innovate achieve lead

Machine Learning
DSECL ZG565

Dr. Chetana Gavankar, Ph.D,
IIT Bombay-Monash University Australia
Chetana.gavankar@pilani.bits-pilani.ac.in

BITS Pilani
Pilani Campus

innovate achieve lead

BITS Pilani
Pilani Campus

Lecture No. – 5 | Probabilistic Discriminative Classifiers

Date – 23/11/2019

Time – 9:00 AM – 11:00 AM

Session Content

- Review of Naïve Bayes
  - Text classification model, image classification
- Discriminant Functions
- Probabilistic Discriminative Classifiers
- Logistic regression
- Difference between Naïve Bayes Classifier and Logistic Regression

Naive Bayes Classifier

- Assume target function $f : X \rightarrow V$, where each instance x described by attributes <a_1, a_2 … a_n>.
- Most probable value of $f(x)$ is:

$$v_{MAP} = \underset{v_j \in V}{\operatorname{argmax}} P(v_j | a_1, a_2 \ldots a_n)$$

$$v_{MAP} = \underset{v_j \in V}{\operatorname{argmax}} \frac{P(a_1, a_2 \ldots a_n | v_j) P(v_j)}{P(a_1, a_2 \ldots a_n)}$$

$$= \underset{v_j \in V}{\operatorname{argmax}} P(a_1, a_2 \ldots a_n | v_j) P(v_j)$$

Naive Bayes assumption:

$$P(a_1, a_2 \ldots a_n | v_j) = \prod_i P(a_i | v_j)$$

Naive Bayes classifier:

$$v_{NB} = \underset{v_j \in V}{\operatorname{argmax}} P(v_j) \prod_i P(a_i | v_j)$$

Naive Bayes Algorithm

- Naive Bayes Learn(*examples*)

  For each target value v_j

  $$\hat{P}(v_j) \leftarrow \text{estimate } P(v_j)$$

  For each attribute value a_i of each attribute a

  $$\hat{P}(a_i \mid v_j) \leftarrow \text{estimate } P(a_i \mid v_j)$$

- Classify New Instance(x)

$$v_{NB} = \underset{v_j \in V}{\operatorname{argmax}} \hat{P}(v_j) \prod_{a_i \in x} \hat{P}(a_i|v_j)$$

Example 1

Name	Give Birth	Can Fly	Live in Water	Have Legs	Class
human	yes	no	no	yes	mammals
python	no	no	no	no	non-mammals
salmon	no	no	yes	no	non-mammals
whale	yes	no	yes	no	mammals
frog	no	no	sometimes	yes	non-mammals
komodo	no	no	no	yes	non-mammals
bat	yes	yes	no	yes	mammals
pigeon	no	yes	no	yes	non-mammals
cat	yes	no	no	yes	mammals
leopard shark	yes	no	yes	no	non-mammals
turtle	no	no	sometimes	yes	non-mammals
penguin	no	no	sometimes	yes	non-mammals
porcupine	yes	no	no	yes	mammals
eel	no	no	yes	no	non-mammals
salamander	no	no	sometimes	yes	non-mammals
gila monster	no	no	no	yes	non-mammals
platypus	no	no	no	yes	mammals
owl	no	yes	no	yes	non-mammals
dolphin	yes	no	yes	no	mammals
eagle	no	yes	no	yes	non-mammals

Give Birth	Can Fly	Live in Water	Have Legs	Class
yes	no	yes	no	?

A: attributes

M: mammals

N: non-mammals

$$P(A|M) = \frac{6}{7} \times \frac{6}{7} \times \frac{2}{7} \times \frac{2}{7} = 0.06$$

$$P(A|N) = \frac{1}{13} \times \frac{10}{13} \times \frac{3}{13} \times \frac{4}{13} = 0.0042$$

$$P(A|M)P(M) = 0.06 \times \frac{7}{20} = 0.021$$

$$P(A|N)P(N) = 0.004 \times \frac{13}{20} = 0.0027$$

P(A|M)P(M) > P(A|N)P(N)

=> Mammals

Example 2

Text	Tag
“A great game”	Sports
“The election was over”	Not sports
“Very clean match”	Sports
“A clean but forgettable game”	Sports
“It was a close election”	Not sports

Which tag does the sentence *A very close game* belong to? i.e. P(sports| *A very close game*)

Feature Engineering: Bag of words i.e use word frequencies without considering order

Using Bayes Theorem:

P(sports| *A very close game*)

$$= \frac{P(\textit{A very close game} \mid \text{sports})\, P(\text{sports})}{P(\textit{A very close game})}$$

We assume that every word in a sentence is **independent** of the other ones

$$P(a\,very\,close\,game) = P(a) \times P(very) \times P(close) \times P(game)$$

$$P(a\,very\,close\,game|Sports) = P(a|Sports) \times P(very|Sports) \times P(close|Sports) \times P(game|Sports)$$

“close” doesn’t appear in sentences of sports tag, So P(close | sports) = 0, which makes product 0

Laplace smoothing

- Laplace smoothing: we add 1 or in general constant k to every count so it’s never zero.
- To balance this, we add the number of possible words to the divisor, so the division will never be greater than 1
- In our case, the possible words are ['a', 'great', 'very', 'over', 'it', 'but', 'game', 'election', 'clean', 'close', 'the', 'was', 'forgettable', 'match'].

