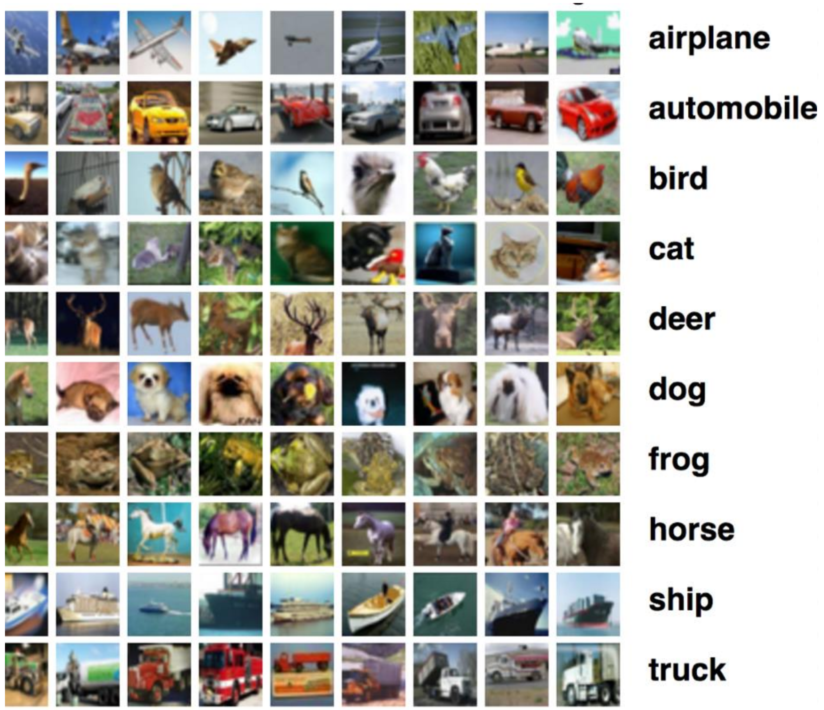
ECE 4252/8803: Fundamentals of Machine Learning (FunML)
Fall 2024

Lecture 3: Classifiers



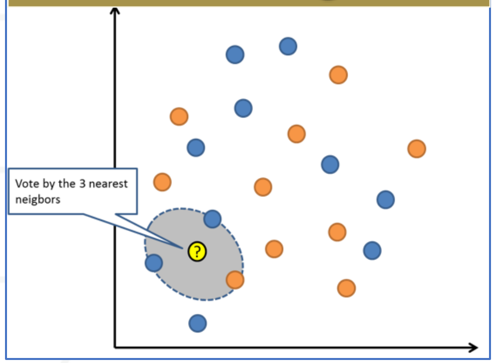
OLIVES @GeorgiaTech

Georgia Tech

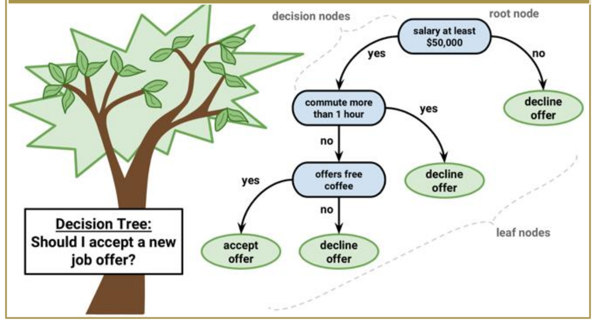
Overview

In This Course..

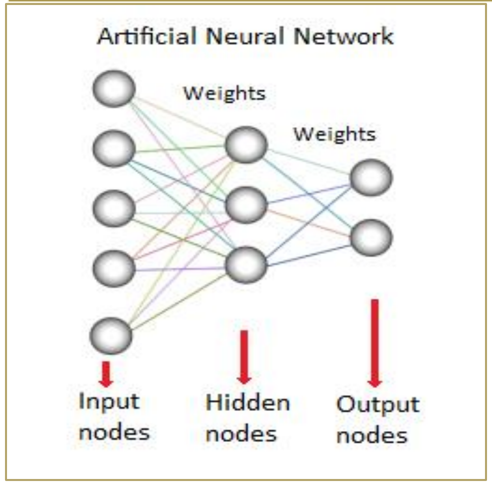
Nearest Neighbor



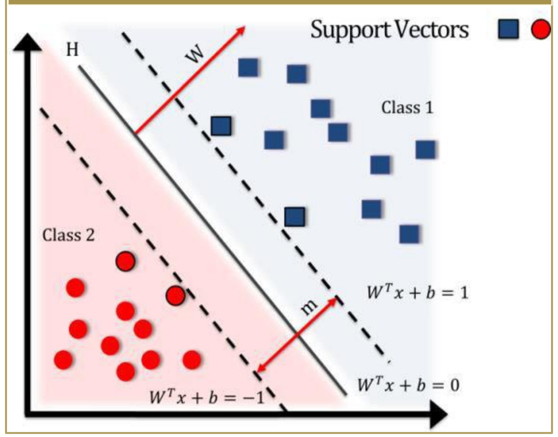
Decision Trees



Artificial Neural Networks



Support Vector Machines



Naive Bayes Classifier

Likelihood

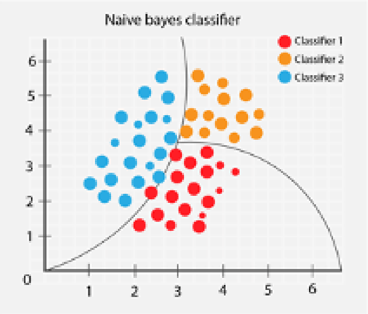
Class Prior Probability

$$P(c \mid x) = \frac{P(x \mid c)P(c)}{P(x)}$$

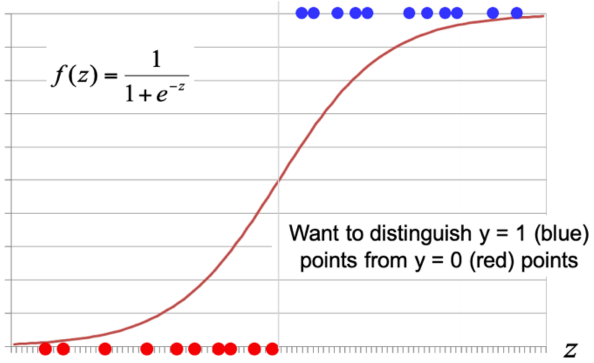
Posterior Probability

Predictor Prior Probability

$$P(c \mid X) = P(x_1 \mid c) \times P(x_2 \mid c) \times \cdots \times P(x_n \mid c) \times P(c)$$



Logistic Regression



Overview

In This Course..

- Nearest Neighbor
- Naïve Bayes
  - Overview
  - Example: a) single feature
  - Example: b) two or more features
- Linear Classifiers
- Logistic Regression
- Decision Trees
- Support Vector Machines
- Artificial Neural Networks

Naïve Bayes
Overview

- Used for both classification and regression
- Assumes that all features in a dataset are:
  - Equally important (no weights), and
  - Independent
- Requires very small computational power, thus works fast even with large data
- Key terms:
  - Prior Probability
  - Likelihood
  - Marginal likelihood

Naïve Bayes

Case Study: Single Feature

- Weather condition and corresponding target variable "Play"
- Objective: classify (predict) whether to play or not based on weather condition
- Steps:
  - Step 1: Convert the data set into a frequency table (also called contingency table)
  - Step 2: Create Likelihood table.
  - Step 3: Use Naive Bayesian equation to calculate the posterior probability for each class.

The class with the highest posterior probability is the outcome of prediction.

Weather	Play
Sunny	No
Overcast	Yes
Rainy	Yes
Sunny	Yes
Sunny	Yes
Overcast	Yes
Rainy	No
Rainy	No
Sunny	Yes
Rainy	Yes
Sunny	No
Overcast	Yes
Overcast	Yes
Rainy	No

Naïve Bayes

Case Study: Single Feature

- Step 1 and Step 2

Data set

Weather	Play
Sunny	No
Overcast	Yes
Rainy	Yes
Sunny	Yes
Sunny	Yes
Overcast	Yes
Rainy	No
Rainy	No
Sunny	Yes
Rainy	Yes
Sunny	No
Overcast	Yes
Overcast	Yes
Rainy	No

Frequency (contingency) Table

	No	Yes	Total
Overcast	0	4	4
Rainy	3	2	5
Sunny	2	3	5
Total	5	9	**14**

$$P(rainy) = \frac{5}{14}$$

$$P(Yes) = \frac{9}{14}$$

$$P(No) = \frac{5}{14}$$

$$P(rainy \mid Yes) = \frac{2}{9}$$

$$P(rainy \mid No) = \frac{3}{5}$$

$$P(Yes \mid rainy) = ?$$

$$P(No \mid rainy) = ?$$

Naïve Bayes

Case Study: Single Feature

	No	Yes	Total
Overcast	0	4	4
Rainy	3	2	5
Sunny	2	3	5
Total	5	9	14

The terms ***Likelihood***, ***Marginal Likelihood***, and ***Prior Probability*** (or ***Class Prior Probability***, as it is related to classes "Yes" or "No") that were mentioned above are shown below:

Likelihood (lime color divided by olive drab color)
Example: P (*Overcast* | "*Yes*") = 4/9

Marginal Likelihood or **Predictor Prior Probabilities** (red color)
Example: P (*Sunny*) = 5/14

Likelihood = *P* (*Feature* | Class)

Weather value	No	Yes	Total
Overcast	0	4/14	4/14
Rainy	3/14	2/14	5/14
Sunny	2/14	3/14	5/14
Total	5/14	9/14	N

Feature value	Class A	Class B	Total
Value A	a/N	d/N	(a+d)/N
Value B	b/N	e/N	(b+e)/N
Value C	c/N	f/N	(c+f)/N
Total	(a+b+c)/N	(d+e+f)/N	N

Prior Probabilities or **Class Prior Probabilities** (olive drab color)
Example: P (*No*) = 5/14

Prior Probabilities = *P (Class)*

Marginal Likelihood = *P (Feature)*

- Step 3:
  - Use Bayes' Formula to calculate the *posterior probability* for each class.
  - The class with the highest posterior probability is the outcome of prediction.

Likelihood
P(feature|class)

Class Prior Probability
P(class)

$$P(y|\boldsymbol{x}) = \frac{P(\boldsymbol{x}|y)P(y)}{P(\boldsymbol{x})}$$

Predictor Prior Probability
P(feature)

Posterior Probability
P(class|feature)

Naïve Bayes

Case Study: Single Feature

- Step 3:
  - Use Bayes' Formula to calculate the *posterior probability* for each class.
  - The class with the highest posterior probability is the outcome of prediction.

Likelihood
P(feature|class)

Class Prior Probability
P(class)

$$P(y|\boldsymbol{x}) = \frac{P(\boldsymbol{x}|y)P(y)}{P(\boldsymbol{x})}$$

Posterior Probability
P(class|feature)

Predictor Prior Probability
P(feature)

$$P(Yes \mid rainy) = \frac{P(rainy \mid Yes)\,P(Yes)}{P(rainy)} = \frac{\frac{2}{9} \times \frac{9}{14}}{\frac{5}{14}} = \frac{2}{5} = 0.4$$

Naïve Bayes

Case Study: Single Feature

- Step 3:
  - Use Bayes' Formula to calculate the *posterior probability* for each class.
  - The class with the highest posterior probability is the outcome of prediction.

