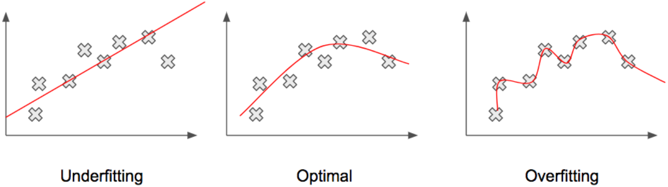


Bias and Underfitting

- Bias: the expected deviation from the true value of the function.

$$\texttt{Bias}[\hat{\mathtt{y}}_{\texttt{test}}, \mathtt{y}_{\texttt{test}}] = \mathtt{E}[\hat{\mathtt{y}}_{\texttt{test}}] - \mathtt{y}_{\texttt{test}}$$

- A high-bias model is most likely to underfit the training data.
  - High error value on the training set.
- Underfitting happens when the model is too simple to learn the underlying structure of the data.



Variance and Overfitting

- Variance: how much a model changes if you train it on a different training set.

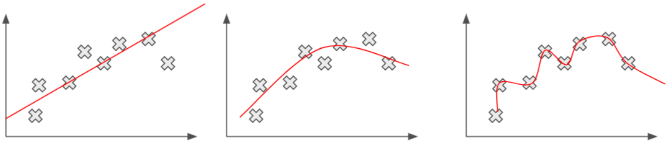
$$\mathtt{Var}[\hat{y}_{\mathtt{test}}] = \mathtt{E}[(\hat{y}_{\mathtt{test}} - \mathtt{E}[\hat{y}_{\mathtt{test}}])^2]$$

Variance and Overfitting

- Variance: how much a model changes if you train it on a different training set.

$$\text{Var}[\hat{y}_{\text{test}}] = \text{E}[(\hat{y}_{\text{test}} - \text{E}[\hat{y}_{\text{test}}])^2]$$

- A high-variance model is most likely to overfit the training data.
  - The gap between the training error and test error is too large.



Underfitting Optimal Overfitting

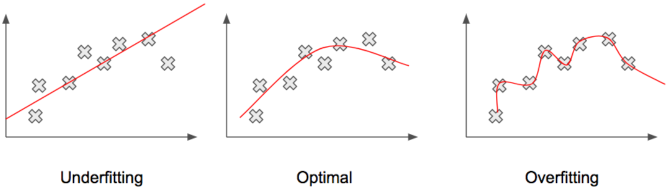
Variance and Overfitting

- Variance: how much a model changes if you train it on a different training set.

$$\mathtt{Var}[\hat{y}_{\mathtt{test}}] = \mathtt{E}[(\hat{y}_{\mathtt{test}} - \mathtt{E}[\hat{y}_{\mathtt{test}}])^2]$$

- A high-variance model is most likely to overfit the training data.
  - The gap between the training error and test error is too large.

- Overfitting happens when the model is too complex relative to the amount and noisiness of the training data.



The Bias/Variance Tradeoff (1/2)

- Assume a model with two parameters w_0 (intercept) and w_1 (slope): $\hat{y} = w_0 + w_1 x$

The Bias/Variance Tradeoff (1/2)

- Assume a model with two parameters w_0 (intercept) and w_1 (slope): $\hat{y} = w_0 + w_1 x$
- They give the learning algorithm two degrees of freedom.

The Bias/Variance Tradeoff (1/2)

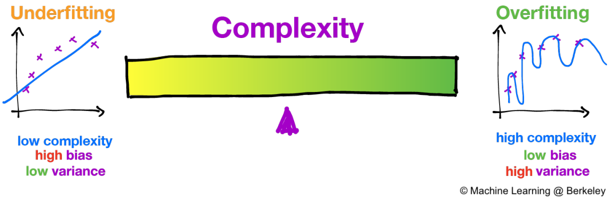
- Assume a model with two parameters w_0 (intercept) and w_1 (slope): $\hat{y} = w_0 + w_1 x$
- They give the learning algorithm two degrees of freedom.
- We tweak both the w_0 and w_1 to adapt the model to the training data.

The Bias/Variance Tradeoff (1/2)

- Assume a model with two parameters w_0 (intercept) and w_1 (slope): $\hat{y} = w_0 + w_1 x$
- They give the learning algorithm two degrees of freedom.
- We tweak both the w_0 and w_1 to adapt the model to the training data.
- If we forced $w_0 = 0$, the algorithm would have only one degree of freedom and would have a much harder time fitting the data properly.

The Bias/Variance Tradeoff (2/2)

- Increasing degrees of freedom will typically increase its variance and reduce its bias.
- Decreasing degrees of freedom increases its bias and reduces its variance.
- This is why it is called a tradeoff.



[https://ml.berkeley.edu/blog/2017/07/13/tutorial-4]

Regularization (1/2)

- One way to reduce the risk of overfitting is to have fewer degrees of freedom.
- Regularization is a technique to reduce the risk of overfitting.
- For a linear model, regularization is achieved by constraining the weights of the model.

$$\mathtt{J}(\mathbf{w}) = \mathtt{MSE}(\mathbf{w}) + \lambda \mathtt{R}(\mathbf{w})$$

Regularization (2/2)

- Lasso regression ($l1$): $\mathtt{R}(\mathbf{w}) = \lambda \sum_{\mathtt{i}=1}^{\mathtt{n}} |\mathtt{w}_{\mathtt{i}}|$ is added to the cost function:

$$\mathtt{J}(\mathbf{w}) = \mathtt{MSE}(\mathbf{w}) + \lambda \sum_{\mathtt{i}=1}^{\mathtt{n}} |\mathtt{w}_{\mathtt{i}}|$$

Regularization (2/2)

- Lasso regression ($l1$): $\mathtt{R}(\mathbf{w}) = \lambda \sum_{\mathtt{i}=1}^{\mathtt{n}} |\mathtt{w}_\mathtt{i}|$ is added to the cost function:

$$\mathtt{J}(\mathbf{w}) = \mathtt{MSE}(\mathbf{w}) + \lambda \sum_{\mathtt{i}=1}^{\mathtt{n}} |\mathtt{w}_\mathtt{i}|$$

- Ridge regression ($l2$): $\mathtt{R}(\mathbf{w}) = \lambda \sum_{\mathtt{i}=1}^{\mathtt{n}} \mathtt{w}_\mathtt{i}^2$ is added to the cost function.

$$\mathtt{J}(\mathbf{w}) = \mathtt{MSE}(\mathbf{w}) + \lambda \sum_{\mathtt{i}=1}^{\mathtt{n}} \mathtt{w}_\mathtt{i}^2$$

Regularization (2/2)

- Lasso regression ($l1$): $\mathtt{R}(\mathbf{w}) = \lambda \sum_{\mathtt{i}=1}^{\mathtt{n}} |\mathtt{w}_\mathtt{i}|$ is added to the cost function:

$$\mathtt{J}(\mathbf{w}) = \mathtt{MSE}(\mathbf{w}) + \lambda \sum_{\mathtt{i}=1}^{\mathtt{n}} |\mathtt{w}_\mathtt{i}|$$

- Ridge regression ($l2$): $\mathtt{R}(\mathbf{w}) = \lambda \sum_{\mathtt{i}=1}^{\mathtt{n}} \mathtt{w}_\mathtt{i}^2$ is added to the cost function.

$$\mathtt{J}(\mathbf{w}) = \mathtt{MSE}(\mathbf{w}) + \lambda \sum_{\mathtt{i}=1}^{\mathtt{n}} \mathtt{w}_\mathtt{i}^2$$

- ElasticNet: a middle ground between $l1$ and $l2$ regularization.

$$\mathtt{J}(\mathbf{w}) = \mathtt{MSE}(\mathbf{w}) + \alpha\lambda \sum_{\mathtt{i}=1}^{\mathtt{n}} |\mathtt{w}_\mathtt{i}| + (1-\alpha)\lambda \sum_{\mathtt{i}=1}^{\mathtt{n}} \mathtt{w}_\mathtt{i}^2$$

Hyperparameters

Hyperparameters and Validation Sets (1/2)

- Hyperparameters are settings that we can use to control the behavior of a learning algorithm.

Hyperparameters and Validation Sets (1/2)

- Hyperparameters are settings that we can use to control the behavior of a learning algorithm.
- The values of hyperparameters are not adapted by the learning algorithm itself.
  - E.g., the α and λ values for regularization.