Apply Laplace Smoothing

Word	P(word | Sports)	P(word | Not Sports)
a	2+1 / 11+14	1+1 / 9+14
very	1+1 / 11+14	0+1 / 9+14
close	0+1 / 11+14	1+1 / 9+14
game	2+1 / 11+14	0+1 / 9+14

$$P(a|Sports) \times P(very|Sports) \times P(close|Sports) \times P(game|Sports) \times P(Sports)$$
$$= 2.76 \times 10^{-5}$$
$$= 0.0000276$$

$$P(a|Not\ Sports) \times P(very|Not\ Sports) \times P(close|Not\ Sports) \times P(game|Not\ Sports) \times P(Not\ Sports)$$
$$= 0.572 \times 10^{-5}$$
$$= 0.00000572$$

Learning to Classify Text

LEARN_NAIVE_BAYES_TEXT (*Examples*, V)

1. *collect all words and other tokens that occur in Examples*

- *Vocabulary* $\leftarrow$ all distinct words and other tokens in *Examples*

2. *calculate the required $P(v_j)$ and $P(w_k \mid v_j)$ probability terms*

- For each target value v_j in V do
  - $docs_j \leftarrow$ subset of *Examples* for which the target value is v_j
  - $P(v_j) \leftarrow \frac{|docs_j|}{|Examples|}$
  - $Text_j \leftarrow$ a single document created by concatenating all members of $docs_j$

Learning to Classify Text

- $n \leftarrow$ total number of words in $Text_j$ (counting duplicate words multiple times)
- for each word w_k in *Vocabulary*
  - * $n_k \leftarrow$ number of times word w_k occurs in $Text_j$
  - * $P(w_k|v_j) \leftarrow \frac{n_k+1}{n+|Vocabulary|}$

CLASSIFY_NAIVE_BAYES_TEXT (*Doc*)

- *positions* $\leftarrow$ all word positions in *Doc* that contain tokens found in *Vocabulary*
- Return v_{NB} where $v_{NB} = \underset{v_j \in V}{\operatorname{argmax}} P(v_j) \prod_{i \in positions} P(a_i|v_j)$

LEARN_NAIVE_BAYES_TEXT($Examples, V$)

Examples is a set of text documents along with their target values. V is the set of all possible target values. This function learns the probability terms $P(w_k|v_j)$, describing the probability that a randomly drawn word from a document in class v_j will be the English word w_k. It also learns the class prior probabilities $P(v_j)$.

1. *collect all words, punctuation, and other tokens that occur in Examples*
   - $Vocabulary \leftarrow$ the set of all distinct words and other tokens occurring in any text document from $Examples$
2. *calculate the required $P(v_j)$ and $P(w_k|v_j)$ probability terms*
   - For each target value v_j in V do
     - $docs_j \leftarrow$ the subset of documents from $Examples$ for which the target value is v_j
     - $P(v_j) \leftarrow \frac{|docs_j|}{|Examples|}$
     - $Text_j \leftarrow$ a single document created by concatenating all members of $docs_j$
     - $n \leftarrow$ total number of distinct word positions in $Text_j$
     - for each word w_k in $Vocabulary$
       - $n_k \leftarrow$ number of times word w_k occurs in $Text_j$
       - $P(w_k|v_j) \leftarrow \frac{n_k+1}{n+|Vocabulary|}$

CLASSIFY_NAIVE_BAYES_TEXT(Doc)

Return the estimated target value for the document Doc. a_i denotes the word found in the ith position within Doc.

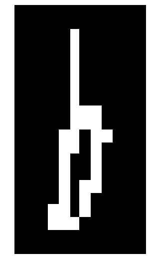
- $positions \leftarrow$ all word positions in Doc that contain tokens found in $Vocabulary$
- Return v_{NB}, where

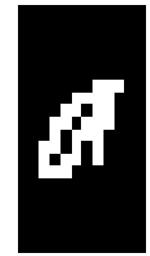
$$v_{NB} = \underset{v_j \in V}{\operatorname{argmax}} P(v_j) \prod_{i \in positions} P(a_i|v_j)$$

Estimating Parameters: X_i Continuous

What if features are continuous?

- E.g., character recognition: X_i is intensity at ith pixel





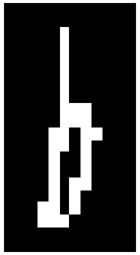
- Gaussian Naïve Bayes (GNB):

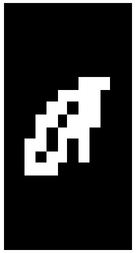
$$P(X_i = x | Y = y_k) = \frac{1}{\sigma_{ik}\sqrt{2\pi}} e^{-\frac{(x-\mu_{ik})^2}{2\sigma_{ik}^2}}$$

distribution of feature X_i is Gaussian with a mean and variance that can depend on the label y_k and which feature X_i it is

What if features are continuous?

- E.g., character recognition: X_i is intensity at ith pixel





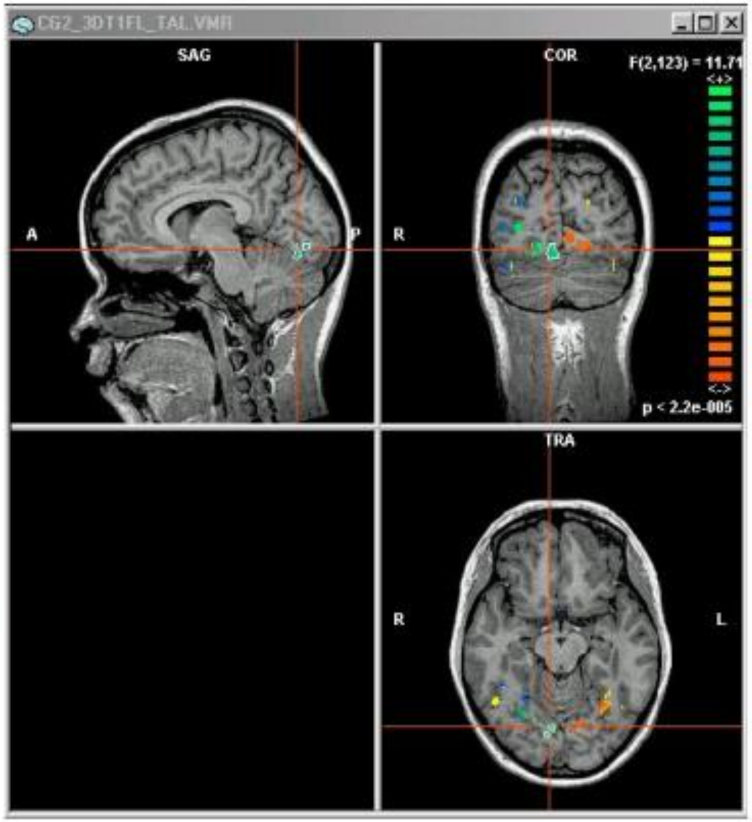
- Gaussian Naïve Bayes (GNB):