Likelihood
P(feature|class)

Class Prior Probability
P(class)

$$P(y|\boldsymbol{x}) = \frac{P(\boldsymbol{x}|y)P(y)}{P(\boldsymbol{x})}$$

Predictor Prior Probability
P(feature)

Posterior Probability
P(class|feature)

$$P(No\,|\,rainy) = \frac{P(rainy\,|\,No)\,P(No)}{P(rainy)} = \frac{\frac{3}{5} \times \frac{5}{14}}{\frac{5}{14}} = \frac{3}{5} = 0.6 = 1 - 0.4$$

OLIVES @GeorgiaTech

Georgia Tech

Naïve Bayes

Case Study: Two or More Feature

- Let's assume we have the additional feature "Wind"
- Let's assume we want to predict the class for the following case:

  Wind = Moderate

  Weather = Sunny
- Thus, we want to estimate

$$P(play = Yes | x_1 = sunny, x_2 = moderate)$$

$$P(play = No | x_1 = sunny, x_2 = moderate)$$

And choose the large value

Weather	Wind	Play
Sunny	Strong	No
Overcast	Weak	Yes
Rainy	Moderate	Yes
Sunny	Weak	Yes
Sunny	Weak	Yes
Overcast	Weak	Yes
Rainy	Strong	No
Rainy	Strong	No
Sunny	Weak	Yes
Rainy	Weak	Yes
Sunny	Strong	No
Overcast	Weak	Yes
Overcast	Weak	Yes
Rainy	Moderate	No

	No	Yes	Total
Overcast	0	4	4
Rainy	3	2	5
Sunny	2	3	5
Total	5	9	14

	No	Yes	Tota l
Strong	4	0	4
Weak	0	8	8
Moderate	1	1	2
Total	5	9	14

Naïve Bayes

Case Study: Two or More Feature

$$P(yes|sunny, moderate) = \frac{P(sunny, moderate|Yes)P(Yes)}{P(sunny, moderate)}$$

Here is the "naive" in Naïve Bayes:

- It assumes that the features are conditionally independent
- If x_1, x_2 are conditionally independent, then:

$$P(x_1, x_2|y) = P(x_1|y)P(x_2|y)$$

Hence,

$$P(yes|sunny, moderate) = \frac{P(sunny|Yes)P(moderate|Yes)P(Yes)}{P(sunny, moderate)}$$

And

$$P(No|sunny, moderate) = \frac{P(sunny|Np)P(moderate|No)P(No)}{P(sunny, moderate)}$$

Weather	Wind	Play
Sunny	Strong	No
Overcast	Weak	Yes
Rainy	Moderate	Yes
Sunny	Weak	Yes
Sunny	Weak	Yes
Overcast	Weak	Yes
Rainy	Strong	No
Rainy	Strong	No
Sunny	Weak	Yes
Rainy	Weak	Yes
Sunny	Strong	No
Overcast	Weak	Yes
Overcast	Weak	Yes
Rainy	Moderate	No

Naïve Bayes

Case Study: Two or More Feature

$$P(yes|sunny, moderate) \propto P(sunny|yes)P(moderate|yes)P(yes)$$

And

$$P(No|sunny, moderate) \propto P(sunny|No)P(moderate|No)P(No)$$

$$P(Yes|sunny, moderate) = \frac{\frac{3}{9} \times \frac{1}{9} \times \frac{9}{14}}{constant} = \frac{\frac{1}{42}}{constant}$$

$$P(No|sunny, moderate) = \frac{\frac{2}{5} \times \frac{1}{5} \times \frac{5}{14}}{constant} = \frac{\frac{1}{35}}{constant}$$

Therefore, the Naïve Bayes classifier will result in **No**

Weather	Wind	Play
Sunny	Strong	No
Overcast	Weak	Yes
Rainy	Moderate	Yes
Sunny	Weak	Yes
Sunny	Weak	Yes
Overcast	Weak	Yes
Rainy	Strong	No
Rainy	Strong	No
Sunny	Weak	Yes
Rainy	Weak	Yes
Sunny	Strong	No
Overcast	Weak	Yes
Overcast	Weak	Yes
Rainy	Moderate	No

	No	Yes	Total
Overcast	0	4	4
Rainy	3	2	5
Sunny	2	3	5
Total	5	9	**14**

	No	Yes	Tota l
Strong	4	0	4
Weak	0	8	8
Moderate	1	1	2
Total	5	9	14

Naïve Bayes

Case Study: Two or More Feature

- Using conditional independence of feature vector $\boldsymbol{x} = \left[x_1, x_2, \dots, x_p\right]^T$ given a class labels vector $y = [y_1, y_1, \dots, y_k]^T$, the posterior probability of a class y_k given the feature vector $\boldsymbol{x}$ is:

$$P(y_k|\boldsymbol{x}) = \frac{P(x_1|y_k)P(x_2|y_k) \dots P(x_p|y_k)P(y_k)}{P(\boldsymbol{x})}$$

$$P(y_k|\boldsymbol{x}) \propto P(x_1|y_k)P(x_2|y_k) \dots P(x_p|y_k)P(y_k)$$

Then, the class can be found as:

$$class = \arg\max_i P(y_i|\boldsymbol{x})$$

Naïve Bayes

Types of Classifiers

Different assumptions about $P(\boldsymbol{x}_{:,i}|y_k)$

- Categorical
  - $P\left(\boldsymbol{x}_{:,i} = a \middle| y_k = b\right) = \frac{N_{a,b}+\alpha}{N_b+\alpha n_i}$
  - where
    - $N_{a,b}$: Number of samples with $\boldsymbol{x}_{:,i} = a$ and $y_k = b$
    - N_b: Number of samples with $y_k = b$
    - n_i: The number of categories in $\boldsymbol{x}_{:,i}$
- Gaussian:
  - $P\left(\boldsymbol{x}_{:,i} \middle| y_k\right) = \frac{1}{\sqrt{2\pi\sigma_{y_k}^2}} \exp\left(-\frac{\left(\boldsymbol{x}_{:,i}-\mu_{y_k}\right)^2}{2\sigma_{y_k}^2}\right)$
- Bernoulli
- Multinomial
- Complement
- More details can be found here: https://scikit-learn.org/stable/modules/naive_bayes.html#naive-bayes

Naïve Bayes
Pros and Cons

- **Pros:**
  - Fast
  - Performance (assuming independence)
  - Better at categorical input
- **Cons:**
  - Zero-frequency
  - Assumption of independence

Naïve Bayes
Common Applications

- **Real time Prediction:** Naive Bayes is an eager learning classifier and it is fast. Thus, it could be used for making predictions in real time.
- **Multi-class Prediction:** This algorithm is also well known for multi-class prediction feature. Here we can predict the probability of multiple classes of target variable.
- **Text classification/ Spam Filtering/ Sentiment Analysis:** Naive Bayes classifiers mostly used in text classification (due to better result in multi class problems and independence rule) have higher success rate as compared to other algorithms. As a result, it is widely used in Spam filtering (identify spam e-mail) and Sentiment Analysis (in social media analysis, to identify positive and negative customer sentiments)
- **Recommendation System:** Naive Bayes Classifier and Collaborative Filtering together builds a Recommendation System that uses machine learning and data mining techniques to filter unseen information and predict whether a user would like a given resource or not

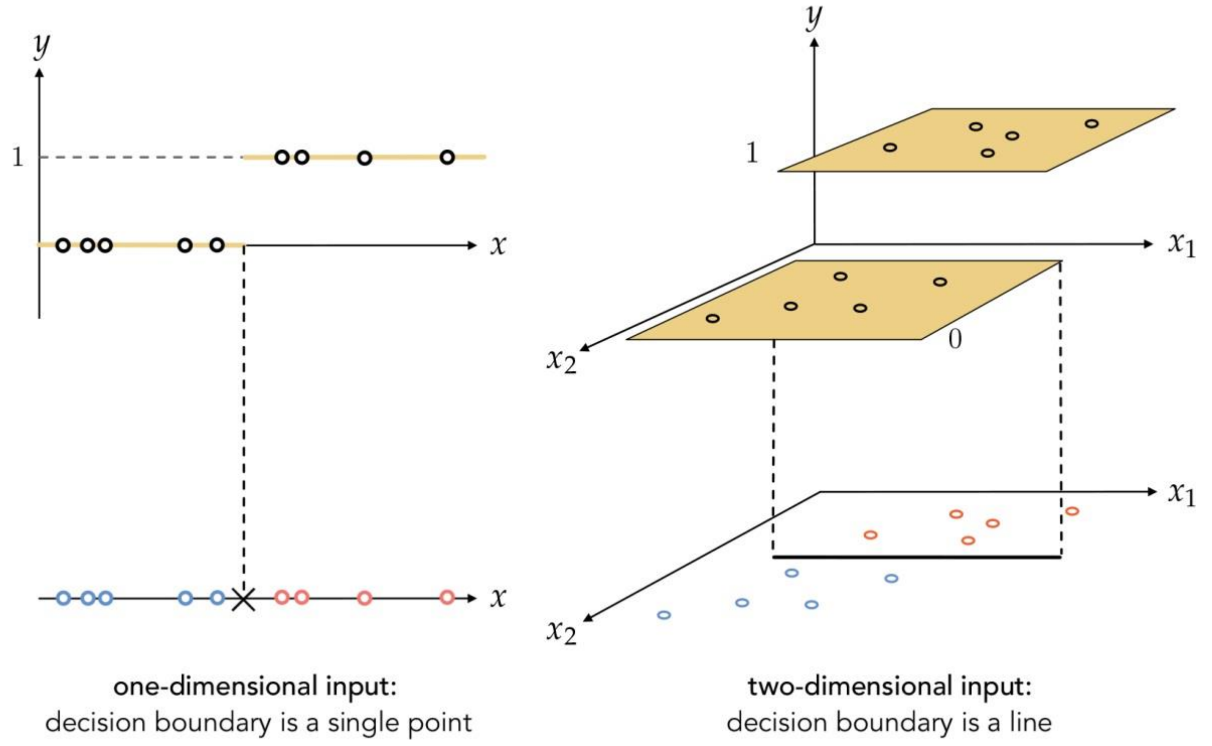
Overview

In This Course..

- Nearest Neighbor
- Naïve Bayes
- Linear Classifiers
  - Overview
  - Example
  - Terminologies
- Logistic Regression
- Decision Trees
- Support Vector Machines
- Artificial Neural Networks

Binary Classifiers
Toy Example

- Considering a binary dataset of N samples $(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2) \dots (\boldsymbol{x}_N, y_N)$. Label $y_i \in \{0, 1\}$. The simplest shape such a dataset can take is a set of ***linearly separated*** adjacent 'steps' as following:



Materials adapted from Machine Learning Refined Book (2nd edition)

Linear Classifiers

Terminologies

- Parameters: Variables used to represent a probability distribution. When the number of variables required to represent a probability distribution (or solve some task like classification) are finite (within some error), then the collection of such variables is termed a parametric model
- Loss function: Given a single sample, loss function measures the error between the true data prediction (label) and parametric model's prediction
- Empirical Risk: Given a training dataset, cost function measures the error between the true data predictions (labels) and parametric model's predictions
- Cost function: Given a training dataset, cost function measures the error between the true data predictions (labels) and parametric model's predictions along with some penalty to account for generalizability (regularization)
- Objective function: An overall goal in ML that drives the learning algorithm. For instance, Maximum Likelihood Estimate on classification can act as a proxy for clustering algorithms

Linear Classifiers
Terminologies

- Data: Data is present (exists) as a joint probability distribution $\mathrm{p}(x, y)$
- Training Data: Some data is sampled from the true underlying distribution $p(x_{train}, y_{train})$
- Empirical risk/loss/cost minimization: A parametric model with fixed set of parameters is learned to obtain $f_\theta\colon \boldsymbol{X} \rightarrow \boldsymbol{y}$, by minimizing some defined cost function over the training data
- The cost function is defined based on some objective function
- The cost function is a collection of loss functions across individual samples within $p(x_{train}, y_{train})$
- The cost function is a regularized version of empirical risk to ensure that the model $f_\theta\colon \boldsymbol{X} \rightarrow \boldsymbol{y}$, is able to predict on non-training data at inference

Linear Classifiers
Parametric Modeling

- Parametric models fit the data with a fixed set of parameters θ, which form a mapping function $f_\theta: \boldsymbol{X} \rightarrow \boldsymbol{y}$. Examples of parametric models include:
  - Logistic Regression
  - Perceptron
  - Linear SVMs
  - Artificial Neural Networks
  - Naive Bayes

Question: How is Naïve Bayes parametric?