Hyperparameters and Validation Sets (1/2)

- Hyperparameters are settings that we can use to control the behavior of a learning algorithm.
- The values of hyperparameters are not adapted by the learning algorithm itself.
  - E.g., the α and λ values for regularization.
- We do not learn the hyperparameter.
  - It is not appropriate to learn that hyperparameter on the training set.
  - If learned on the training set, such hyperparameters would always result in overfitting.

Hyperparameters and Validation Sets (2/2)

- To find hyperparameters, we need a validation set of examples that the training algorithm does not observe.

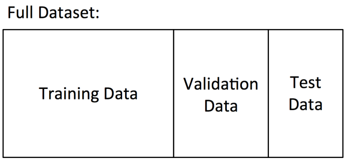
Hyperparameters and Validation Sets (2/2)

- To find hyperparameters, we need a validation set of examples that the training algorithm does not observe.

- We construct the validation set from the training data (not the test data).

Hyperparameters and Validation Sets (2/2)

- To find hyperparameters, we need a validation set of examples that the training algorithm does not observe.
- We construct the validation set from the training data (not the test data).
- We split the training data into two disjoint subsets:
  1. One is used to learn the parameters.
  2. The other one (the validation set) is used to estimate the test error during or after training, allowing for the hyperparameters to be updated accordingly.



Cross-Validation

- Cross-validation: a technique to avoid wasting too much training data in validation sets.

Cross-Validation

- Cross-validation: a technique to avoid wasting too much training data in validation sets.
- The training set is split into complementary subsets.

Cross-Validation

- Cross-validation: a technique to avoid wasting too much training data in validation sets.
- The training set is split into complementary subsets.
- Each model is trained against a different combination of these subsets and validated against the remaining parts.

Logistic Regression

Let's Start with an Example

Example (1/4)

- Given the dataset of `m` cancer tests.

Tumor size	Cancer
330	1
120	0
400	1
⋮	⋮

Example (1/4)

- Given the dataset of `m` cancer tests.

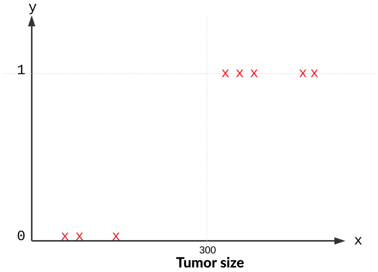
Tumor size	Cancer
330	1
120	0
400	1
⋮	⋮

- Predict the risk of cancer, as a function of the tumor size?

Example (2/4)

Tumor size	Cancer
330	1
120	0
400	1
⋮	⋮

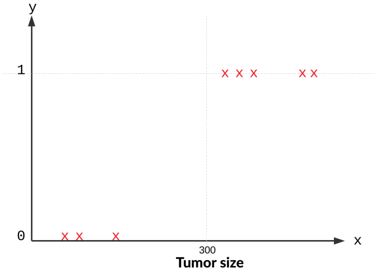
$$\mathbf{x} = \begin{bmatrix} 330 \\ 120 \\ 400 \\ \vdots \end{bmatrix} \qquad \mathbf{y} = \begin{bmatrix} 1 \\ 0 \\ 1 \\ \vdots \end{bmatrix}$$



Example (2/4)

Tumor size	Cancer
330	1
120	0
400	1
⋮	⋮

$$\mathbf{x} = \begin{bmatrix} 330 \\ 120 \\ 400 \\ \vdots \end{bmatrix} \qquad \mathbf{y} = \begin{bmatrix} 1 \\ 0 \\ 1 \\ \vdots \end{bmatrix}$$

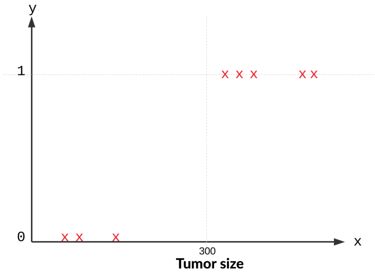


- $\mathbf{x}^{(i)} \in \mathbb{R}$: $x_1^{(i)}$ is the tumor size of the ith instance in the training set.

Example (3/4)

Tumor size	Cancer
330	1
120	0
400	1
⋮	⋮

$$\mathbf{x} = \begin{bmatrix} 330 \\ 120 \\ 400 \\ \vdots \end{bmatrix} \qquad \mathbf{y} = \begin{bmatrix} 1 \\ 0 \\ 1 \\ \vdots \end{bmatrix}$$

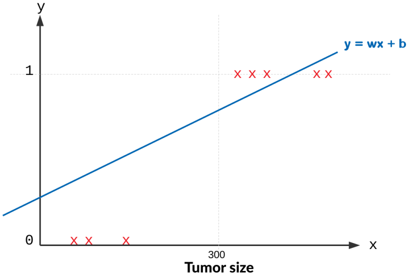


- Predict the risk of cancer $\hat{y}$ as a function of the tumor sizes x_1, i.e., $\hat{y} = f(x_1)$
- E.g., what is $\hat{y}$, if $x_1 = 500$?

Example (3/4)

Tumor size	Cancer
330	1
120	0
400	1
⋮	⋮

$$\mathbf{x} = \begin{bmatrix} 330 \\ 120 \\ 400 \\ \vdots \end{bmatrix} \qquad \mathbf{y} = \begin{bmatrix} 1 \\ 0 \\ 1 \\ \vdots \end{bmatrix}$$

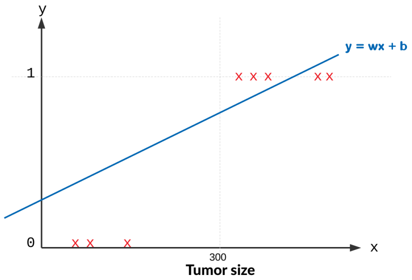


- Predict the risk of cancer $\hat{y}$ as a function of the tumor sizes x_1, i.e., $\hat{y} = f(x_1)$
- E.g., what is $\hat{y}$, if $x_1 = 500$?
- As an initial choice: $\hat{y} = f_w(\mathbf{x}) = w_0 + w_1 x_1$

Example (3/4)

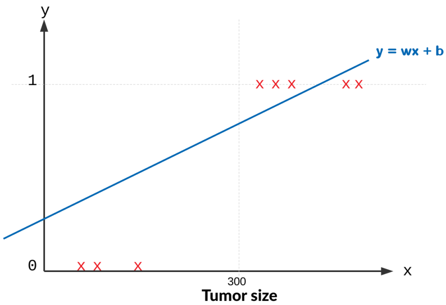
Tumor size	Cancer
330	1
120	0
400	1
⋮	⋮

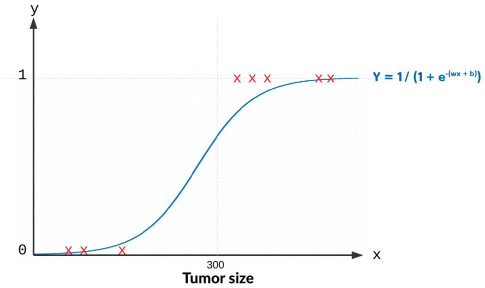
$$\mathbf{x} = \begin{bmatrix} 330 \\ 120 \\ 400 \\ \vdots \end{bmatrix} \qquad \mathbf{y} = \begin{bmatrix} 1 \\ 0 \\ 1 \\ \vdots \end{bmatrix}$$



- Predict the risk of cancer $\hat{y}$ as a function of the tumor sizes x_1, i.e., $\hat{y} = f(x_1)$
- E.g., what is $\hat{y}$, if $x_1 = 500$?
- As an initial choice: $\hat{y} = f_w(\mathbf{x}) = w_0 + w_1 x_1$
- Bad model!

Example (4/4)



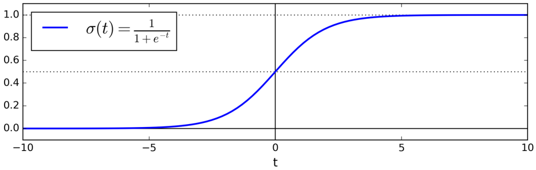


- A better model $\hat{y} = \frac{1}{1+e^{-(w_0+w_1x_1)}}$

Sigmoid Function

- The sigmoid function, denoted by $\sigma(.)$, outputs a number between 0 and 1.

$$\sigma(t) = \frac{1}{1 + e^{-t}}$$

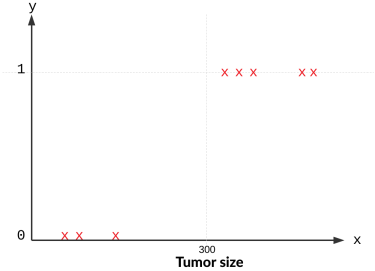


- When $t < 0$, then $\sigma(t) < 0.5$
- when $t \geq 0$, then $\sigma(t) \geq 0.5$

Binomial Logistic Regression

Binomial Logistic Regression (1/2)

- Our goal: to build a system that takes input $\mathbf{x} \in \mathbb{R}^n$ and predicts output $\hat{y} \in \{0, 1\}$.
- To specify which of 2 categories an input $\mathbf{x}$ belongs to.