$$P(X_i = x | Y = y_k) = \frac{1}{\sigma_{ik}\sqrt{2\pi}}\, e^{-\frac{(x-\mu_{ik})^2}{2\sigma_{ik}^2}}$$

- Different mean and variance for each class k and each pixel i.
- Sometimes assume variance:
  - Is independent of Y (i.e., just have σ_i)
  - Or independent of X (i.e., just have σ_k)
  - Or both (i.e., just have σ)

Example: GNB for classifying mental states

[Mitchell et al.]



- Classify a person's cognitive state, based on brain image
  - reading a sentence or viewing a picture?
  - reading the word describing a "Tool" or "Building"?
  - reading the word describing a "Person" or an "Animal"?
- Training: Patients were shown words of different categories and then a measurement was done to see what parts of the brain responded.

Example: GNB for classifying mental states

[Mitchell et al.]

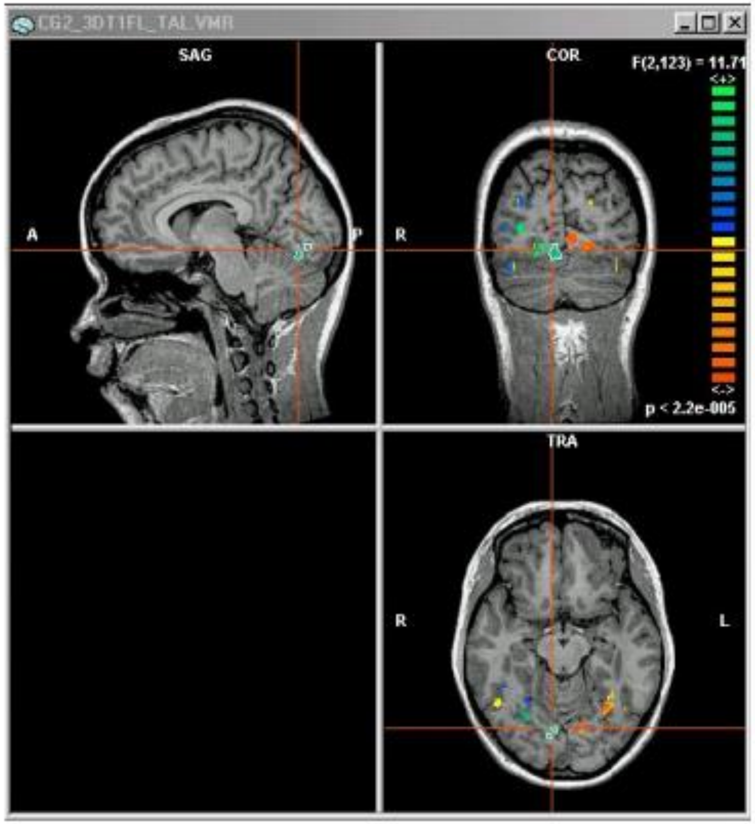
~1mm resolution

~2 images per sec.