- When a decision is made (Ex: **p(y|x) > 0.5**), you are inherently creating a line
- Frequency table is usually not provided to us, or easily computable. Instead, we make assumptions based on some **data generation properties**. For instance, we say data was generated using a gaussian distribution and the data around a mean is more ***likely*** or more ***frequent.*** More in HW 2

Linear Classifiers

Generative vs Discriminative Modeling

- Goal: learn mapping $f_\theta: \boldsymbol{X} \rightarrow \boldsymbol{y}$
- Discriminative classifiers, e.g., logistic regression, SVMs:
  - model the posterior $p(y|\boldsymbol{x})$ directly or learn a direct mapping $f: \boldsymbol{x} \rightarrow y$
- **Generative classifiers**, e.g., naïve bayes:
  - estimate $p_{\boldsymbol{x}|\boldsymbol{y}}(x|y)$, $p(y)$from data
  - Indirectly estimate the posterior using **Bayes rules**

- A Discriminative model models the **decision boundary between the classes**.
- A Generative Model explicitly models the **actual distribution of each class**.
- Both predict the conditional probability P(Class | Features). However, both models learn different probabilities.
- A Generative Model learns the **joint probability distribution p(x,y).** It predicts the conditional probability with the help of **Bayes Theorem**. A Discriminative model learns the **conditional probability distribution p(y|x)**. Both models were generally used in **supervised learning** problems. Recently, generative modeling is used in unsupervised/self-supervised modeling

Generative classifiers:

•Naïve Bayes

•Bayesian networks

•Markov random fields

•Hidden Markov Models (HMM)

Discriminative Classifiers:

•Logistic regression

•Scalar Vector Machine

•Traditional neural networks

•Nearest neighbour

•Conditional Random Fields (CRF)s

Goal: Training classifiers involve estimating f: X -> Y, or P(Y|X)

Generative classifiers

•Assume some functional form for **P(Y), P(X|Y)**

•Estimate parameters of **P(X|Y), P(Y)** directly from training data

•Use Bayes rule to calculate **P(Y |X)**

Discriminative Classifiers

•Assume some functional form for **P(Y|X)**

•Estimate parameters of **P(Y|X)** directly from training data

Overview

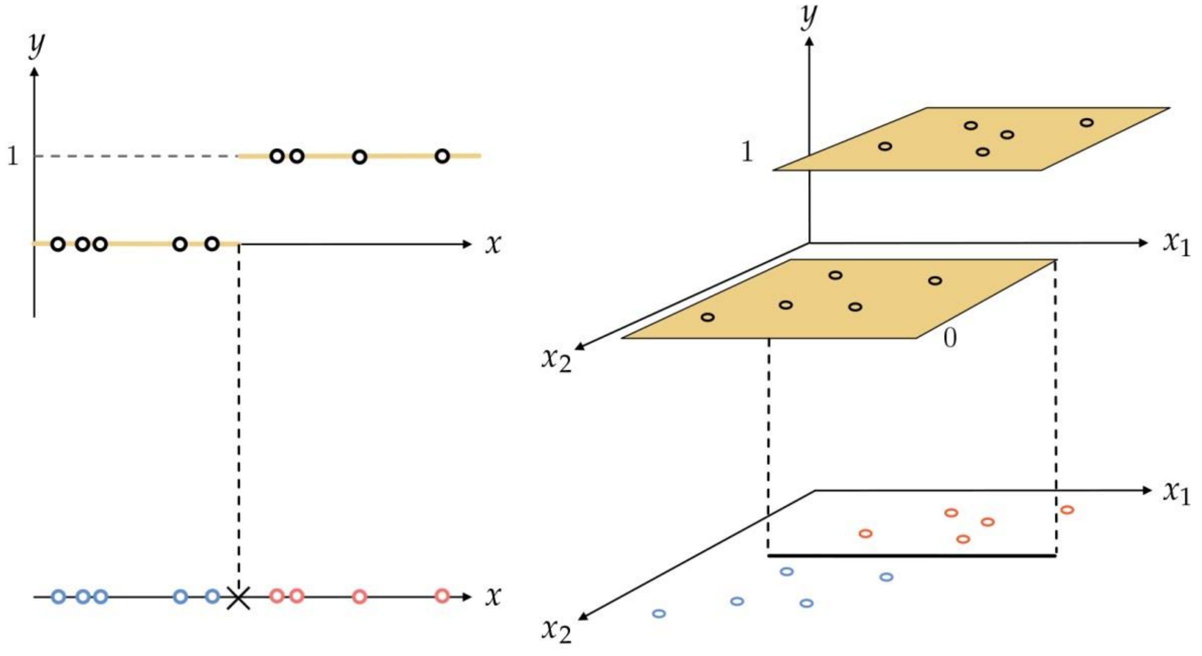
In This Course..

- Nearest Neighbor
- Naïve Bayes
- Logistic Regression
  - Overview
  - Cross Entropy
  - Gradient Descent
  - Normalization
  - Regularization
  - Example
  - Connection to Naïve Bayes
  - Hinge Cost – softplus
- Decision Trees
- Support Vector Machines
- Artificial Neural Networks

Logistic Regression
Overview

- A probabilistic classifier that directly estimates posterior $P(y|\boldsymbol{x})$
- Models probability for two-class classification problem
- Models probability with *sigmoid* function
- Key terms:
    - Sigmoid function
    - Weights
    - Bias

- Considering a binary dataset of N samples $(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2) \dots (\boldsymbol{x}_N, y_N)$. Label $y_i \in \{0, 1\}$. The simplest shape such a dataset can take is a set of ***linearly separated*** adjacent 'steps' as following:



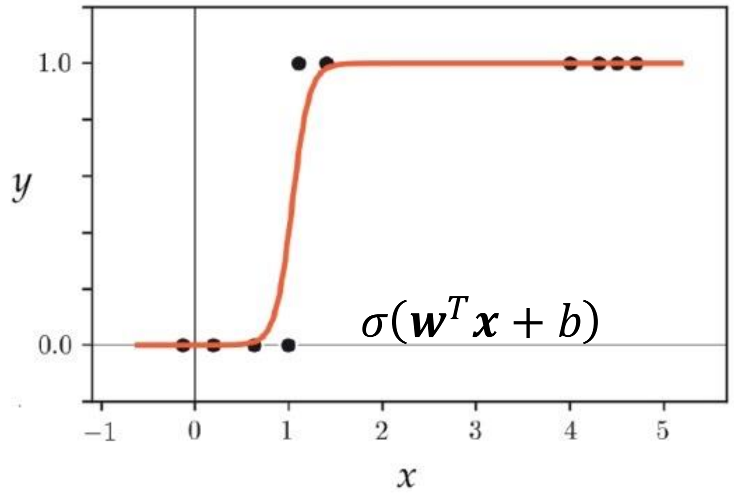
one-dimensional input:
decision boundary is a single point

two-dimensional input:
decision boundary is a line

Materials adapted from Machine Learning Refined Book (2nd edition)

Logistic Regression
Sigmoid

- The sigmoid function, $\sigma(\boldsymbol{x}) = \frac{1}{1+e^{-x}}$, fits such data and gives a set of *approximate* equalities: $\sigma(\boldsymbol{w}^T\boldsymbol{x} + b) \approx y$

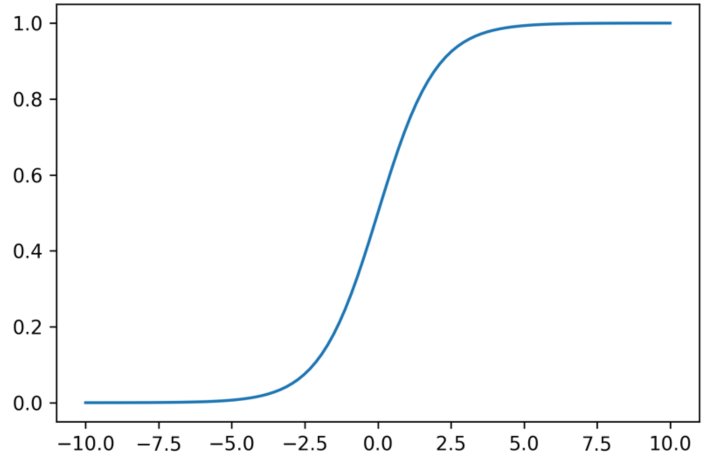


Materials adapted from Machine Learning Refined Book (2nd edition)

Logistic Regression

Key Terms

- Sigmoid function: $\sigma(\boldsymbol{x}) = \frac{1}{1+e^{-x}}$
- Given feature vector $\boldsymbol{x} = [x_1, x_2, \dots, x_p]^T$, logistic regression classifier models the probability with sigmoid function:
$$P(y = 1|\boldsymbol{x}) = \sigma(\boldsymbol{w}^T\boldsymbol{x} + b)$$
- Where $\boldsymbol{w} = [w_1, w_2, \dots, w_p]^T$ is weight, vector b is bias
- Given that $P(y = 1|\boldsymbol{x}) + P(y = 0|\boldsymbol{x}) = 1$:
    - $P(y = 1|\boldsymbol{x}) = \sigma(\boldsymbol{w}^T\boldsymbol{x} + b)$
    - $P(y = 0|\boldsymbol{x}) = 1 - \sigma(\boldsymbol{w}^T\boldsymbol{x} + b)$
- Classification rule:
    - $y = \begin{cases} 1 & P(y = 1|\boldsymbol{x}) > 0.5 \\ 0 & otherwise \end{cases}$



Sigmoid function $\sigma(\boldsymbol{x}) = \frac{1}{1+e^{-x}}$

Logistic Regression
Cost Function

The predicted probabilities of logistic regression classifier:

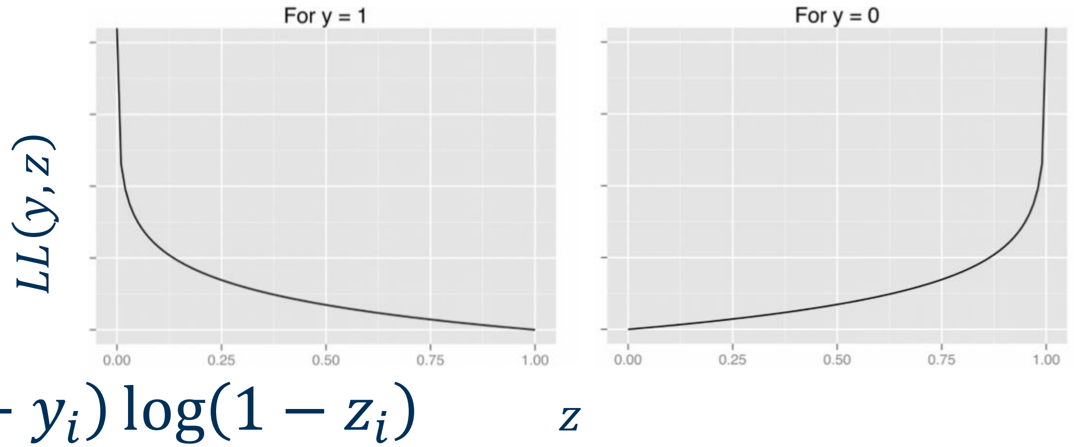
- $P(y=1|\boldsymbol{x}) = \sigma(\boldsymbol{w}^T\boldsymbol{x}+b)$
- $P(y=0|\boldsymbol{x}) = 1-\sigma(\boldsymbol{w}^T\boldsymbol{x}+b)$