Binomial Logistic Regression (2/2)

- Linear regression

$$\hat{y} = w_0x_0 + w_1x_1 + w_2x_2 + \cdots + w_nx_n = \mathbf{w}^\intercal\mathbf{x}$$

Binomial Logistic Regression (2/2)

- Linear regression

$$\hat{y} = w_0x_0 + w_1x_1 + w_2x_2 + \cdots + w_nx_n = \mathbf{w}^\intercal\mathbf{x}$$

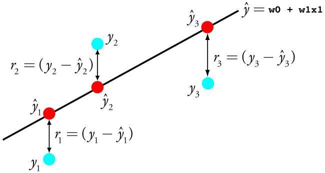
- Binomial logistic regression

$$z = w_0x_0 + w_1x_1 + w_2x_2 + \cdots + w_nx_n = \mathbf{w}^\intercal\mathbf{x}$$

$$\hat{y} = \sigma(z) = \frac{1}{1 + e^{-z}} = \frac{1}{1 + e^{-\mathbf{w}^\intercal\mathbf{x}}}$$

How to Learn Model Parameters $\mathbf{w}$?

Linear Regression - Cost Function



- One reasonable model should make $\hat{y}$ close to y, at least for the training dataset.
- Cost function $J(\mathbf{w})$: the mean squared error (MSE)

$$\text{cost}(\hat{y}^{(i)}, y^{(i)}) = (\hat{y}^{(i)} - y^{(i)})^2$$

$$J(\mathbf{w}) = \frac{1}{m}\sum_{i}^{m} \text{cost}(\hat{y}^{(i)}, y^{(i)}) = \frac{1}{m}\sum_{i}^{m}(\hat{y}^{(i)} - y^{(i)})^2$$

Binomial Logistic Regression - Cost Function (1/5)

- Naive idea: minimizing the Mean Squared Error (MSE)

$$\mathtt{cost}(\hat{y}^{(i)}, y^{(i)}) = (\hat{y}^{(i)} - y^{(i)})^2$$

$$J(\mathbf{w}) = \frac{1}{m}\sum_{i}^{m}\mathtt{cost}(\hat{y}^{(i)}, y^{(i)}) = \frac{1}{m}\sum_{i}^{m}(\hat{y}^{(i)} - y^{(i)})^2$$

Binomial Logistic Regression - Cost Function (1/5)

- Naive idea: minimizing the Mean Squared Error (MSE)

$$\mathtt{cost}(\hat{y}^{(i)}, y^{(i)}) = (\hat{y}^{(i)} - y^{(i)})^2$$

$$J(\mathbf{w}) = \frac{1}{m}\sum_{i}^{m} \mathtt{cost}(\hat{y}^{(i)}, y^{(i)}) = \frac{1}{m}\sum_{i}^{m}(\hat{y}^{(i)} - y^{(i)})^2$$

$$J(\mathbf{w}) = \mathrm{MSE}(\mathbf{w}) = \frac{1}{m}\sum_{i}^{m}\left(\frac{1}{1 + e^{-\mathbf{w}^\intercal \mathbf{x}^{(i)}}} - y^{(i)}\right)^2$$

Binomial Logistic Regression - Cost Function (1/5)

- Naive idea: minimizing the Mean Squared Error (MSE)

$$\text{cost}(\hat{y}^{(i)}, y^{(i)}) = (\hat{y}^{(i)} - y^{(i)})^2$$

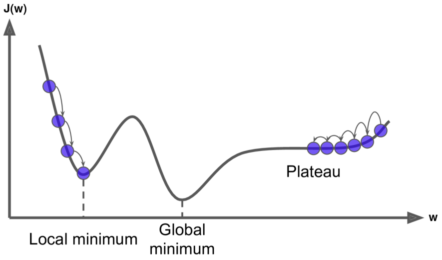
$$J(\mathbf{w}) = \frac{1}{m}\sum_{i}^{m} \text{cost}(\hat{y}^{(i)}, y^{(i)}) = \frac{1}{m}\sum_{i}^{m}(\hat{y}^{(i)} - y^{(i)})^2$$

$$J(\mathbf{w}) = \text{MSE}(\mathbf{w}) = \frac{1}{m}\sum_{i}^{m}\left(\frac{1}{1 + e^{-\mathbf{w}^\intercal \mathbf{x}^{(i)}}} - y^{(i)}\right)^2$$

- This cost function is a non-convex function for parameter optimization.

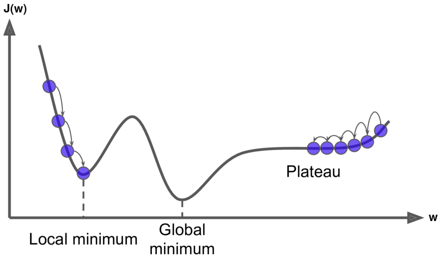
Binomial Logistic Regression - Cost Function (2/5)

- What do we mean by non-convex?
- If a line joining two points on the curve, crosses the curve.
- The algorithm may converge to a local minimum.



Binomial Logistic Regression - Cost Function (2/5)

- What do we mean by non-convex?
- If a line joining two points on the curve, crosses the curve.
- The algorithm may converge to a local minimum.
- We want a convex logistic regression cost function $J(\mathbf{w})$.



Binomial Logistic Regression - Cost Function (3/5)

- The predicted value $\hat{y} = \sigma(\mathbf{w}^\intercal \mathbf{x}) = \frac{1}{1+e^{-\mathbf{w}^\intercal \mathbf{x}}}$
- $\texttt{cost}(\hat{y}^{(i)}, y^{(i)}) = ?$

Binomial Logistic Regression - Cost Function (3/5)

- The predicted value $\hat{y} = \sigma(\mathbf{w}^\intercal \mathbf{x}) = \frac{1}{1+e^{-\mathbf{w}^\intercal \mathbf{x}}}$
- $\texttt{cost}(\hat{y}^{(i)}, y^{(i)}) = ?$
- The $\texttt{cost}(\hat{y}^{(i)}, y^{(i)})$ should be
  - Close to 0, if the predicted value $\hat{y}$ will be close to true value y.
  - Large, if the predicted value $\hat{y}$ will be far from the true value y.

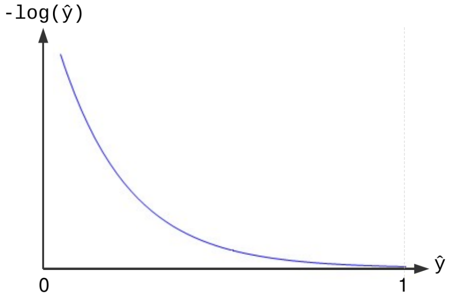
Binomial Logistic Regression - Cost Function (3/5)

- The predicted value $\hat{y} = \sigma(\mathbf{w}^\intercal \mathbf{x}) = \frac{1}{1+e^{-\mathbf{w}^\intercal \mathbf{x}}}$
- $\mathtt{cost}(\hat{y}^{(i)}, y^{(i)}) = ?$
- The $\mathtt{cost}(\hat{y}^{(i)}, y^{(i)})$ should be
  - Close to 0, if the predicted value $\hat{y}$ will be close to true value y.
  - Large, if the predicted value $\hat{y}$ will be far from the true value y.