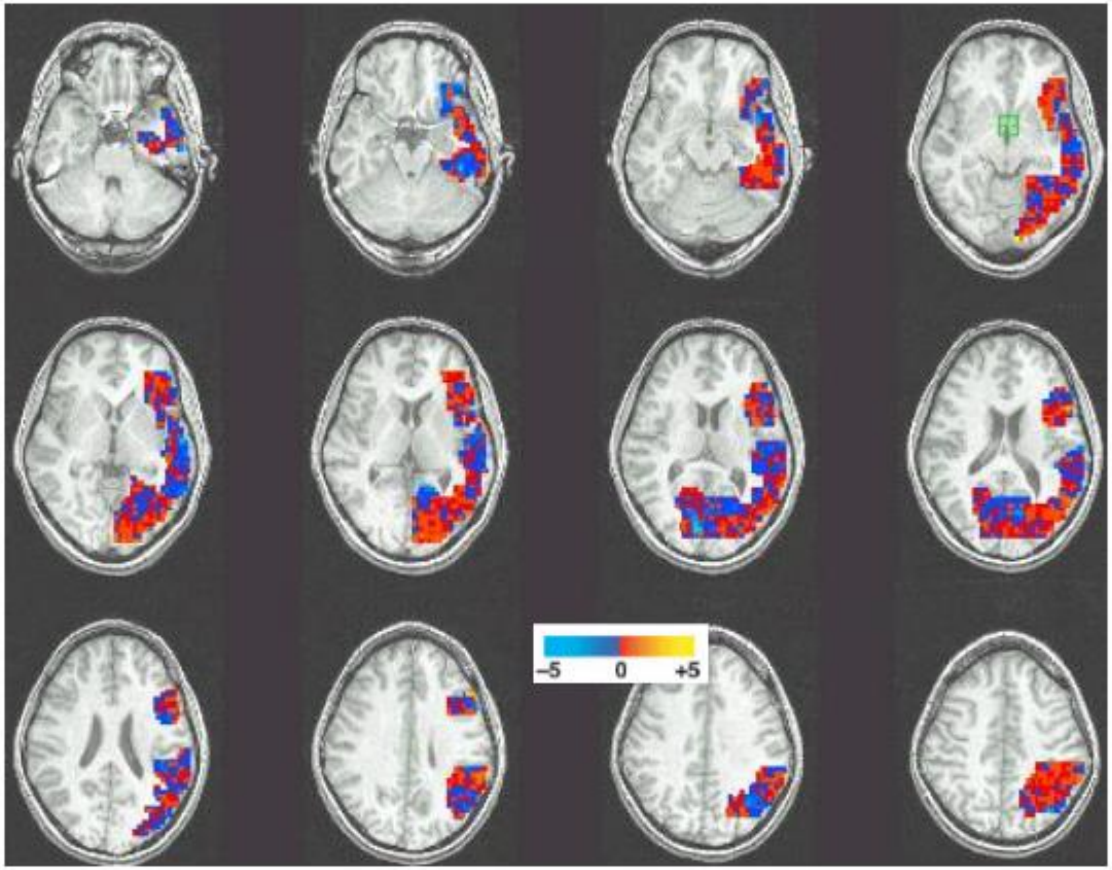
15,000 voxels/image

Non-invasive, save

Measures Blood Oxygen Level Dependent response (BOLD)



Gaussian Naïve Bayes: Learned $\mu_{\text{voxel,word}}$



[Mitchell et al.]

15,000 voxels or features

10 training examples or subjects per class

Logistic Regression

Logistic Regression

Idea:

- Naïve Bayes allows computing P(Y|X) by learning P(Y) and P(X|Y)

- Why not learn P(Y|X) directly?

Linear Regression versus logistic regression

- **Linear Regression** could help us predict the student’s test score on a scale of 0 - 100. Linear regression predictions are continuous (numbers in a range).
- **Logistic Regression** could help use predict whether the student passed or failed. Logistic regression predictions are discrete (only specific values or categories are allowed). We can also view probability scores underlying the model’s classifications.

Linear Regression versus Logistic Regression

Classification requires discrete values:
y = 0 or 1

For linear Regression output values:

$h_\theta(x)$ can be much > 1 or much < 0

Logistic Regression: $0 \leq h_\theta(x) \leq 1$

Sigmoid/Logistic Function

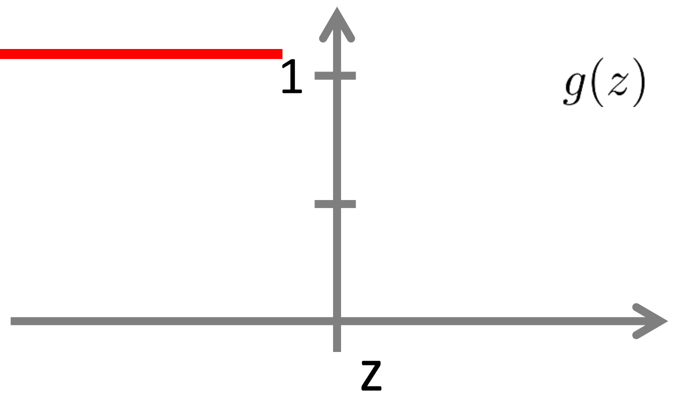
- Sigmoid/logistic function takes a real value as input and outputs another value between 0 and 1
- That framework is called logistic regression
  - Logistic: A special mathematical sigmoid function it uses
  - Regression: Combines a weight vector with observations to create an answer

$$h_\theta(x) = g(\theta^T x)$$

Logistic Regression

$$h_\theta(x) = g(\theta^T x)$$

$$g(z) = \frac{1}{1+e^{-z}}$$



Suppose predict "$y = 1$" if $h_\theta(x) \geq 0.5$

predict "$y = 0$" if $h_\theta(x) < 0.5$

Learning model parameters

Training set: $\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \cdots, (x^{(m)}, y^{(m)})\}$

m examples $\quad x \in \begin{bmatrix} x_0 \\ x_1 \\ \dots \\ x_n \end{bmatrix} \quad x_0 = 1, y \in \{0, 1\}$

$$h_\theta(x) = \frac{1}{1 + e^{-\theta^T x}}$$

How to choose parameters (feature weights) θ ?