Let $z=\sigma(\boldsymbol{w}^T\boldsymbol{x}+b)$, we want the cost function $L(y,z)$ to have the property:

for $y=1 \rightarrow LL(y,z) = \begin{cases} 0, & z=y \\ \infty, & z\rightarrow 0 \end{cases}$, for $y=0 \rightarrow LL(y,z) = \begin{cases} 0, & z=y \\ \infty, & z\rightarrow 1 \end{cases}$

assign more penalty when predicting 1 while the label is 0

$LL(y,z) = \begin{cases} -\log(z), & y=1 \\ -\log(1-z), & y=0 \end{cases}$ satisfies the property



The above could be rewritten as:

$$LL(\boldsymbol{y},\boldsymbol{z}) = -\frac{1}{N}\sum_{i=1}^{N} y_i\log(z_i) + (1-y_i)\log(1-z_i)$$

N: number of samples, y_i: desired output ,z_i: predicted output

OLIVES @GeorgiaTech
Georgia Tech

Sigmoid with Cross-Entropy Loss

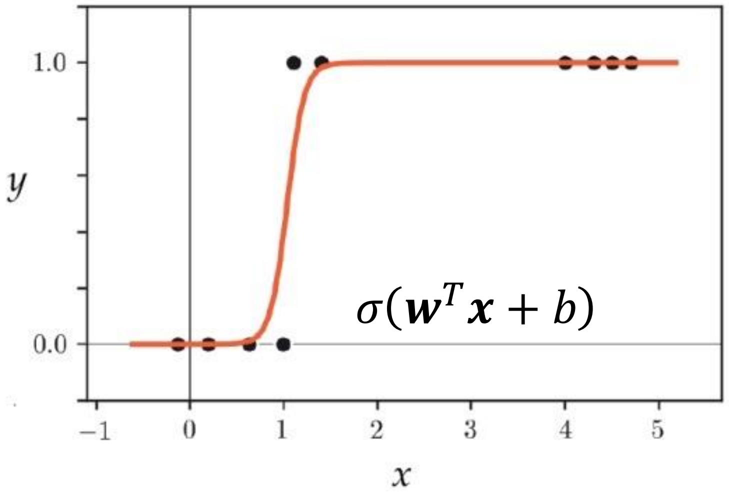
- The non-convex least squares cost is universally defined, regardless of the values taken by the label y.
- However, we can employ a **convex** cost function without local minima as y can be modeled as a binary random variable drawn from a *Bernoulli distribution*, conditioned on the data $\boldsymbol{x}$.
- Recall the PMF of *Bernoulli distribution*: $f(k;p) = p^k(1-p)^{1-k}$
- The conditional probability $P(y|\boldsymbol{x}) = (\sigma(\boldsymbol{x}))^y\big(1-\sigma(\boldsymbol{x})\big)^{1-y}$, taking the log probability:

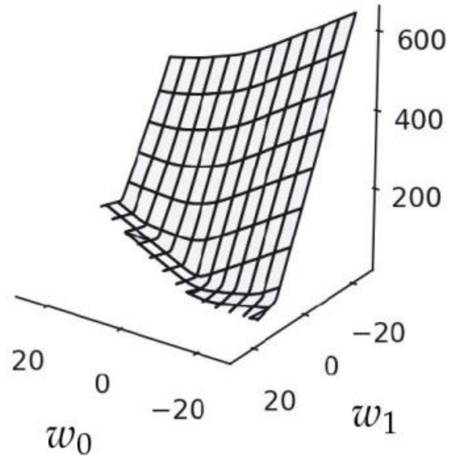
$$\log P(y|\boldsymbol{x}) = y\log(\sigma(\boldsymbol{x})) + (1-y)\log(1-\sigma(\boldsymbol{x}))$$

where $\sigma(\boldsymbol{x}) = \sigma(\boldsymbol{w}^T\boldsymbol{x} + b)$

- We want to maximize such log probability $\log P(y|\boldsymbol{x})$ over all possible configurations of $\boldsymbol{\theta} = \{\boldsymbol{w}, b\}$ using Maximize Likelihood Estimation. Equivalently, we **minimize** the **negative log-likelihood** $-\log P(y|\boldsymbol{x})$:

$$-\log P(y|\boldsymbol{x}) = -y\log\big(\sigma(\boldsymbol{x})\big) - (1-y)\log(1-\sigma(\boldsymbol{x}))$$

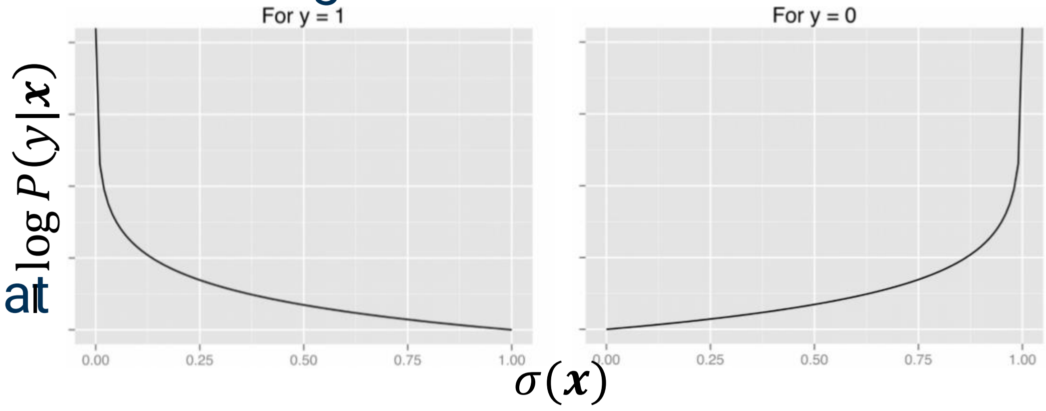




Convex negative log-likelihood

Negative log-likelihood $-\log P(y|\boldsymbol{x})$ can be re-formulated as following:

$$-\log P(y|\boldsymbol{x}) = \begin{cases} -\log(\sigma(\boldsymbol{x})), & if\ y = 1 \\ -\log\big(1 - \sigma(\boldsymbol{x})\big), & if\ y = 0 \end{cases}$$



The cost is *nonnegative* and takes on a minimum value at

The overall negative log-likelihood cost function:

$$LL(\boldsymbol{x}, y) = -\frac{1}{N}\sum_{i=1}^{N} y_i \log\big(\sigma(\boldsymbol{x}_i)\big) + (1 - y_i)\log\big(1 - \sigma(\boldsymbol{x}_i)\big)$$

N: number of samples, y_i: desired output ,z_i: predicted output

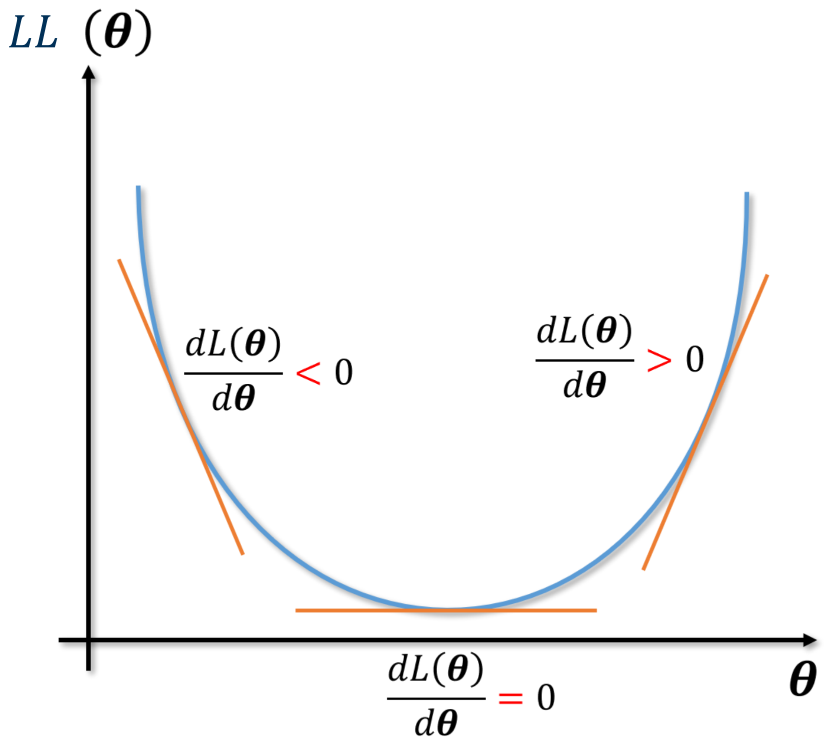
$LL(\boldsymbol{x}, y)$is referred to as the **Cross Entropy Cost** for logistic regression. In binary classification, this is also referred to as the **Binary Cross Entropy Cost**

- We want to find the **optimum** $\boldsymbol{\theta}^*$ such that:

$$\boldsymbol{\theta}^* = \underset{\boldsymbol{\theta}}{\operatorname{argmin}}\, LL(\boldsymbol{x}, y, \boldsymbol{\theta})$$

where $\boldsymbol{\theta} = \{\boldsymbol{w}, b\}$

- Considering the change in the loss function with respect to $\boldsymbol{\theta}$, the loss function is minimum when $\frac{dLL(\boldsymbol{x},y,\boldsymbol{\theta})}{d\boldsymbol{\theta}} = 0$.
- Finding $\boldsymbol{\theta}^*$ can be achieved by gradient descent



Logistic Regression

Review of Derivatives of NLL

Let $\boldsymbol{x} = [1, x_1, .., x_P]^{\boldsymbol{T}}, \boldsymbol{\theta} = [b, w_1, .., w_P]^{\boldsymbol{T}}$. Thus, $\boldsymbol{\theta^T x} = \boldsymbol{w^T x} + b$

Let $h = \boldsymbol{\theta^T x}$, logistic regression: $z = \sigma(h) = \frac{1}{1+e^{-h}}$

$$LL(\boldsymbol{x}, y, \boldsymbol{\theta}) = -\frac{1}{N}\sum_{i=1}^{N} y_i \log\big(\sigma(\boldsymbol{x}_i)\big) + (1 - y_i)\log\big(1 - \sigma(\boldsymbol{x_i})\big)$$