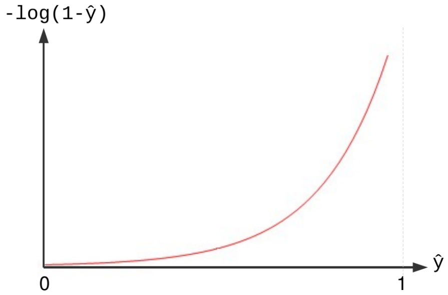
$$\mathtt{cost}(\hat{y}^{(i)}, y^{(i)}) = \begin{cases} -\log(\hat{y}^{(i)}) & \text{if} \quad y^{(i)} = 1 \\ -\log(1 - \hat{y}^{(i)}) & \text{if} \quad y^{(i)} = 0 \end{cases}$$

Binomial Logistic Regression - Cost Function (4/5)

$$\texttt{cost}(\hat{y}^{(i)}, y^{(i)}) = \begin{cases} -\log(\hat{y}^{(i)}) & \text{if} \quad y^{(i)} = 1 \\ -\log(1 - \hat{y}^{(i)}) & \text{if} \quad y^{(i)} = 0 \end{cases}$$



when $y = 1$



when $y = 0$

Binomial Logistic Regression - Cost Function (5/5)

- We can define $\mathtt{J}(\mathbf{w})$ as below

$$\mathtt{cost}(\hat{\mathtt{y}}^{(\mathtt{i})}, \mathtt{y}^{(\mathtt{i})}) = \begin{cases} -\mathtt{log}(\hat{\mathtt{y}}^{(\mathtt{i})}) & \text{if} \quad \mathtt{y}^{(\mathtt{i})} = 1 \\ -\mathtt{log}(1 - \hat{\mathtt{y}}^{(\mathtt{i})}) & \text{if} \quad \mathtt{y}^{(\mathtt{i})} = 0 \end{cases}$$

Binomial Logistic Regression - Cost Function (5/5)

- We can define $\mathtt{J}(\mathbf{w})$ as below

$$\mathtt{cost}(\hat{y}^{(i)}, y^{(i)}) = \begin{cases} -\log(\hat{y}^{(i)}) & \text{if} \quad y^{(i)} = 1 \\ -\log(1 - \hat{y}^{(i)}) & \text{if} \quad y^{(i)} = 0 \end{cases}$$

$$\mathtt{J}(\mathbf{w}) = \frac{1}{m}\sum_{i}^{m} \mathtt{cost}(\hat{y}^{(i)}, y^{(i)}) = -\frac{1}{m}\sum_{i}^{m}(y^{(i)}\log(\hat{y}^{(i)}) + (1 - y^{(i)})\log(1 - \hat{y}^{(i)}))$$

How to Learn Model Parameters $\mathbf{w}$?

- We want to choose $\mathbf{w}$ so as to minimize $J(\mathbf{w})$.
- An approach to find $\mathbf{w}$: gradient descent
  - Batch gradient descent
  - Stochastic gradient descent
  - Mini-batch gradient descent

Binomial Logistic Regression Gradient Descent (1/2)

- Goal: find $\mathbf{w}$ that minimizes $J(\mathbf{w}) = -\frac{1}{m}\sum_{i}^{m}(y^{(i)}\log(\hat{y}^{(i)}) + (1 - y^{(i)})\log(1 - \hat{y}^{(i)}))$.

Binomial Logistic Regression Gradient Descent (1/2)

- Goal: find $\mathbf{w}$ that minimizes $J(\mathbf{w}) = -\frac{1}{m}\sum_{i}^{m}(y^{(i)}\log(\hat{y}^{(i)}) + (1 - y^{(i)})\log(1 - \hat{y}^{(i)}))$.
- Start at a random point, and repeat the following steps, until the stopping criterion is satisfied:

Binomial Logistic Regression Gradient Descent (1/2)

- Goal: find $\mathbf{w}$ that minimizes $J(\mathbf{w}) = -\frac{1}{m}\sum_{i}^{m}(y^{(i)}\log(\hat{y}^{(i)}) + (1 - y^{(i)})\log(1 - \hat{y}^{(i)}))$.
- Start at a random point, and repeat the following steps, until the stopping criterion is satisfied:
  1. Determine a descent direction $\frac{\partial J(\mathbf{w})}{\partial \mathbf{w}}$

Binomial Logistic Regression Gradient Descent (1/2)

- Goal: find $\mathbf{w}$ that minimizes $\mathtt{J}(\mathbf{w}) = -\frac{1}{\mathtt{m}}\sum_{\mathtt{i}}^{\mathtt{m}}(\mathtt{y}^{(\mathtt{i})}\mathtt{log}(\hat{\mathtt{y}}^{(\mathtt{i})}) + (1 - \mathtt{y}^{(\mathtt{i})})\mathtt{log}(1 - \hat{\mathtt{y}}^{(\mathtt{i})}))$.
- Start at a random point, and repeat the following steps, until the stopping criterion is satisfied:
  1. Determine a descent direction $\frac{\partial \mathtt{J}(\mathbf{w})}{\partial \mathtt{w}}$
  2. Choose a step size η

Binomial Logistic Regression Gradient Descent (1/2)

- Goal: find $\mathbf{w}$ that minimizes $J(\mathbf{w}) = -\frac{1}{m}\sum_{i}^{m}(y^{(i)}\log(\hat{y}^{(i)}) + (1 - y^{(i)})\log(1 - \hat{y}^{(i)}))$.
- Start at a random point, and repeat the following steps, until the stopping criterion is satisfied:
  1. Determine a descent direction $\frac{\partial J(\mathbf{w})}{\partial w}$
  2. Choose a step size η
  3. Update the parameters: $w^{(\text{next})} = w - \eta\frac{\partial J(\mathbf{w})}{\partial w}$ (simultaneously for all parameters)

Binomial Logistic Regression Gradient Descent (2/2)

- 1. Determine a descent direction $\frac{\partial J(\mathbf{w})}{\partial \mathtt{w}}$.

$$J(\mathbf{w}) = \frac{1}{m}\sum_{i}^{m} \mathtt{cost}(\hat{y}^{(i)}, y^{(i)}) = -\frac{1}{m}\sum_{i}^{m}(y^{(i)}\log(\hat{y}^{(i)}) + (1 - y^{(i)})\log(1 - \hat{y}^{(i)}))$$

$$\frac{\partial J(\mathbf{w})}{\partial w_j} = \frac{1}{m}\sum_{i}^{m}(\hat{y}^{(i)} - y^{(i)})x_j$$

Binomial Logistic Regression Gradient Descent (2/2)

- 1. Determine a descent direction $\frac{\partial J(\mathbf{w})}{\partial w}$.

$$J(\mathbf{w}) = \frac{1}{m}\sum_{i}^{m} \mathtt{cost}(\hat{y}^{(i)}, y^{(i)}) = -\frac{1}{m}\sum_{i}^{m}(y^{(i)}\log(\hat{y}^{(i)}) + (1 - y^{(i)})\log(1 - \hat{y}^{(i)}))$$

$$\frac{\partial J(\mathbf{w})}{\partial w_j} = \frac{1}{m}\sum_{i}^{m}(\hat{y}^{(i)} - y^{(i)})x_j$$

- 2. Choose a step size η

Binomial Logistic Regression Gradient Descent (2/2)

- 1. Determine a descent direction $\frac{\partial J(\mathbf{w})}{\partial w}$.

$$J(\mathbf{w}) = \frac{1}{m}\sum_{i}^{m} \text{cost}(\hat{y}^{(i)}, y^{(i)}) = -\frac{1}{m}\sum_{i}^{m}(y^{(i)}\log(\hat{y}^{(i)}) + (1 - y^{(i)})\log(1 - \hat{y}^{(i)}))$$

$$\frac{\partial J(\mathbf{w})}{\partial w_j} = \frac{1}{m}\sum_{i}^{m}(\hat{y}^{(i)} - y^{(i)})x_j$$

- 2. Choose a step size η

- 3. Update the parameters: $w_j^{(next)} = w_j - \eta\frac{\partial J(\mathbf{w})}{\partial w_j}$
  - $0 \leq j \leq n$, where n is the number of features.

Binomial Logistic Regression Gradient Descent - Example (1/4)

Tumor size	Cancer
330	1
120	0
400	1

$$\mathbf{X} = \begin{bmatrix} 1 & 330 \\ 1 & 120 \\ 1 & 400 \end{bmatrix} \qquad \mathbf{y} = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}$$

- Predict the risk of cancer $\hat{y}$ as a function of the tumor sizes x_1.
- E.g., what is $\hat{y}$, if $x_1 = 500$?