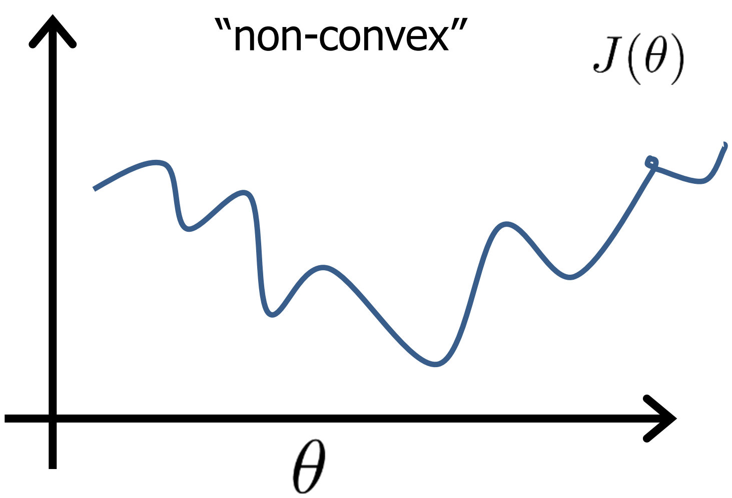
Error (Cost) Function

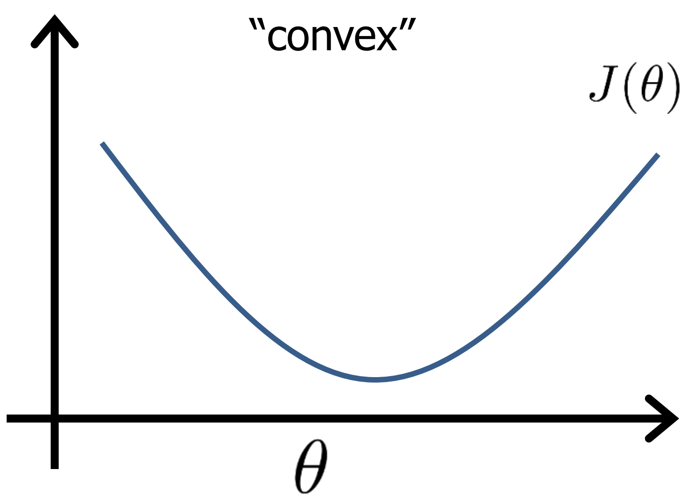
- Our prediction function is non-linear (due to sigmoid transform)
- Squaring this prediction as we do in MSE results in a non-convex function with many local minima.
- If our cost function has many local minimums, gradient descent may not find the optimal global minimum.
- So instead of Mean Squared Error, we use a error/ cost function called Cross-Entropy, also known as Log Loss.

MSE Cost Function

Linear regression: $J(\theta) = \frac{1}{m} \sum_{i=1}^{m} \frac{1}{2} \left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$

$$\mathrm{Cost}(h_\theta(x^{(i)}), y^{(i)}) = \frac{1}{2} \left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$$

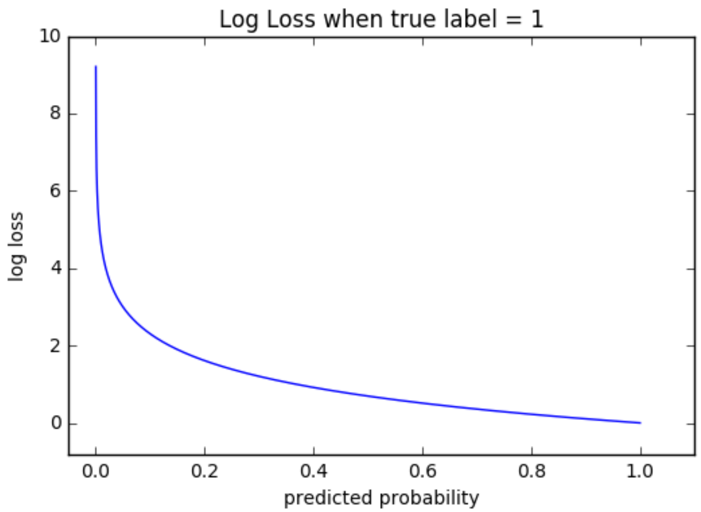




Cross Entropy

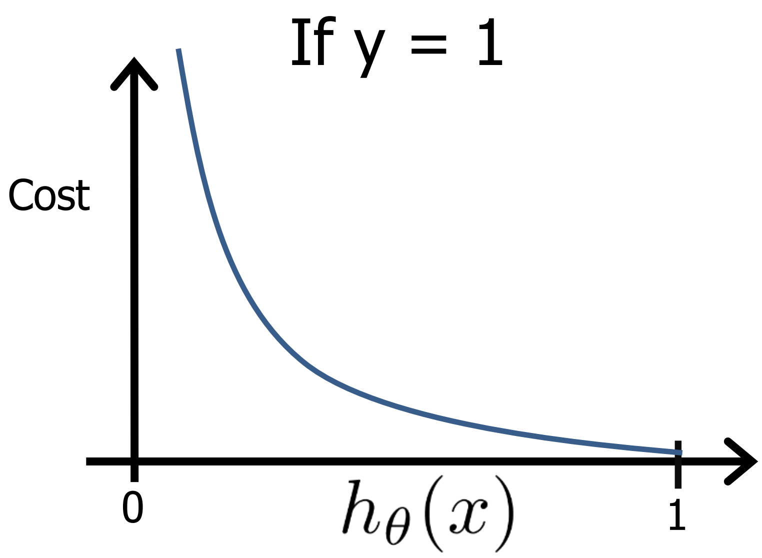
- Cross-entropy loss, or log loss, measures the performance of a classification model whose output is a probability value between 0 and 1.
- Cross-entropy loss increases as the predicted probability diverges from the actual label.
- So predicting a probability of .012 when the actual observation label is 1 would be bad and result in a high loss value.
- A perfect model would have a log loss of 0.

- Cross-entropy loss can be divided into two separate cost functions:
  one for y=1 and
  one for y=0.



Logistic regression cost function (cross entropy)

$$\mathrm{Cost}(h_\theta(x), y) = \begin{cases} -\log(h_\theta(x)) & \text{if } y = 1 \\ -\log(1 - h_\theta(x)) & \text{if } y = 0 \end{cases}$$