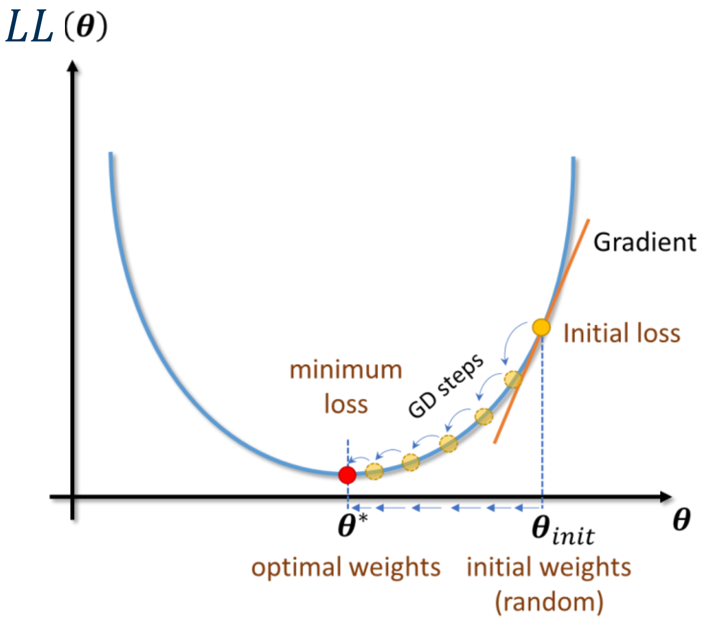
Using chain rule: $\frac{\partial LL(\boldsymbol{x_i}, y_i, \boldsymbol{\theta})}{\partial \boldsymbol{\theta}} = \frac{\partial LL(\boldsymbol{x_i}, y_i, \boldsymbol{\theta})}{\partial z_i} \frac{\partial z_i}{\partial h_i} \frac{\partial h_i}{\partial \boldsymbol{\theta}}$

$$\frac{\partial LL(\boldsymbol{x_i}, y_i, \boldsymbol{\theta})}{\partial z_i} = -\left(\frac{y_i}{z_i} - \frac{1 - y_i}{1 - z_i}\right) = \frac{z_i - y_i}{z_i(1 - z_i)}$$

Recall the derivatives of sigmoid function:

$$\frac{\partial z_i}{\partial h_i} = z_i(1 - z_i)$$

$$\frac{\partial h_i}{\partial \boldsymbol{\theta}} = \boldsymbol{x_i}$$



Thus $\frac{\partial LL(\boldsymbol{x_i}, y_i, \boldsymbol{\theta})}{\partial \boldsymbol{\theta}} = \frac{1}{N}\sum_{i=1}^{N} \frac{z_i - y_i}{z_i(1 - z_i)} z_i(1 - z_i)\boldsymbol{x_i} = -\frac{1}{N}\sum_{i=1}^{N} \Big(y_i - \sigma\big(\boldsymbol{\theta^T x_i}\big)\Big)\boldsymbol{x_i}$

Logistic Regression

Optimum Parameters by Gradient Descent

- The goal of Gradient Descent is to minimize the cost function $LL(\boldsymbol{x}, y, \boldsymbol{\theta})$on a set of parameters $\boldsymbol{\theta} = \{\boldsymbol{W}, \boldsymbol{b}\}$
- The basic idea is to compute the gradient (derivative) of the loss function in a sequence of epochs (intake of the dataset) and update the weights and bias **in proportion to the gradient** according to the update rule:

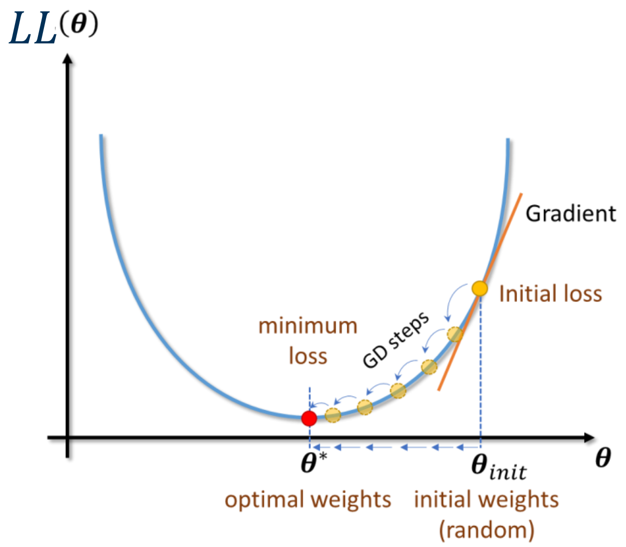
$$\boldsymbol{\theta}(t+1) = \boldsymbol{\theta}(t) - \alpha \frac{\partial LL(\boldsymbol{x}, y, \boldsymbol{\theta})}{\partial \boldsymbol{\theta}}$$

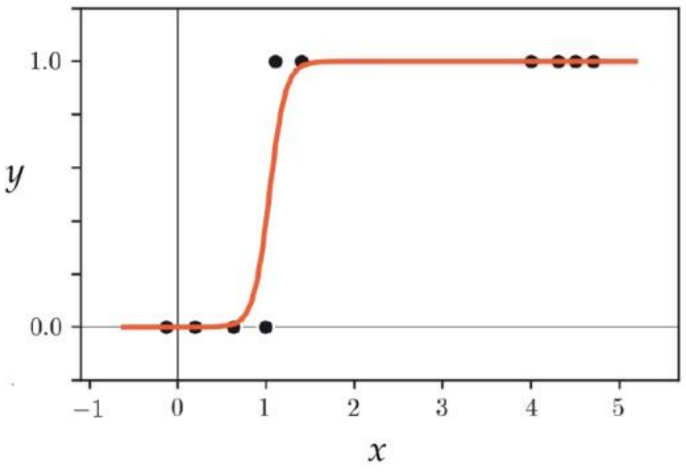
$$\boldsymbol{W}(t+1) = \boldsymbol{W}(t) - \alpha \frac{\partial\, LL(\boldsymbol{\theta})}{\partial \boldsymbol{W}}$$

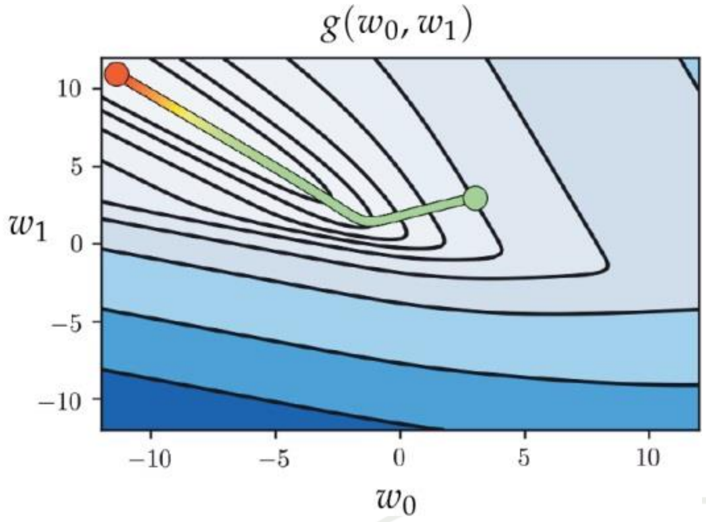
$$\boldsymbol{b}(\mathrm{t}+1) = \boldsymbol{b}(\mathrm{t}) - \alpha \frac{\partial LL(\boldsymbol{\theta})}{\partial \boldsymbol{b}}$$

where $\frac{\partial LL(\boldsymbol{x}_i, y_i, \boldsymbol{\theta})}{\partial \boldsymbol{\theta}} = -\frac{1}{N}\sum_{i=1}^{N}\left(y_i - \sigma\left(\boldsymbol{\theta}^T \boldsymbol{x_i}\right)\right)\boldsymbol{x_i}$

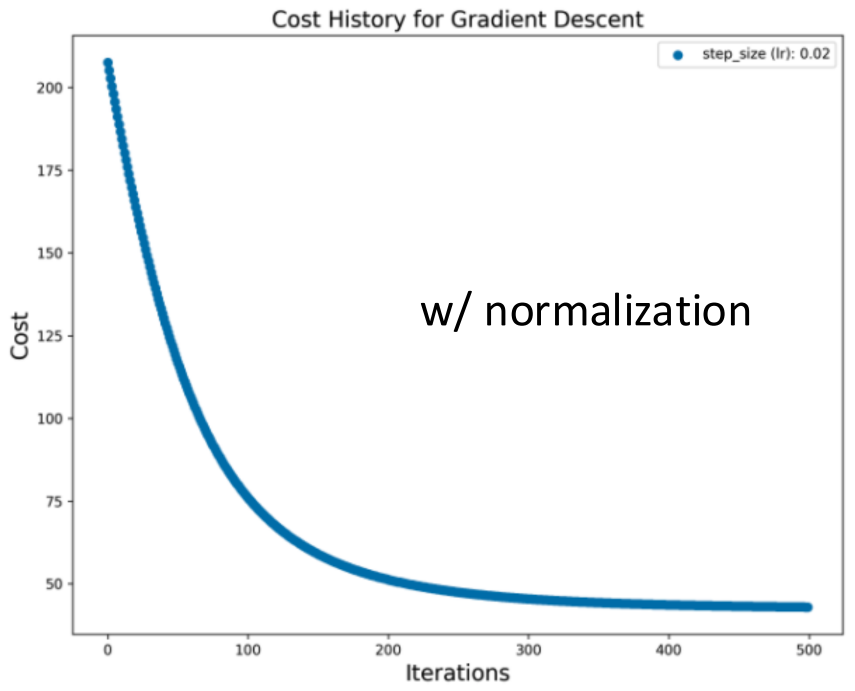
$t+1$ and t indicate the new and current epochs, respectively, and α is a learning rate (step size)

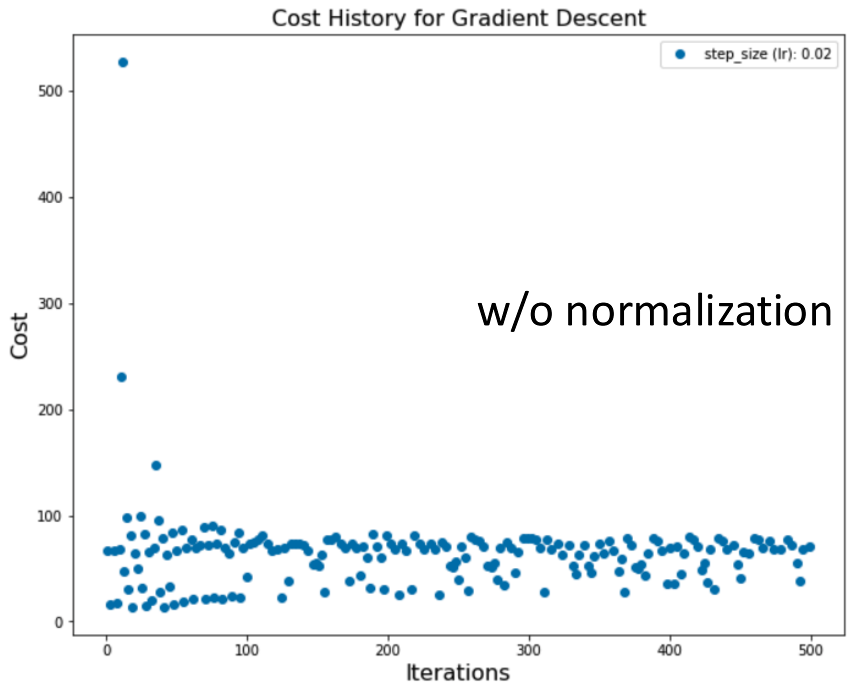






- To ensure that the gradient descent moves the cost smoothly towards the minima, the weights should be updated at the same rate by **normalizing the features** before training the model.
- One common normalization method is min-max normalization as $\boldsymbol{x}_{norm} = \frac{\boldsymbol{x} - min(\boldsymbol{x})}{max(\boldsymbol{x}) - min(\boldsymbol{x})}$