Binomial Logistic Regression Gradient Descent - Example (2/4)

$$\mathbf{X} = \left[\begin{array}{c|c} 1 & 330 \\ 1 & 120 \\ 1 & 400 \end{array}\right] \qquad \mathbf{y} = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}$$

$$\hat{y} = \sigma(w_0 + w_1 x_1) = \frac{1}{1 + e^{-(w_0 + w_1 x_1)}}$$

$$J(\mathbf{w}) = -\frac{1}{m}\sum_{i}^{m}(y^{(i)}\log(\hat{y}^{(i)}) + (1 - y^{(i)})\log(1 - \hat{y}^{(i)}))$$

Binomial Logistic Regression Gradient Descent - Example (2/4)

$$\mathbf{X} = \left[\begin{array}{c|c} 1 & 330 \\ 1 & 120 \\ 1 & 400 \end{array}\right] \qquad \mathbf{y} = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}$$

$$\hat{y} = \sigma(w_0 + w_1 x_1) = \frac{1}{1 + e^{-(w_0 + w_1 x_1)}}$$

$$J(\mathbf{w}) = -\frac{1}{m}\sum_{i}^{m}(y^{(i)}\log(\hat{y}^{(i)}) + (1 - y^{(i)})\log(1 - \hat{y}^{(i)}))$$

$$\frac{\partial J(\mathbf{w})}{\partial w_0} = \frac{1}{3}\sum_{i}^{3}(\hat{y}^{(i)} - y^{(i)})x_0$$

$$= \frac{1}{3}[(\frac{1}{1 + e^{-(w_0 + 330 w_1)}} - 1) + (\frac{1}{1 + e^{-(w_0 + 120 w_1)}} - 0) + (\frac{1}{1 + e^{-(w_0 + 400 w_1)}} - 1)]$$

Binomial Logistic Regression Gradient Descent - Example (3/4)

$$\mathbf{X} = \left[\begin{array}{c|c} 1 & 330 \\ 1 & 120 \\ 1 & 400 \end{array}\right] \qquad \mathbf{y} = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}$$

$$\hat{y} = \sigma(w_0 + w_1 x_1) = \frac{1}{1 + e^{-(w_0 + w_1 x_1)}}$$

$$J(\mathbf{w}) = -\frac{1}{m}\sum_{i}^{m}(y^{(i)}\log(\hat{y}^{(i)}) + (1 - y^{(i)})\log(1 - \hat{y}^{(i)}))$$

Binomial Logistic Regression Gradient Descent - Example (3/4)

$$\mathbf{X} = \begin{bmatrix} 1 & 330 \\ 1 & 120 \\ 1 & 400 \end{bmatrix} \qquad \mathbf{y} = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}$$

$$\hat{y} = \sigma(w_0 + w_1 x_1) = \frac{1}{1 + e^{-(w_0 + w_1 x_1)}}$$

$$J(\mathbf{w}) = -\frac{1}{m}\sum_{i}^{m}(y^{(i)}\log(\hat{y}^{(i)}) + (1 - y^{(i)})\log(1 - \hat{y}^{(i)}))$$

$$\frac{\partial J(\mathbf{w})}{\partial w_1} = \frac{1}{3}\sum_{i}^{3}(\hat{y}^{(i)} - y^{(i)})x_1$$

$$= \frac{1}{3}[330(\frac{1}{1 + e^{-(w_0 + 330 w_1)}} - 1) + 120(\frac{1}{1 + e^{-(w_0 + 120 w_1)}} - 0) + 400(\frac{1}{1 + e^{-(w_0 + 400 w_1)}} - 1)]$$

Binomial Logistic Regression Gradient Descent - Example (4/4)

$$\mathtt{w}_0^{(\mathtt{next})} = \mathtt{w}_0 - \eta \frac{\partial \mathtt{J}(\mathbf{w})}{\partial \mathtt{w}_0}$$

$$\mathtt{w}_1^{(\mathtt{next})} = \mathtt{w}_1 - \eta \frac{\partial \mathtt{J}(\mathbf{w})}{\partial \mathtt{w}_1}$$

Logistic Regression in TensorFlow - First Implementation

```
x_train = [1, 2, 3]
y_train = [1, 2, 3]

X = tf.placeholder(tf.float32)
y_true = tf.placeholder(tf.float32)

w = tf.Variable(5.)
b = tf.Variable(5.)
```

Logistic Regression in TensorFlow - First Implementation

```
x_train = [1, 2, 3]
y_train = [1, 2, 3]

X = tf.placeholder(tf.float32)
y_true = tf.placeholder(tf.float32)

w = tf.Variable(5.)
b = tf.Variable(5.)
```

```
z = tf.matmul(w, tf.transpose(x)) + b
y_hat = tf.sigmoid(z)

cost = -y_true * tf.log(y_hat) - (1 - y_true) * tf.log(1 - y_hat)
cost = tf.reduce_mean(cost)
```

Logistic Regression in TensorFlow - First Implementation

```
x_train = [1, 2, 3]
y_train = [1, 2, 3]

X = tf.placeholder(tf.float32)
y_true = tf.placeholder(tf.float32)

w = tf.Variable(5.)
b = tf.Variable(5.)
```

```
z = tf.matmul(w, tf.transpose(x)) + b
y_hat = tf.sigmoid(z)

cost = -y_true * tf.log(y_hat) - (1 - y_true) * tf.log(1 - y_hat)
cost = tf.reduce_mean(cost)
```

```
learning_rate = 0.1
optimizer = tf.train.GradientDescentOptimizer(learning_rate=learning_rate)
op = optimizer.minimize(cost)
```

Logistic Regression in TensorFlow - Second Implementation

```
x_train = [1, 2, 3]
y_train = [1, 2, 3]

X = tf.placeholder(tf.float32)
y_true = tf.placeholder(tf.float32)

w = tf.Variable(5.)
b = tf.Variable(5.)
```

Logistic Regression in TensorFlow - Second Implementation

```
x_train = [1, 2, 3]
y_train = [1, 2, 3]

X = tf.placeholder(tf.float32)
y_true = tf.placeholder(tf.float32)

w = tf.Variable(5.)
b = tf.Variable(5.)
```

```
z = tf.matmul(w, tf.transpose(x)) + b
y_hat = tf.sigmoid(z)

cost = tf.nn.sigmoid_cross_entropy_with_logits(labels=y_true, logits=z)
cost = tf.reduce_mean(cost)
```

Logistic Regression in TensorFlow - Second Implementation

```
x_train = [1, 2, 3]
y_train = [1, 2, 3]

X = tf.placeholder(tf.float32)
y_true = tf.placeholder(tf.float32)

w = tf.Variable(5.)
b = tf.Variable(5.)
```

```
z = tf.matmul(w, tf.transpose(x)) + b
y_hat = tf.sigmoid(z)

cost = tf.nn.sigmoid_cross_entropy_with_logits(labels=y_true, logits=z)
cost = tf.reduce_mean(cost)
```

```
learning_rate = 0.1
optimizer = tf.train.GradientDescentOptimizer(learning_rate=learning_rate)
op = optimizer.minimize(cost)
```

Multinomial Logistic Regression

Multinomial Logistic Regression

- Multinomial classifiers can distinguish between more than two classes.
- Instead of $\mathtt{y} \in \{0, 1\}$, we have $\mathtt{y} \in \{1, 2, \cdots, \mathtt{k}\}$.

Binomial vs. Multinomial Logistic Regression (1/2)

- In a binomial classifier, $y \in \{0, 1\}$, the estimator is $\hat{y} = p(y = 1 \mid \mathbf{x}; \mathbf{w})$.
  - We find one set of parameters $\mathbf{w}$.

$$\mathbf{w}^\intercal = [w_0, w_1, \cdots, w_n]$$

Binomial vs. Multinomial Logistic Regression (1/2)

- In a binomial classifier, $y \in \{0, 1\}$, the estimator is $\hat{y} = p(y = 1 \mid \mathbf{x}; \mathbf{w})$.
  - We find one set of parameters $\mathbf{w}$.

$$\mathbf{w}^\intercal = [w_0, w_1, \cdots, w_n]$$

- In multinomial classifier, $y \in \{1, 2, \cdots, k\}$, we need to estimate the result for each individual label, i.e., $\hat{y}_j = p(y = j \mid \mathbf{x}; \mathbf{w})$.