$\mathrm{Cost} = 0 \text{ if } y = 1, h_\theta(x) = 1$

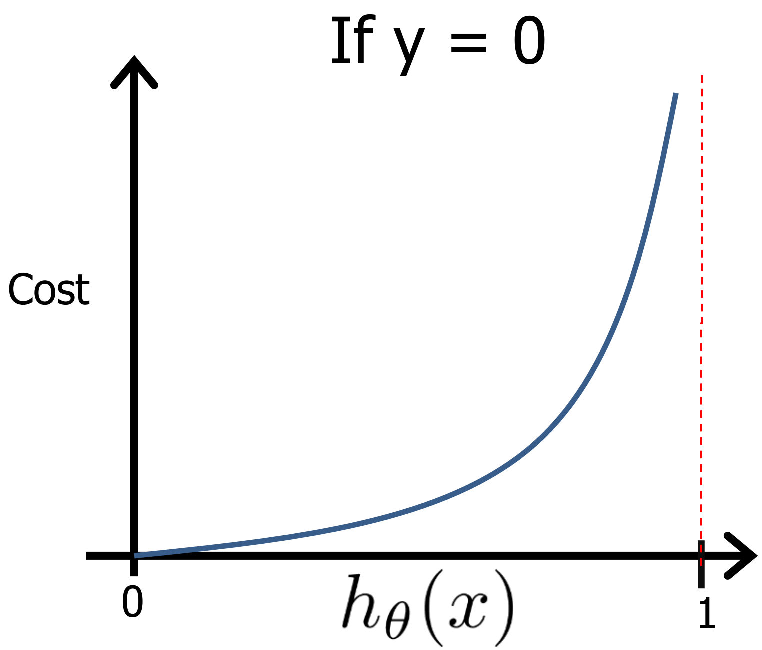
But as $h_\theta(x) \to 0$

$Cost \to \infty$

Captures intuition that if $h_\theta(x) = 0$, (predict $P(y = 1|x; \theta) = 0$), but $y = 1$, we'll penalize learning algorithm by a very large cost.

Logistic regression cost function

$$\mathrm{Cost}(h_\theta(x), y) = \begin{cases} -\log(h_\theta(x)) & \text{if } y = 1 \\ -\log(1 - h_\theta(x)) & \text{if } y = 0 \end{cases}$$



Cost=0; If y=0 and $h_\theta(x)$=0

Cost function

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m} \mathrm{Cost}(h_\theta(x^{(i)}), y^{(i)})$$
$$= -\frac{1}{m}[\sum_{i=1}^{m} y^{(i)} \log h_\theta(x^{(i)}) + (1 - y^{(i)}) \log (1 - h_\theta(x^{(i)}))]$$

To fit parameters θ : [Apply Gradient Descent Algorithm]

$$\min_\theta J(\theta)$$

To make a prediction given new x:

Output $h_\theta(x) = \frac{1}{1+e^{-\theta^T x}}$

Derivative of sigmoid function

- Maximum likelihood to determine the parameters of the logistic regression model.
- To do this, we shall make use of the derivative of the logistic sigmoid function
- Use any algorithm like the gradient descent algorithm to minimize cost function by using derivative

https://towardsdatascience.com/derivative-of-the-sigmoid-function-536880cf918e

Logistic Regression

- Consider learning f: X → Y, where
  - X is a vector of real-valued features, < X_1 … X_n >
  - Y is boolean
  - assume all X_i are conditionally independent given Y
  - model $P(X_i \mid Y = y_k)$ as Gaussian $N(\mu_{ik},\sigma_i)$
  - model P(Y) as Bernoulli (two-point) distribution(π)

- What does that imply about the form of P(Y|X)?

$$P(Y = 1|X =< X_1, ...X_n >) = \frac{1}{1 + exp(w_0 + \sum_i w_i X_i)}$$

Very convenient!

$$P(Y=1|X=<X_1,...X_n>) = \frac{1}{1+exp(w_0+\sum_i w_i X_i)}$$

implies

$$P(Y=0|X=<X_1,...X_n>) = \frac{exp(w_0+\sum_i w_i X_i)}{1+exp(w_0+\sum_i w_i X_i)}$$

implies

$$\frac{P(Y=0|X)}{P(Y=1|X)} = exp(w_0+\sum_i w_i X_i)$$

implies

$$\ln\frac{P(Y=0|X)}{P(Y=1|X)} = w_0+\sum_i w_i X_i$$

Very convenient!

$$P(Y=1|X=<X_1,...X_n>) = \frac{1}{1+exp(w_0+\sum_i w_i X_i)}$$

implies

$$P(Y=0|X=<X_1,...X_n>) = \frac{exp(w_0+\sum_i w_i X_i)}{1+exp(w_0+\sum_i w_i X_i)}$$

implies

$$\frac{P(Y=0|X)}{P(Y=1|X)} = exp(w_0+\sum_i w_i X_i)$$

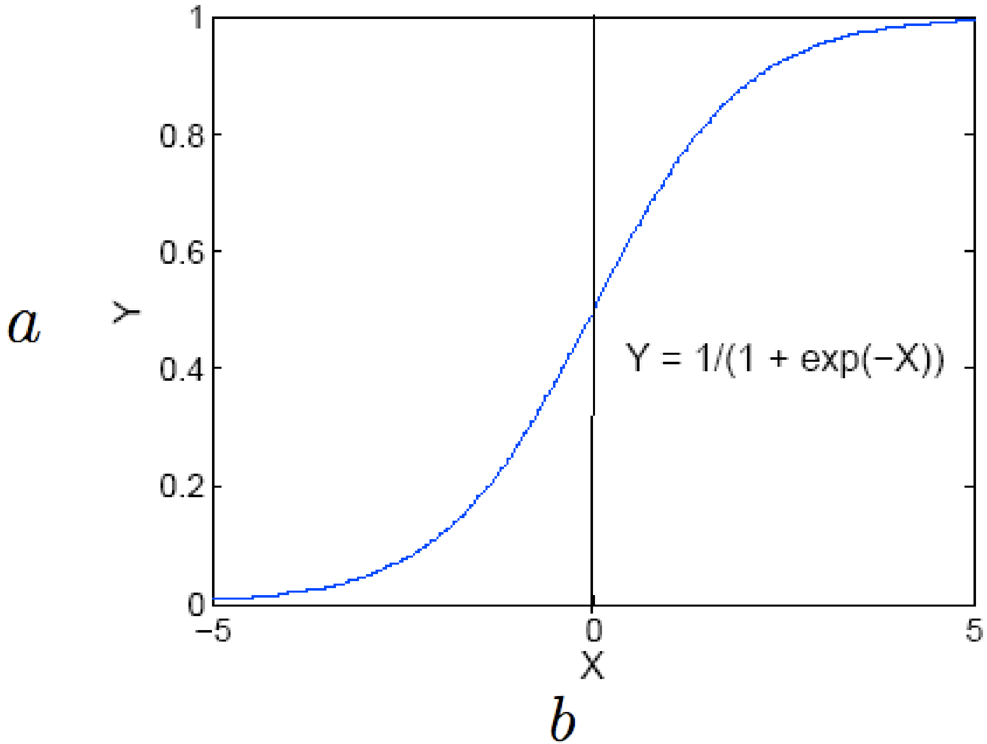
linear classification rule!