Logistic Regression
Regularization

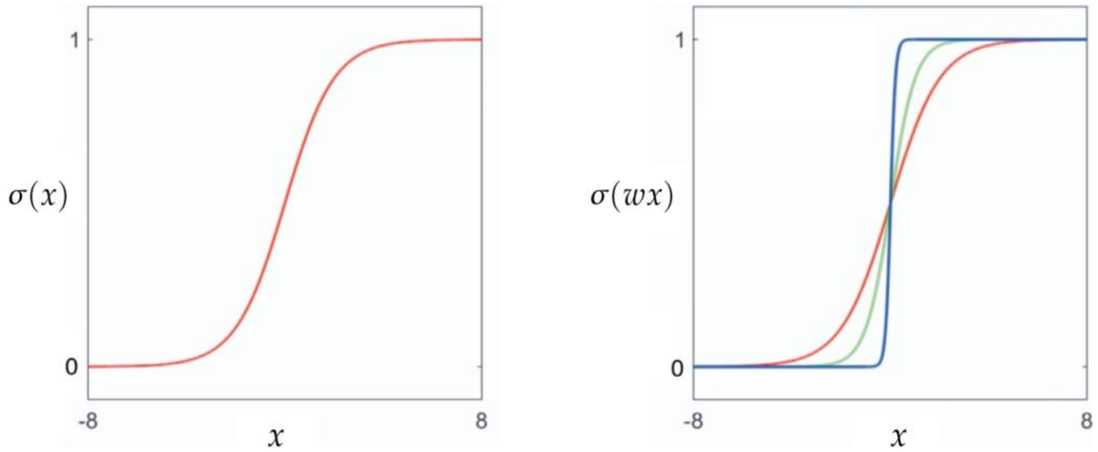
- Recall $\boldsymbol{\theta}(t+1) = \boldsymbol{\theta}(t) - \alpha \frac{\partial LL(\boldsymbol{x},y,\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}$, where $\frac{\partial LL(\boldsymbol{x}_i,y_i,\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} = -\frac{1}{N}\sum_{i=1}^{N}\left(y_i - \sigma(\boldsymbol{\theta}^T \boldsymbol{x}_i)\right)\boldsymbol{x}_i$
- To find the best $\sigma(\boldsymbol{\theta}^T \boldsymbol{x}_i) = \frac{1}{1+e^{-\theta^T x_i}}$, there could be an undesirable outcome as following:
  - For y_i = 0, the best $\sigma(\boldsymbol{\theta}^T \boldsymbol{x}_i)$ will try to approximate as close 0 as possible. As such, $\boldsymbol{\theta} \to -\infty$
  - For y_i = 1, the best $\sigma(\boldsymbol{\theta}^T \boldsymbol{x}_i)$ will try to approximate as close 1 as possible . As such, $\boldsymbol{\theta} \to +\infty$
- This leads to overfitting issues. The model will not be able to generalize well to unseen samples.
- Regularization is a technique to solve the problem of overfitting by penalizing the cost function.
  One common regularization is L_2 norm regularization:
  - $L_{reg}(\boldsymbol{x}, y, \boldsymbol{\theta}) = LL(\boldsymbol{x}, y, \boldsymbol{\theta}) + \frac{\boldsymbol{\lambda}}{\boldsymbol{2N}}\|\boldsymbol{\theta}\|_{\boldsymbol{2}}$
  - $\frac{\partial L_{reg}(\boldsymbol{x}_i,y_i,\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} = -\frac{1}{N}\sum_{i=1}^{N}\left(y_i - \sigma(\boldsymbol{\theta}^T \boldsymbol{x}_i)\right)\boldsymbol{x}_i + \frac{\lambda}{N}\boldsymbol{\theta}$
- $\boldsymbol{\lambda}$ is the regularization hyper-parameter. It controls the trade-off between fitting the training data well versus penalizing the parameters to avoid overfitting.

Logistic Regression
Summary

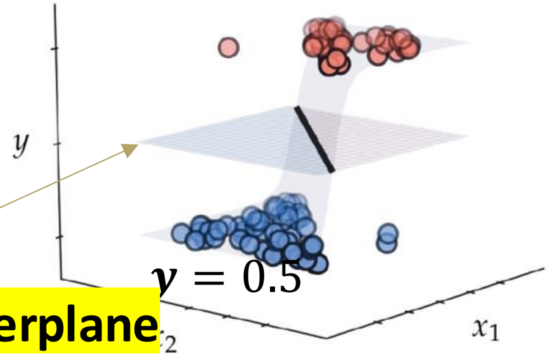
- The modeling process we just went through is called logistic regression
- Logistic Regression is a **parametric model**
- Given feature vector $\boldsymbol{x} = [x_1, \dots, x_p]^T$, logistic regression classifier estimates the **posterior** $P(y|\boldsymbol{x})$ with *sigmoid function* $\sigma(\boldsymbol{x})$:

$$P(y = 1|\boldsymbol{x}) = \sigma(\boldsymbol{w}^T\boldsymbol{x} + b)$$
$$P(y = 0|\boldsymbol{x}) = 1 - \sigma(\boldsymbol{w}^T\boldsymbol{x} + b)$$

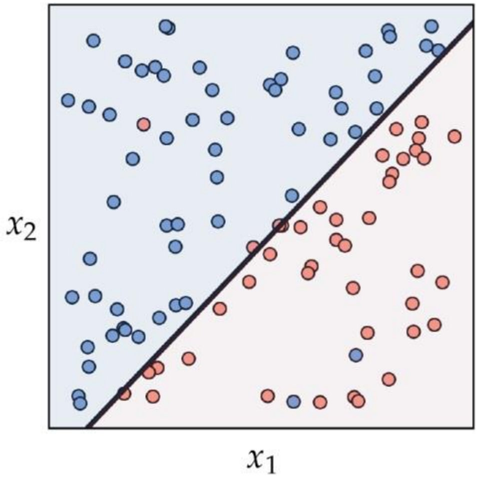
- Where $\boldsymbol{w} = [w_1, \dots, w_p]^T$ is **weight**, vector b is **bias**
- Classification rule:
  - $y = \begin{cases} 1 & P(y = 1|\boldsymbol{x}) > 0.5 \\ 0 & otherwise \end{cases}$



Sigmoid function $\sigma(\boldsymbol{x}) = \dfrac{1}{1 + e^{-\boldsymbol{x}}}$



Linear separation hyperplane



Logistic Regression

Example: Sentiment Classification

Let's assume we want to predict the binary sentiment for the following movie review:

Input: "It's hokey. There are virtually no surprises, and the writing is second-rate. So why was it so enjoyable? For one thing, the cast is great. Another nice touch is the music. I was overcome with the urge to get off the couch and start dancing. It sucked me in, and it'll do the same to you."

Output: positive (1) or negative (0)

Materials adapted from Prof. Dan Jurafsky at Stanford

Logistic Regression

Example: Sentiment Classification

Input: "It's hokey. There are virtually no surprises , and the writing is second-rate. So why was it so enjoyable? For one thing, the cast is great. Another nice touch is the music. I was overcome with the urge to get off the couch and start dancing. It sucked me in, and it'll do the same to you."

Output: positive (1) or negative (0)

Let's assume the sentiment is represented by 6 features

$$\boldsymbol{x} = [x_1, \dots x_6]$$

Thus, we want to estimate

$$P(semtiment = 1|\boldsymbol{x})$$

$$P(sentiment = 0|\boldsymbol{x})$$

feature	description	value
x_1	Count positive words	
x_2	Count negative words	
x_3	$\begin{cases} 1 & if \text{ "no" } in\ text \\ 0 & otherwise \end{cases}$	
x_4	Count 1^{st} and 2^{nd} pronouns	
x_5	$\begin{cases} 1 & if \text{ "!" } in\ text \\ 0 & otherwise \end{cases}$	
x_6	$\log(word\ count)$	

Materials adapted from Prof. Dan Jurafsky at Stanford

Logistic Regression

Example: Sentiment Classification

$x_3 = 1$ $x_2 = 2$

"It's hokey. There are virtually no surprises, and the writing is second-rate. So why was it so enjoyable? For one thing, the cast is great. Another nice touch is the music. I was overcome with the urge to get off the couch and start dancing. It sucked me in, and it'll do the same to you."

$x_4 = 3$ $x_5 = 0$ $x_1 = 3$

feature	description	value
x_1	Count positive words	3
x_2	Count negative words	2
x_3	$\begin{cases} 1 & if\ \text{"no"}\ in\ text \\ 0 & otherwise \end{cases}$	1
x_4	Count 1st and 2nd pronouns	3
x_5	$\begin{cases} 1 & if\ \text{"!"}\ in\ text \\ 0 & otherwise \end{cases}$	0
x_6	$\ln(word\ count)$	ln(56)=4.025

Materials adapted from Prof. Dan Jurafsky at Stanford

Logistic Regression

Example: Sentiment Classification

Input: "It's hokey . There are virtually no surprises , and the writing is second-rate . So why was it so enjoyable ? For one thing , the cast is great . Another nice touch is the music . I was overcome with the urge to get off the couch and start dancing . It sucked me in , and it'll do the same to you."

feature	description	value
x_1	Count positive words	3
x_2	Count negative words	2
x_3	$\begin{cases} 1 & \textit{if "no" in text} \\ 0 & \textit{otherwise} \end{cases}$	1
x_4	Count 1st and 2nd pronouns	3
x_5	$\begin{cases} 1 & \textit{if "!" in text} \\ 0 & \textit{otherwise} \end{cases}$	0
x_6	$\ln(\textit{word count})$	4.025

Output: positive (1) or negative (0)

Let's assume the weights and bias are:

$$\boldsymbol{w} = [w_1, \dots w_6]^T = [2.5, -5.0, -1.2, 0.5, 2.0, 0.7]^T$$

$$b = 0.1$$

Thus

$$P(semtiment = 1|\boldsymbol{x}) = \sigma\left(\boldsymbol{w}^T\boldsymbol{x} + b\right)$$

$$= \sigma([2.5, -5.0, -1.2, 0.5, 2.0, 0.7][3,2,1,3,0,4.025]^T + 0.1)$$

$$= 0.67$$

$$P(sentiment = 0|\boldsymbol{x}) = \boldsymbol{1} - \boldsymbol{\sigma}(\boldsymbol{wx} + b) = 0.33$$

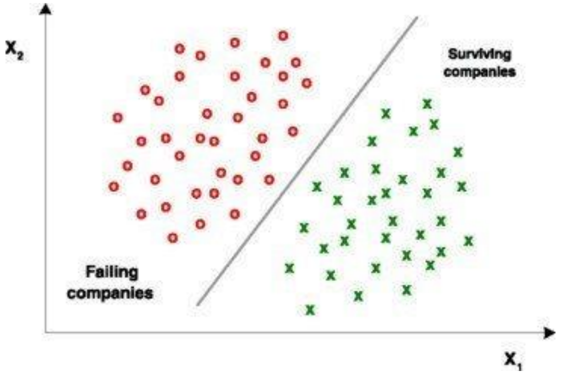
Materials adapted from Prof. Dan Jurafsky at Stanford