Binomial vs. Multinomial Logistic Regression (1/2)

- In a binomial classifier, $y \in \{0, 1\}$, the estimator is $\hat{y} = p(y = 1 \mid \mathbf{x}; \mathbf{w})$.
  - We find one set of parameters $\mathbf{w}$.

$$\mathbf{w}^\mathsf{T} = [w_0, w_1, \cdots, w_n]$$

- In multinomial classifier, $y \in \{1, 2, \cdots, k\}$, we need to estimate the result for each individual label, i.e., $\hat{y}_j = p(y = j \mid \mathbf{x}; \mathbf{w})$.
  - We find k set of parameters $\mathbf{W}$.

$$\mathbf{W} = \begin{bmatrix} [w_{0,1}, w_{1,1}, \cdots, w_{n,1}] \\ [w_{0,2}, w_{1,2}, \cdots, w_{n,2}] \\ \vdots \\ [w_{0,k}, w_{1,k}, \cdots, w_{n,k}] \end{bmatrix} = \begin{bmatrix} \mathbf{w}_1^\mathsf{T} \\ \mathbf{w}_2^\mathsf{T} \\ \vdots \\ \mathbf{w}_k^\mathsf{T} \end{bmatrix}$$

Binomial vs. Multinomial Logistic Regression (2/2)

- In a binary class, $y \in \{0, 1\}$, we use the sigmoid function.

$$\mathbf{w}^\intercal \mathbf{x} = w_0 x_0 + w_1 x_1 + \cdots + w_n x_n$$

$$\hat{y} = p(y = 1 \mid \mathbf{x}; \mathbf{w}) = \sigma(\mathbf{w}^\intercal \mathbf{x}) = \frac{1}{1 + e^{-\mathbf{w}^\intercal \mathbf{x}}}$$

Binomial vs. Multinomial Logistic Regression (2/2)

- In a binary class, $y \in \{0, 1\}$, we use the sigmoid function.

$$\mathbf{w}^\mathsf{T}\mathbf{x} = w_0 x_0 + w_1 x_1 + \cdots + w_n x_n$$

$$\hat{y} = p(y = 1 \mid \mathbf{x}; \mathbf{w}) = \sigma(\mathbf{w}^\mathsf{T}\mathbf{x}) = \frac{1}{1 + e^{-\mathbf{w}^\mathsf{T}\mathbf{x}}}$$

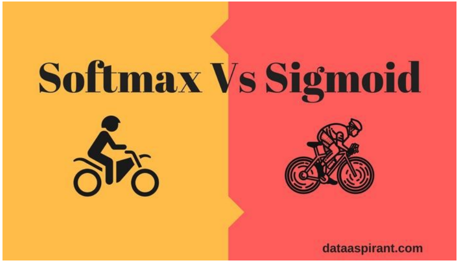
- In multiclasses, $y \in \{1, 2, \cdots, k\}$, we use the softmax function.

$$\mathbf{w}_j^\mathsf{T}\mathbf{x} = w_{0,j} x_0 + w_{1,j} x_1 + \cdots + w_{n,j} x_n, \ 1 \leq j \leq k$$

$$\hat{y}_j = p(y = j \mid \mathbf{x}; \mathbf{w}_j) = \sigma(\mathbf{w}_j^\mathsf{T}\mathbf{x}) = \frac{e^{\mathbf{w}_j^\mathsf{T}\mathbf{x}}}{\sum_{i=1}^{k} e^{\mathbf{w}_i^\mathsf{T}\mathbf{x}}}$$

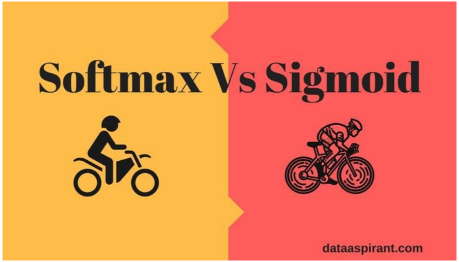
Sigmoid vs. Softmax

- Sigmoid function: $\sigma(\mathbf{w}^\intercal\mathbf{x}) = \frac{1}{1+e^{-\mathbf{w}^\intercal\mathbf{x}}}$



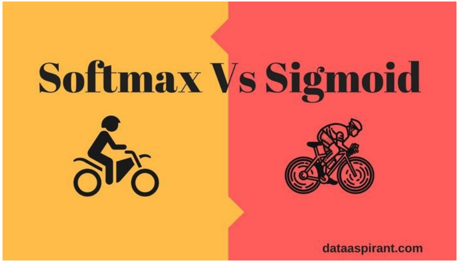
Sigmoid vs. Softmax

- Sigmoid function: $\sigma(\mathbf{w}^\top \mathbf{x}) = \frac{1}{1+e^{-\mathbf{w}^\top \mathbf{x}}}$
- Softmax function: $\sigma(\mathbf{w}_j^\top \mathbf{x}) = \frac{e^{\mathbf{w}_j^\top \mathbf{x}}}{\sum_{i=1}^{k} e^{\mathbf{w}_i^\top \mathbf{x}}}$



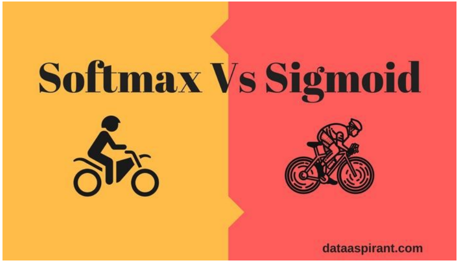
Sigmoid vs. Softmax

- Sigmoid function: $\sigma(\mathbf{w}^\intercal \mathbf{x}) = \frac{1}{1+e^{-\mathbf{w}^\intercal \mathbf{x}}}$
- Softmax function: $\sigma(\mathbf{w}_j^\intercal \mathbf{x}) = \frac{e^{\mathbf{w}_j^\intercal \mathbf{x}}}{\sum_{i=1}^{k} e^{\mathbf{w}_i^\intercal \mathbf{x}}}$
  - Calculate the probabilities of each target class over all possible target classes.



Sigmoid vs. Softmax

- Sigmoid function: $\sigma(\mathbf{w}^\mathsf{T}\mathbf{x}) = \frac{1}{1+e^{-\mathbf{w}^\mathsf{T}\mathbf{x}}}$
- Softmax function: $\sigma(\mathbf{w}_j^\mathsf{T}\mathbf{x}) = \frac{e^{\mathbf{w}_j^\mathsf{T}\mathbf{x}}}{\sum_{i=1}^{k} e^{\mathbf{w}_i^\mathsf{T}\mathbf{x}}}$
  - Calculate the probabilities of each target class over all possible target classes.
  - The softmax function for two classes is equivalent the sigmoid function.



Softmax Model Estimation and Prediction - Example (1/2)

- Assume we have a training set consisting of $\mathtt{m} = 4$ instances from $\mathtt{k} = 3$ classes.

$$\mathbf{x}^{(1)} \rightarrow \mathtt{class1}, \mathbf{y}^{(1)\intercal} = [1\ 0\ 0]$$
$$\mathbf{x}^{(2)} \rightarrow \mathtt{class2}, \mathbf{y}^{(2)\intercal} = [0\ 1\ 0]$$
$$\mathbf{x}^{(3)} \rightarrow \mathtt{class3}, \mathbf{y}^{(3)\intercal} = [0\ 0\ 1]$$
$$\mathbf{x}^{(4)} \rightarrow \mathtt{class3}, \mathbf{y}^{(4)\intercal} = [0\ 0\ 1]$$

$$\mathbf{Y} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix}$$

Softmax Model Estimation and Prediction - Example (1/2)

- Assume we have a training set consisting of $\mathtt{m} = 4$ instances from $\mathtt{k} = 3$ classes.

$$\mathbf{x}^{(1)} \rightarrow \mathtt{class1}, \mathbf{y}^{(1)\intercal} = [1\ 0\ 0]$$
$$\mathbf{x}^{(2)} \rightarrow \mathtt{class2}, \mathbf{y}^{(2)\intercal} = [0\ 1\ 0]$$
$$\mathbf{x}^{(3)} \rightarrow \mathtt{class3}, \mathbf{y}^{(3)\intercal} = [0\ 0\ 1]$$
$$\mathbf{x}^{(4)} \rightarrow \mathtt{class3}, \mathbf{y}^{(4)\intercal} = [0\ 0\ 1]$$

$$\mathbf{Y} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix}$$

- Assume training set $\mathbf{X}$ and random parameters $\mathbf{W}$ are as below:

$$\mathbf{X} = \left[\begin{array}{c|cc} 1 & 0.1 & 0.5 \\ 1 & 1.1 & 2.3 \\ 1 & -1.1 & -2.3 \\ 1 & -1.5 & -2.5 \end{array}\right] \qquad \mathbf{W} = \begin{bmatrix} 0.01 & 0.1 & 0.1 \\ 0.1 & 0.2 & 0.3 \\ 0.1 & 0.2 & 0.3 \end{bmatrix}$$

Softmax Model Estimation and Prediction - Example (2/2)