implies

$$\ln\frac{P(Y=0|X)}{P(Y=1|X)} = w_0+\sum_i w_i X_i$$

Logistic function

$$a = \frac{1}{1 + \exp(-b)}$$



$$P(Y = 1|X) = \frac{1}{1 + \exp(w_0 + \sum_{i=1}^{n} w_i X_i)}$$

Logistic Regression and Gaussian Naïve Bayes Classifier

- Interestingly, the parametric form of P(Y|X) used by Logistic Regression is precisely the form implied by the assumptions of a Gaussian Naive Bayes classifier.

- Therefore, we can view Logistic Regression as a closely related alternative to GNB, though the two can produce different results in many cases

innovate achieve lead

Derive form for P(Y|X) for Gaussian $P(X_i \mid Y=y_k)$ assuming $\sigma_{ik} = \sigma_i$

$$P(Y=1|X) = \frac{P(Y=1)P(X|Y=1)}{P(Y=1)P(X|Y=1) + P(Y=0)P(X|Y=0)}$$

$$= \frac{1}{1 + \frac{P(Y=0)P(X|Y=0)}{P(Y=1)P(X|Y=1)}}$$

$$= \frac{1}{1 + \exp(\ln \frac{P(Y=0)P(X|Y=0)}{P(Y=1)P(X|Y=1)})}$$

$$P(x \mid y_k) = \frac{1}{\sigma_{ik}\sqrt{2\pi}} \; e^{\frac{-(x-\mu_{ik})^2}{2\sigma_{ik}^2}}$$

$; As\ P(Y=1) = \pi$

$$= \frac{1}{1 + \exp(\ (\ln \frac{1-\pi}{\pi}) + \boxed{\sum_i \ln \frac{P(X_i|Y=0)}{P(X_i|Y=1)}})}$$

$$\boxed{\sum_i \left(\frac{\mu_{i0} - \mu_{i1}}{\sigma_i^2} X_i + \frac{\mu_{i1}^2 - \mu_{i0}^2)}{2\sigma_i^2} \right)}$$

$$P(Y=1|X) = \frac{1}{1 + \exp(w_0 + \sum_{i=1}^n w_i X_i)}$$

Gaussian P (X_i|Y =y_k)

$$
\begin{aligned}
\sum_i \ln \frac{P(X_i|Y=0)}{P(X_i|Y=1)} &= \sum_i \ln \frac{\frac{1}{\sqrt{2\pi\sigma_i^2}} \exp\left(\frac{-(X_i-\mu_{i0})^2}{2\sigma_i^2}\right)}{\frac{1}{\sqrt{2\pi\sigma_i^2}} \exp\left(\frac{-(X_i-\mu_{i1})^2}{2\sigma_i^2}\right)} \\
&= \sum_i \ln \exp\left(\frac{(X_i-\mu_{i1})^2-(X_i-\mu_{i0})^2}{2\sigma_i^2}\right) \\
&= \sum_i \left(\frac{(X_i-\mu_{i1})^2-(X_i-\mu_{i0})^2}{2\sigma_i^2}\right) \\
&= \sum_i \left(\frac{(X_i^2-2X_i\mu_{i1}+\mu_{i1}^2)-(X_i^2-2X_i\mu_{i0}+\mu_{i0}^2)}{2\sigma_i^2}\right) \\
&= \sum_i \left(\frac{2X_i(\mu_{i0}-\mu_{i1})+\mu_{i1}^2-\mu_{i0}^2}{2\sigma_i^2}\right) \\
&= \sum_i \left(\frac{\mu_{i0}-\mu_{i1}}{\sigma_i^2}X_i+\frac{\mu_{i1}^2-\mu_{i0}^2}{2\sigma_i^2}\right)
\end{aligned}
$$

Parameter estimation of generic logistic regression

- Logistic Regression holds in many problem settings beyond the GNB problem

- General method required for estimating it in a more broad range of cases.

- In many cases we may suspect the GNB assumptions are not perfectly satisfied.

- We may wish to estimate the wi parameters directly from the data

Estimating parameters

- we have L training examples: $\{\langle X^1, Y^1\rangle, \ldots \langle X^L, Y^L\rangle\}$

- maximum likelihood estimate for parameters W

$$= \arg\max_W \prod_l P(< X^l, Y^l > | W)$$

- maximum <u>conditional</u> likelihood estimate

Training Logistic Regression: MCLE

- Choose parameters $W=<w_0, \ldots w_n>$ to maximize conditional likelihood of training data

where
$$P(Y=0|X,W) = \frac{1}{1+exp(w_0+\sum_i w_i X_i)}$$
$$P(Y=1|X,W) = \frac{exp(w_0+\sum_i w_i X_i)}{1+exp(w_0+\sum_i w_i X_i)}$$