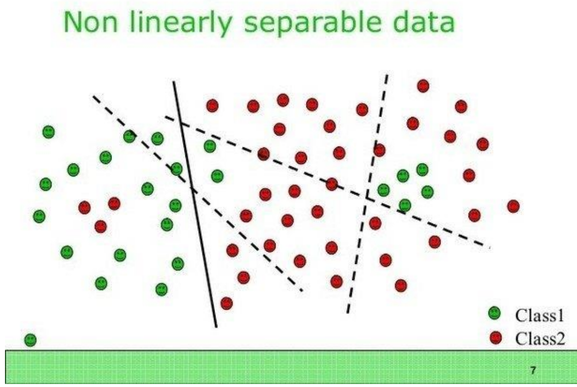
Logistic Regression

Pros and Cons

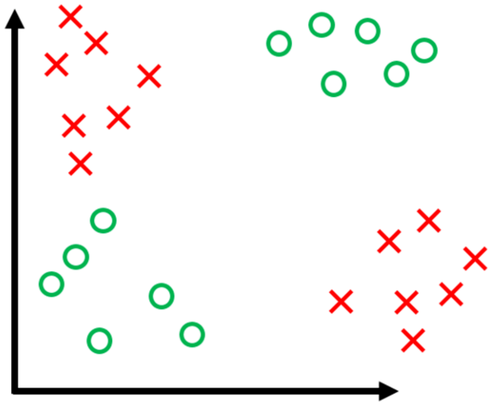
- Pros:
  - Simple to implement
  - No assumptions on feature probability distributions
  - Effective for linearly separable data
  - Provides probabilistic view of model prediction
- Cons:
  - Linear decision boundary: does not perform well on linearly non-separable data

linearly separable data





linearly non-separable data

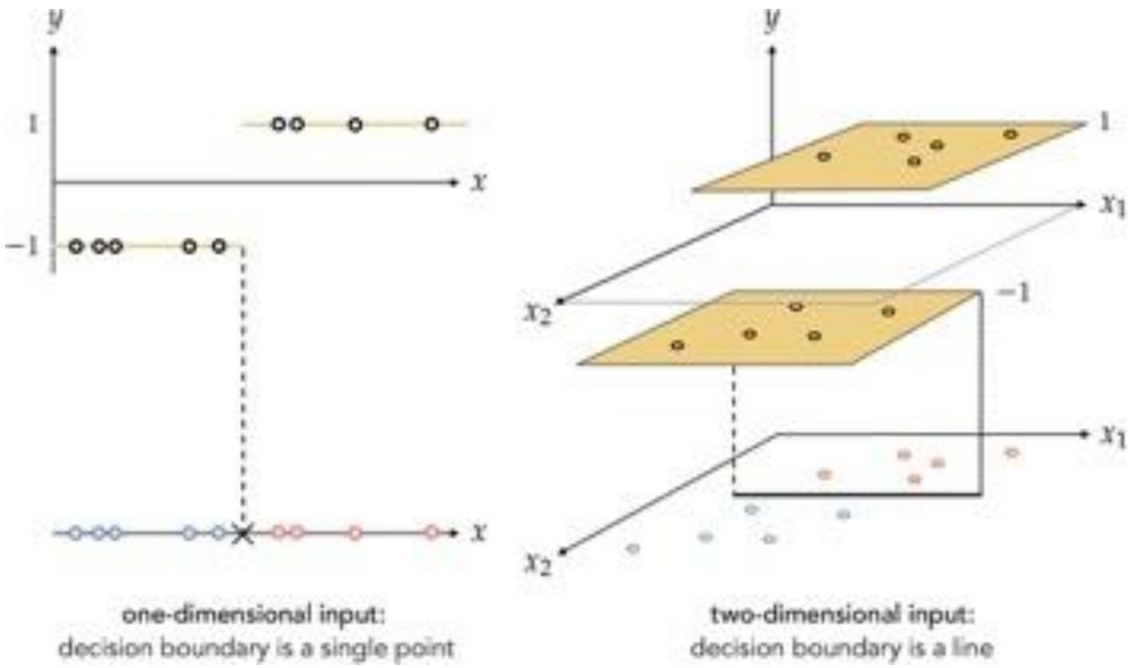


Logistic Regression

Common Applications

- Text categorization: Logistic regression could be used in document classification based on keywords. Common applications include sentiment classification and spam filtering.
- Health assessment: Logistic regression could be used to predict the risk of developing a given disease, based on observed characteristics of the patient
- Marketing: prediction of the tendency of customers to purchase a product or halt a subscription

- What if the labels come from different values?
- Considering a similar binary dataset of N samples $(\boldsymbol{x}_1, y_1)$, $(\boldsymbol{x}_2, y_2) \dots (\boldsymbol{x}_N, y_N)$.However, label y_i now takes on different values {-1,+1}:



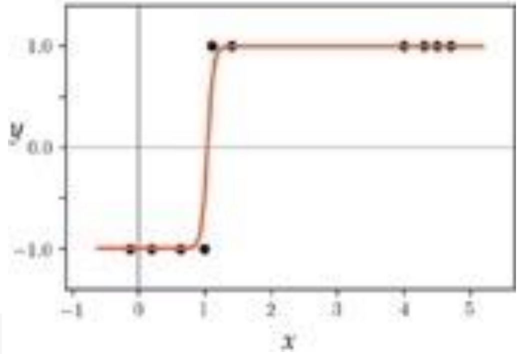
Materials adapted from Machine Learning Refined Book (2nd edition)

- The nonnegative sigmoid function does not fit such data. Instead, we could use a *sign* function:

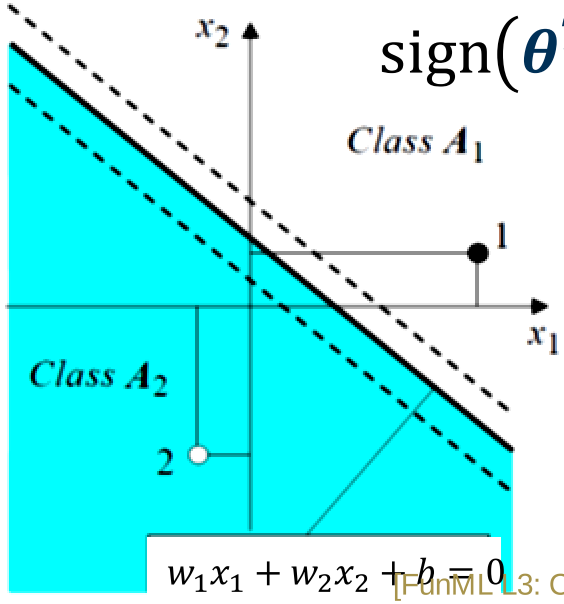
$$\mathrm{sign}(x) = \begin{cases} +1, if\ x > 0 \\ -1, if\ x < 0 \end{cases}$$

- Let $\mathrm{sign}(\boldsymbol{x}) = \mathrm{sign}(\boldsymbol{\theta}^T\boldsymbol{x})$, the function becomes a **linear model** that **directly *learns* a *linear decision boundary*** $\boldsymbol{\theta}^T\boldsymbol{x}$=0

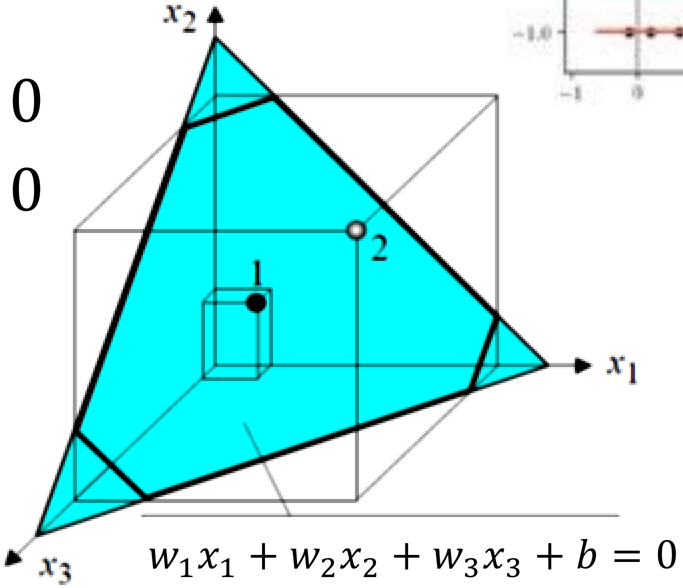
$$\mathrm{sign}(\boldsymbol{\theta}^T\boldsymbol{x}) = \begin{cases} +1, if\ \boldsymbol{\theta}^T\boldsymbol{x} > 0 \\ -1, if\ \boldsymbol{\theta}^T\boldsymbol{x} < 0 \end{cases}$$



$\mathrm{sign}(\boldsymbol{\theta}^T\boldsymbol{x})$



(*a*) Two-input perceptron.



(*b*) Three-input perceptron.

Materials adapted from Machine Learning Refined Book (2nd edition)

Logistic Regression
Extensions: Hinge Cost

- The desired set of parameters of $\text{sign}(\boldsymbol{\theta}^T \boldsymbol{x})$ define a ***linear* hyperplane** where:

$$\begin{cases} \boldsymbol{\theta}^T \boldsymbol{x} > 0, & if\ y = +1 \\ \boldsymbol{\theta}^T \boldsymbol{x} < 0, & if\ y = -1 \end{cases}$$

- The above conditions could be rewritten as following:

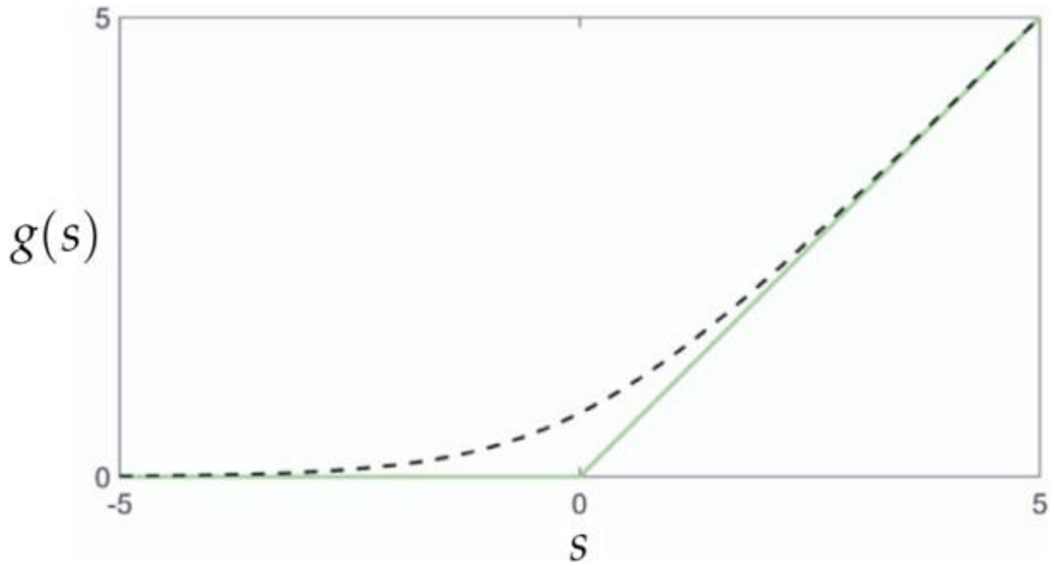
$$-y\boldsymbol{\theta}^T \boldsymbol{x} < 0$$

- Taking the maximum of $-y\boldsymbol{\theta}^T \boldsymbol{x}$ and 0, we can then write the cost function $L(\boldsymbol{x}, y, \boldsymbol{\theta})$ as:

$$L(\boldsymbol{x}, y, \boldsymbol{\theta}) = \max(0, -y\boldsymbol{\theta}^T \boldsymbol{x})$$

The function returns a 0 when the classification is correct and a +ve value when the classification is incorrect.