- Now, let's compute the softmax activation:

$$\hat{y}_j^{(i)} = p(y^{(i)} = j \mid \mathbf{x}^{(i)}; \mathbf{w}_j) = \sigma(\mathbf{w}_j^T \mathbf{x}^{(i)}) = \frac{e^{\mathbf{w}_j^T \mathbf{x}^{(i)}}}{\sum_{l=1}^{k} e^{\mathbf{w}_l^T \mathbf{x}^{(i)}}}$$

Softmax Model Estimation and Prediction - Example (2/2)

- Now, let's compute the softmax activation:

$$\hat{y}_j^{(i)} = p(y^{(i)} = j \mid \mathbf{x}^{(i)}; \mathbf{w}_j) = \sigma(\mathbf{w}_j^\intercal \mathbf{x}^{(i)}) = \frac{e^{\mathbf{w}_j^\intercal \mathbf{x}^{(i)}}}{\sum_{l=1}^{k} e^{\mathbf{w}_l^\intercal \mathbf{x}^{(i)}}}$$

$$\hat{\mathbf{Y}} = \begin{bmatrix} \hat{\mathbf{y}}^{(1)\intercal} \\ \hat{\mathbf{y}}^{(2)\intercal} \\ \hat{\mathbf{y}}^{(3)\intercal} \\ \hat{\mathbf{y}}^{(4)\intercal} \end{bmatrix} = \begin{bmatrix} 0.29 & 0.34 & 0.36 \\ 0.21 & 0.33 & 0.46 \\ 0.43 & 0.33 & 0.24 \\ 0.45 & 0.33 & 0.22 \end{bmatrix} \quad \text{the predicted classes} = \begin{bmatrix} 3 \\ 3 \\ 1 \\ 1 \end{bmatrix} \quad \text{The correct classes} = \begin{bmatrix} 1 \\ 2 \\ 3 \\ 3 \end{bmatrix}$$

- They are terribly wrong.

Softmax Model Estimation and Prediction - Example (2/2)

- Now, let's compute the softmax activation:

$$\hat{y}_j^{(i)} = p(y^{(i)} = j \mid \mathbf{x}^{(i)}; \mathbf{w}_j) = \sigma(\mathbf{w}_j^\intercal \mathbf{x}^{(i)}) = \frac{e^{\mathbf{w}_j^\intercal \mathbf{x}^{(i)}}}{\sum_{l=1}^{k} e^{\mathbf{w}_l^\intercal \mathbf{x}^{(i)}}}$$

$$\hat{\mathbf{Y}} = \begin{bmatrix} \hat{\mathbf{y}}^{(1)\intercal} \\ \hat{\mathbf{y}}^{(2)\intercal} \\ \hat{\mathbf{y}}^{(3)\intercal} \\ \hat{\mathbf{y}}^{(4)\intercal} \end{bmatrix} = \begin{bmatrix} 0.29 & 0.34 & 0.36 \\ 0.21 & 0.33 & 0.46 \\ 0.43 & 0.33 & 0.24 \\ 0.45 & 0.33 & 0.22 \end{bmatrix} \quad \text{the predicted classes} = \begin{bmatrix} 3 \\ 3 \\ 1 \\ 1 \end{bmatrix} \quad \text{The correct classes} = \begin{bmatrix} 1 \\ 2 \\ 3 \\ 3 \end{bmatrix}$$

- They are terribly wrong.
- We need to update the weights based on the cost function.

Softmax Model Estimation and Prediction - Example (2/2)

- Now, let's compute the softmax activation:

$$\hat{y}_j^{(i)} = p(y^{(i)} = j \mid \mathbf{x}^{(i)}; \mathbf{w}_j) = \sigma(\mathbf{w}_j^\mathsf{T}\mathbf{x}^{(i)}) = \frac{e^{\mathbf{w}_j^\mathsf{T}\mathbf{x}^{(i)}}}{\sum_{l=1}^{k} e^{\mathbf{w}_l^\mathsf{T}\mathbf{x}^{(i)}}}$$

$$\hat{\mathbf{Y}} = \begin{bmatrix} \hat{\mathbf{y}}^{(1)\mathsf{T}} \\ \hat{\mathbf{y}}^{(2)\mathsf{T}} \\ \hat{\mathbf{y}}^{(3)\mathsf{T}} \\ \hat{\mathbf{y}}^{(4)\mathsf{T}} \end{bmatrix} = \begin{bmatrix} 0.29 & 0.34 & 0.36 \\ 0.21 & 0.33 & 0.46 \\ 0.43 & 0.33 & 0.24 \\ 0.45 & 0.33 & 0.22 \end{bmatrix} \quad \text{the predicted classes} = \begin{bmatrix} 3 \\ 3 \\ 1 \\ 1 \end{bmatrix} \quad \text{The correct classes} = \begin{bmatrix} 1 \\ 2 \\ 3 \\ 3 \end{bmatrix}$$

- They are terribly wrong.
- We need to update the weights based on the cost function.
- What is the cost function?

Multinomial Logistic Regression - Cost Function (1/2)

- The objective is to have a model that estimates a high probability for the target class, and consequently a low probability for the other classes.

Multinomial Logistic Regression - Cost Function (1/2)

- The objective is to have a model that estimates a high probability for the target class, and consequently a low probability for the other classes.
- Cost function: the cross-entropy between the correct classes and predicted class for all classes.

$$J(\mathbf{w}_j) = -\frac{1}{m}\sum_{i=1}^{m}\sum_{j=1}^{k} y_j^{(i)} \log(\hat{y}_j^{(i)})$$

Multinomial Logistic Regression - Cost Function (1/2)

- The objective is to have a model that estimates a high probability for the target class, and consequently a low probability for the other classes.
- Cost function: the cross-entropy between the correct classes and predicted class for all classes.

$$J(\mathbf{w}_j) = -\frac{1}{m}\sum_{i=1}^{m}\sum_{j=1}^{k} y_j^{(i)} \log(\hat{y}_j^{(i)})$$

- $y_j^{(i)}$ is 1 if the target class for the ith instance is j, otherwise, it is 0.

Multinomial Logistic Regression - Cost Function (2/2)

- If there are two classes ($k = 2$), this cost function is equivalent to the logistic regression's cost function.

$$J(\mathbf{w}) = -\frac{1}{m}\sum_{i=1}^{m}[y^{(i)}\log(\hat{y}^{(i)}) + (1 - y^{(i)})\log(1 - \hat{y}^{(i)})]$$

How to Learn Model Parameters $\mathbf{W}$?

- Goal: find $\mathbf{W}$ that minimizes $J(\mathbf{W})$.

How to Learn Model Parameters $\mathbf{W}$?

- Goal: find $\mathbf{W}$ that minimizes $J(\mathbf{W})$.
- Start at a random point, and repeat the following steps, until the stopping criterion is satisfied:

How to Learn Model Parameters $\mathbf{W}$?

- Goal: find $\mathbf{W}$ that minimizes $\mathtt{J}(\mathbf{W})$.
- Start at a random point, and repeat the following steps, until the stopping criterion is satisfied:
  1. Determine a descent direction $\frac{\partial \mathtt{J}(\mathbf{W})}{\partial \mathtt{w}}$

How to Learn Model Parameters $\mathbf{W}$?

- Goal: find $\mathbf{W}$ that minimizes $\mathtt{J}(\mathbf{W})$.
- Start at a random point, and repeat the following steps, until the stopping criterion is satisfied:
  1. Determine a descent direction $\frac{\partial \mathtt{J}(\mathbf{W})}{\partial \mathtt{w}}$
  2. Choose a step size η

How to Learn Model Parameters **W**?

- Goal: find **W** that minimizes $\mathtt{J}(\mathbf{W})$.
- Start at a random point, and repeat the following steps, until the stopping criterion is satisfied:
  1. Determine a descent direction $\frac{\partial \mathtt{J}(\mathbf{W})}{\partial \mathtt{w}}$
  2. Choose a step size η
  3. Update the parameters: $\mathtt{w}^{(\mathtt{next})} = \mathtt{w} - \eta \frac{\partial \mathtt{J}(\mathbf{W})}{\partial \mathtt{w}}$ (simultaneously for all parameters)

Performance Measures

Evaluation of Classification Models (1/3)

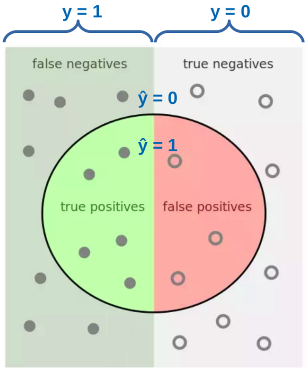
- In a classification problem, there exists a true output y and a model-generated predicted output $\hat{y}$ for each data point.