- This **convex** cost function is called ***hinge cost***
- Note that the *hinge* cost has a **trivial solution** at $\boldsymbol{\theta}$=0



$L(\boldsymbol{x}, y, \boldsymbol{\theta})$ (green)
Its softmax in dash curve

Materials adapted from Machine Learning Refined Book (2nd edition)

Logistic Regression

Extensions: Hinge Cost via Softplus

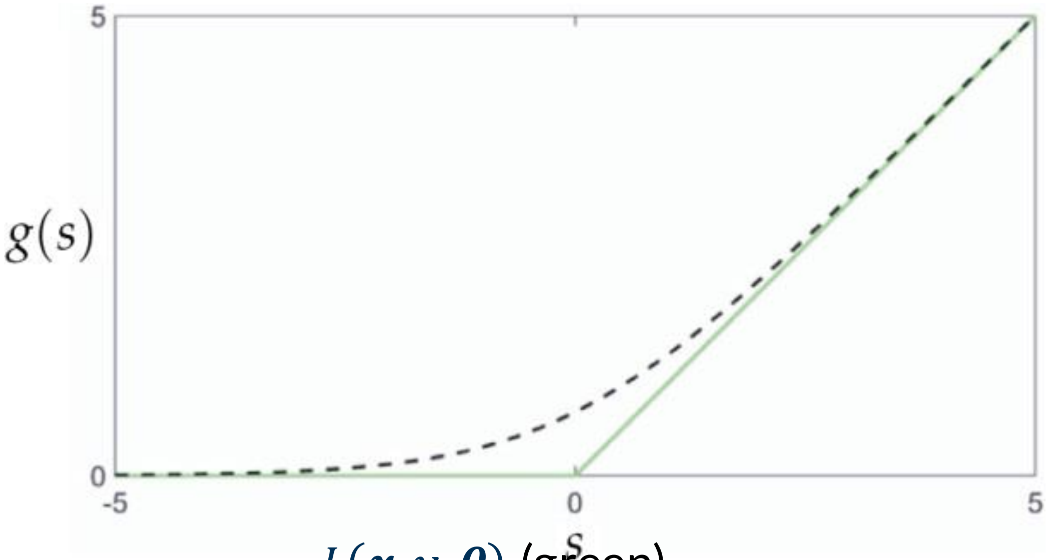
- $\max\left(0, -y\boldsymbol{\theta}^{\boldsymbol{T}}\boldsymbol{x}\right)$ is always nonnegative, we can take the average over the entire dataset to form a cost function:

$$L(\boldsymbol{x}, y, \boldsymbol{\theta}) = \frac{1}{N}\sum_{i=1}^{N} \max\left(0, -y_i\boldsymbol{\theta}^{\boldsymbol{T}}\boldsymbol{x}_i\right)$$

- This cost function has only a **single (discontinuous) derivative** in each input dimension, limiting specific optimization tools such as second-order methods.
- We could use the **Softplus** function $\log(1 + e^x)$ to approximate the hinge cost:

$$L(\boldsymbol{x}, y, \boldsymbol{\theta}) \approx \frac{1}{N}\sum_{i=1}^{N} \log\left(1 + e^{-y_i\boldsymbol{\theta}^{\boldsymbol{T}}\boldsymbol{x}_i}\right)$$

- The **convex softplus** cost has **infinitely many derivatives**, gradient descent and other methods can be applied
- Moreover, softplus does not have a trivial solution at 0



$L(\boldsymbol{x}, y, \boldsymbol{\theta})$ (green)
Softplus function (dashed line)

To see why the Softmax approximates the max function let us look at the simple case when $C = 2$. Suppose momentarily that $s_0 \leq s_1$, so that $\max(s_0, s_1) = s_1$. Therefore $\max(s_0, s_1)$ can be written as $\max(s_0, s_1) = s_0 + (s_1 - s_0)$, or equivalently as $\max(s_0, s_1) = \log(e^{s_0}) + \log(e^{s_1 - s_0})$. Written in this way we can see that $\log(e^{s_0}) + \log(1 + e^{s_1 - s_0}) = \log(e^{s_0} + e^{s_1}) = \text{soft}(s_0, s_1)$ is always larger than $\max(s_0, s_1)$ but not by much, especially when $e^{s_1 - s_0} \gg 1$. Since the same argument can be made if $s_0 \geq s_1$ we can say generally that $\text{soft}(s_0, s_1) \approx \max(s_0, s_1)$. The more general case follows similarly as well.

Materials adapted from Machine Learning Refined Book (2nd edition)

Logistic Regression

Extensions: Hinge Cost via Softplus

Recall the negative log-likelihood in logistic regression :

$$L(\boldsymbol{x}, y, \boldsymbol{\theta}) = \begin{cases} -\log\left(\sigma(\boldsymbol{\theta}^T \boldsymbol{x})\right), & if\ y = 1 \\ -\log\left(1 - \sigma(\boldsymbol{\theta}^T \boldsymbol{x})\right), & if\ y = 0 \end{cases}$$

Note that $1 - \sigma(x) = \sigma(-x)$, if y takes values from {1,-1}:

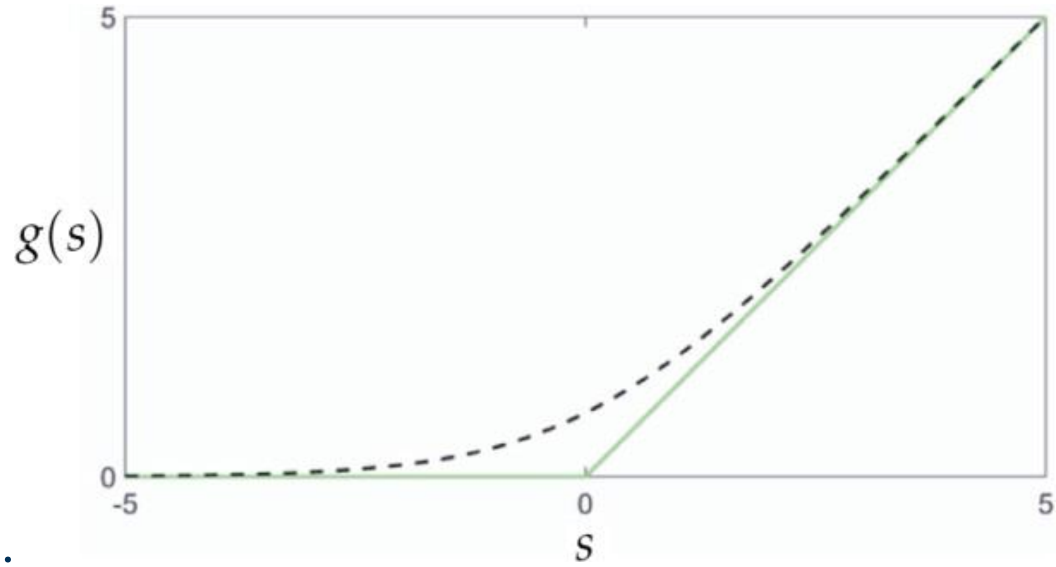
$$L(\boldsymbol{x}, y, \boldsymbol{\theta}) = \begin{cases} -\log\left(\sigma(\boldsymbol{\theta}^T \boldsymbol{x})\right), & if\ y = 1 \\ -\log\left(\sigma(-\boldsymbol{\theta}^T \boldsymbol{x})\right), & if\ y = -1 \end{cases}$$

Moving y into each σ, the above form can be re-written as:

$$L(\boldsymbol{x}, y, \boldsymbol{\theta}) = \frac{1}{N} \sum_{i=1}^{N} -\log\left(\sigma(y_i \boldsymbol{\theta}^T \boldsymbol{x_i})\right)$$

finally the above cost could be written Approximated hinge cost

$$L(\boldsymbol{x}, y, \boldsymbol{\theta}) = \frac{1}{N} \sum_{i=1}^{N} \log(1 + e^{-y_i \boldsymbol{\theta}^T \boldsymbol{x_i}})$$



$L(\boldsymbol{x}, y, \boldsymbol{\theta})$ (green)
softplus function (dashed line)

When the *softplus* is employed for optimizing *hinge cost*, there is no qualitative difference from the *logistic regression*

Logistic Regression vs Naïve Bayes

Naïve Bayes Recap

- Use Bayes' Formula to calculate the *posterior probability* for each class.
- The class with the highest posterior probability is the outcome of prediction.

Likelihood
P(feature|class)

Class Prior Probability
P(class)

$$P(y|\boldsymbol{x}) = \frac{P(\boldsymbol{x}|y)P(y)}{P(\boldsymbol{x})}$$

Predictor Prior Probability
P(feature)

Posterior Probability
P(class|feature)

Logistic Regression vs Naïve Bayes

Naïve Bayes Assumptions

- Naïve bayes assume that features are :
  - **equally important (no weights),** and
  - **conditionally independent**
- If two features x_1, x_2 are conditionally independent, then:

$$P(x_1, x_2|y) = P(x_1|y)P(x_2|y)$$

Hence, for conditional independent feature vector $\boldsymbol{x} = \left[x_1, x_2, \dots, x_p\right]^T$, the posterior probability of a class $y = c$ given the feature vector $\boldsymbol{x}$ is:

$$P(y = c|\boldsymbol{x}) = \frac{\prod_j P\left(x_j|y = c\right)P(y = c)}{P(\boldsymbol{x})} \propto \prod_j P\left(x_j|y = c\right)P(y = c)$$

Then, the class can be found as:

Maximum a posteriori (MAP) $$class = \arg\max_i P(y_i|\boldsymbol{x})$$

However, *feature correlation* could violate th assumption of conditionally independence

Logistic Regression vs Naïve Bayes

Connection to Naïve Bayes

- Suppose that number of classes K=2 and $X|Y=k \sim \mathcal{N}(\mu_k, \Sigma)$
- $p(x|y=1) = \phi(x; \mu_1, \Sigma)$
- Posterior $P(y=1|x) = \frac{p(y=1)\phi(x;\mu_1,\Sigma)}{p(y=1)\phi(x;\mu_1,\Sigma)+p(y=0)\phi(x;\mu_0,\Sigma)}$

$$= \frac{p(y=1)e^{-1/2(x-\mu_1)^T\Sigma^{-1}(x-\mu_1)}}{p(y=1)e^{-1/2(x-\mu_1)^T\Sigma^{-1}(x-\mu_1)} + p(y=0)e^{-1/2(x-\mu_0)^T\Sigma^{-1}(x-\mu_0)}}$$

$$= \frac{1}{1+\frac{p(y=0)}{p(y=1)}e^{1/2(x-\mu_1)^T\Sigma^{-1}(x-\mu_1) \quad -1/2(x-\mu_0)^T\Sigma^{-1}(x-\mu_0)}}$$

$$= \frac{1}{1+e^{-(\boldsymbol{w}^T\boldsymbol{x}+\boldsymbol{b})}}$$

- Gaussian Naïve Bayes classifier is linear classifier.
- Gaussian Naïve bayes has a similar form as logistic regression, however, Gaussian Naïve bayes assumes conditional independent features

Classifier Comparison

Methods	Assumptions on Feat. Dist.	Feat. Normalization	Cost Function	Regularization	Linear Classifier	Prob. View of Prediction	Generative/Discriminative	Parametric/ Non-parametric	Overfitting
Logistic Regression	No	Required	BCE (convex)	Additional term	Linear	Yes	Discriminative	Parametric	Not often
K Nearest Neighbors	No	Required	N/A	N/A	Non-linear	N/A	Discriminative	Non-parametric	when k is too small
Decision Trees	No	Not Required	N/A	N/A	Non-linear	N/A	Discriminative	Non-parametric	with large depth
Support Vector Machines	No	Required	Hinge (convex)	C (control robustness)	Linear/ Non-linear(kernel)	N/A	Discriminative	Parametric	Not often
Naïve Bayes	Conditional independent	Not Required	N/A	N/A	Non-linear /Linear (Gaussian)	Yes	Generative	Parametric	Not often
Artificial Neural Networks	No	Required	Non-convex	Additional term	Non-linear	Yes	Discriminative/ Generative	Parametric	with many layers

Appendix A: Notations

- x_i: a single feature
- $\boldsymbol{x}_i$: feature vector (data sample)
- $\boldsymbol{X}$: matrix of feature vectors (dataset)
- N: number of data samples
- m: degree of polynomial
- P: number of features in a feature vector
- θ_i: a single model coefficient (parameter)
- $\boldsymbol{\theta}$: coefficient vector
- ε: error margin
- α: learning rate
- γ: bias factor
- Bold letter/symbol: vector
- Bold capital letters/symbol: matrix