Evaluation of Classification Models (1/3)

- In a classification problem, there exists a true output y and a model-generated predicted output $\hat{y}$ for each data point.
- The results for each instance point can be assigned to one of four categories:
  - True Positive (TP)
  - True Negative (TN)
  - False Positive (FP)
  - False Negative (FN)

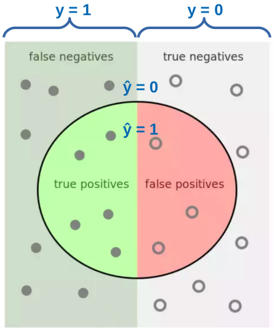
Evaluation of Classification Models (2/3)

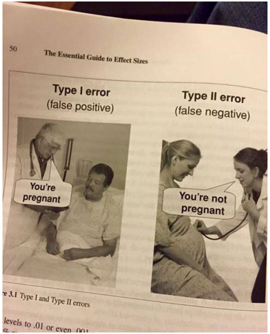
- True Positive (TP): the label y is positive and prediction $\hat{y}$ is also positive.
- True Negative (TN): the label y is negative and prediction $\hat{y}$ is also negative.



Evaluation of Classification Models (3/3)

- False Positive (FP): the label y is negative but prediction $\hat{y}$ is positive (type I error).
- False Negative (FN): the label y is positive but prediction $\hat{y}$ is negative (type II error).





Why Pure Accuracy Is Not A Good Metric?

- Accuracy: how close the prediction is to the true value.

Why Pure Accuracy Is Not A Good Metric?

- Accuracy: how close the prediction is to the true value.
- Assume a highly unbalanced dataset
- E.g., a dataset where 95% of the data points are not fraud and 5% of the data points are fraud.

Why Pure Accuracy Is Not A Good Metric?

- Accuracy: how close the prediction is to the true value.
- Assume a highly unbalanced dataset
- E.g., a dataset where 95% of the data points are not fraud and 5% of the data points are fraud.
- A a naive classifier that predicts not fraud, regardless of input, will be 95% accurate.

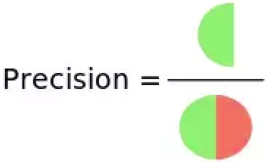
Why Pure Accuracy Is Not A Good Metric?

- Accuracy: how close the prediction is to the true value.
- Assume a highly unbalanced dataset
- E.g., a dataset where 95% of the data points are not fraud and 5% of the data points are fraud.
- A a naive classifier that predicts not fraud, regardless of input, will be 95% accurate.
- For this reason, metrics like precision and recall are typically used.

Precision

- It is the accuracy of the positive predictions.

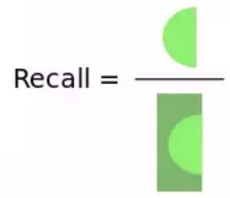
$$\texttt{Precision} = p(y = 1 \mid \hat{y} = 1) = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}}$$



Recall

- Is is the ratio of positive instances that are correctly detected by the classifier.
- Also called sensitivity or true positive rate (TPR).

$$\mathtt{Recall} = p(\hat{y} = 1 \mid y = 1) = \frac{\mathtt{TP}}{\mathtt{TP} + \mathtt{FN}}$$



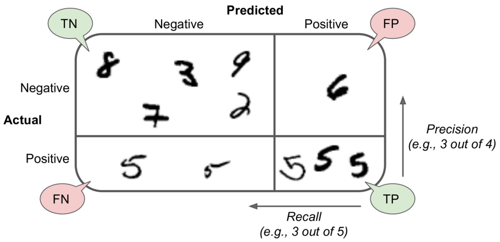
F1 Score

- $F1$ score: combine precision and recall into a single metric.
- The harmonic mean of precision and recall.
- $F1$ only gets high score if both recall and precision are high.

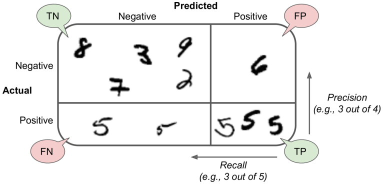
$$\mathtt{F1} = \frac{2}{\frac{1}{\mathtt{precision}} + \frac{1}{\mathtt{recall}}}$$

Confusion Matrix

- The confusion matrix is $K \times K$, where K is the number of classes.



Confusion Matrix - Example



$$\mathtt{TP} = 3, \mathtt{TN} = 5, \mathtt{FP} = 1, \mathtt{FN} = 2$$

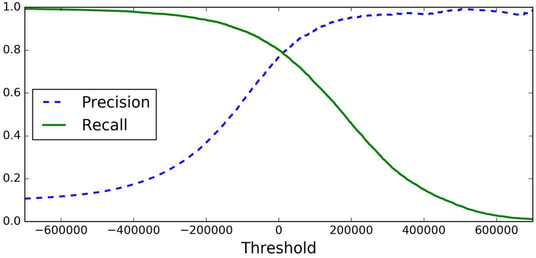
$$\mathtt{Precision} = \frac{\mathtt{TP}}{\mathtt{TP} + \mathtt{FP}} = \frac{3}{3+1} = \frac{3}{4}$$

$$\mathtt{Recall\ (TPR)} = \frac{\mathtt{TP}}{\mathtt{TP} + \mathtt{FN}} = \frac{3}{3+2} = \frac{3}{5}$$

$$\mathtt{FPR} = \frac{\mathtt{FP}}{\mathtt{TN} + \mathtt{FP}} = \frac{1}{5+1} = \frac{5}{6}$$

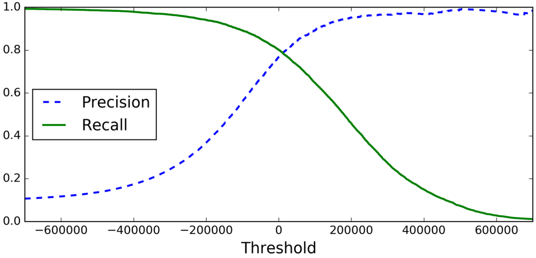
Precision-Recall Tradeoff

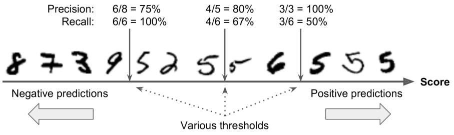
- Precision-recall tradeoff: increasing precision reduces recall, and vice versa.



Precision-Recall Tradeoff

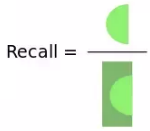
- Precision-recall tradeoff: increasing precision reduces recall, and vice versa.

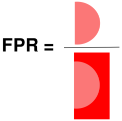




The ROC Curve

- True positive rate (TPR) (recall): $p(\hat{y} = 1 \mid y = 1)$
- False positive rate (FPR): $p(\hat{y} = 1 \mid y = 0)$

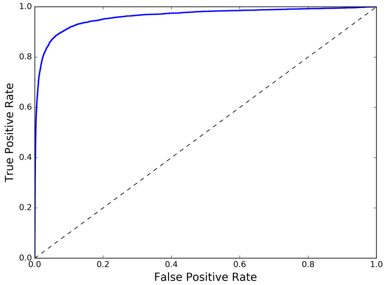




The ROC Curve

- True positive rate (TPR) (recall): $p(\hat{y} = 1 \mid y = 1)$
- False positive rate (FPR): $p(\hat{y} = 1 \mid y = 0)$

- The receiver operating characteristic (ROC) curves summarize the trade-off between the TPR and FPR for a model using different probability thresholds.



Summary

Summary

- Linear regression model $\hat{y} = \mathbf{w}^\top \mathbf{x}$
  - Learning parameters $\mathbf{w}$
  - Cost function $J(\mathbf{w})$
  - Learn parameters: normal equation, gradient descent (batch, stochastic, mini-batch)
- Generalization
  - Overfitting vs. underfitting
  - Bias vs. variance
  - Regularization: Lasso regression, Ridge regression, ElasticNet
- Hyperparameters and cross-validation

Summary

- Binomial logistic regression
  - $y \in \{0, 1\}$
  - Sigmoid function
  - Minimize the cross-entropy
- Multinomial logistic regression
  - $y \in \{1, 2, \cdots, k\}$
  - Softmax function
  - Minimize the cross-entropy
- Performance measurements
  - TP, TF, FP, FN
  - Precision, recall, F1
  - Threshold and ROC

Questions?