- Training data D = $\{\langle X^1, Y^1\rangle, \ldots \langle X^L, Y^L\rangle\}$
- Data likelihood = $\prod_l P(X^l, Y^l|W)$
- Data conditional likelihood = $\prod_l P(Y^l|X^l, W)$

$$W_{MCLE} = \arg\max_W \prod_l P(Y^l|W, X^l)$$

Expressing Conditional Log Likelihood

$$l(W) \equiv \ln \prod_l P(Y^l|X^l, W) = \sum_l \ln P(Y^l|X^l, W)$$

$$P(Y = 0|X, W) = \frac{1}{1 + exp(w_0 + \sum_i w_i X_i)}$$

$$P(Y = 1|X, W) = \frac{exp(w_0 + \sum_i w_i X_i)}{1 + exp(w_0 + \sum_i w_i X_i)}$$

$$\begin{aligned} l(W) &= \sum_l Y^l \ln P(Y^l = 1|X^l, W) + (1 - Y^l) \ln P(Y^l = 0|X^l, W) \\ &= \sum_l Y^l \ln \frac{P(Y^l = 1|X^l, W)}{P(Y^l = 0|X^l, W)} + \ln P(Y^l = 0|X^l, W) \\ &= \sum_l Y^l (w_0 + \sum_i^n w_i X_i^l) - \ln(1 + exp(w_0 + \sum_i^n w_i X_i^l)) \end{aligned}$$

Maximizing Conditional Log Likelihood

$$P(Y=0|X,W) = \frac{1}{1+exp(w_0+\sum_i w_i X_i)}$$

$$P(Y=1|X,W) = \frac{exp(w_0+\sum_i w_i X_i)}{1+exp(w_0+\sum_i w_i X_i)}$$

$$\begin{aligned} l(W) &\equiv \ln \prod_l P(Y^l|X^l,W) \\ &= \sum_l Y^l(w_0+\sum_i^n w_i X_i^l) - \ln(1+exp(w_0+\sum_i^n w_i X_i^l)) \end{aligned}$$

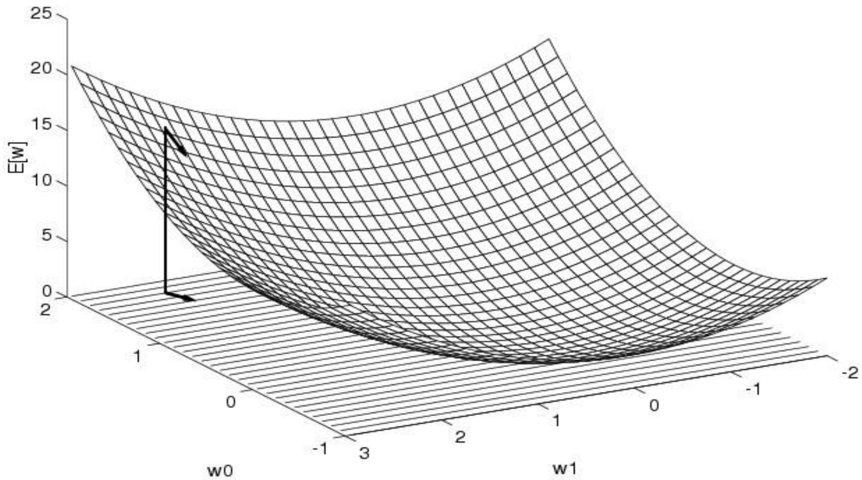
Bad news: no closed-form solution(that can be evaluated in a finite number of operations) to maximize *l(W)*

Maximize Conditional Log Likelihood: Gradient Ascent

$$
\begin{aligned}
l(W) &\equiv \ln \prod_l P(Y^l | X^l, W) \\
&= \sum_l Y^l (w_0 + \sum_i^n w_i X_i^l) - \ln(1 + exp(w_0 + \sum_i^n w_i X_i^l))
\end{aligned}
$$

$$
\frac{\partial l(W)}{\partial w_i} = \sum_l X_i^l (Y^l - \hat{P}(Y^l = 1 | X^l, W))
$$

Gradient Descent



Gradient

$$\nabla E[\vec{w}] \equiv \left[\frac{\partial E}{\partial w_0}, \frac{\partial E}{\partial w_1}, \cdots \frac{\partial E}{\partial w_n}\right]$$

Training rule:

$$\Delta \vec{w} = -\eta \nabla E[\vec{w}]$$

i.e.,

$$\Delta w_i = -\eta \frac{\partial E}{\partial w_i}$$

Maximize Conditional Log Likelihood: Gradient Ascent

$$l(W) \equiv \ln \prod_l P(Y^l|X^l, W)$$

$$= \sum_l Y^l(w_0 + \sum_i^n w_i X_i^l) - \ln(1 + exp(w_0 + \sum_i^n w_i X_i^l))$$

$$\frac{\partial l(W)}{\partial w_i} = \sum_l X_i^l(Y^l - \hat{P}(Y^l = 1|X^l, W))$$

Gradient ascent algorithm: iterate until change < ε

For all i, repeat

$$w_i \leftarrow w_i + \eta \sum_l X_i^l(Y^l - \hat{P}(Y^l = 1|X^l, W))$$

MAP

- Maximum conditional likelihood estimate

$$W \leftarrow \arg\max_W \ \ln \prod_l P(Y^l | X^l, W)$$

$$\boxed{w_i \leftarrow w_i + \eta \sum_l X_i^l (Y^l - \hat{P}(Y^l = 1 | X^l, W))}$$

MAP estimates and Regularization

- Maximum a posteriori estimate

$$W \leftarrow \arg\max_W \ \ln \prod_l P(Y^l | X^l, W)$$

$$\boxed{w_i \leftarrow w_i + \eta \sum_l X_i^l (Y^l - \hat{P}(Y^l = 1 | X^l, W))}$$

$$\boxed{w_i \leftarrow w_i - \eta \lambda w_i + \eta \sum_l X_i^l (Y^l - \hat{P}(Y^l = 1 | X^l, W))}$$

λ is called a "regularization" term
- helps reduce overfitting
- keep weights nearer to zero
- used very frequently in Logistic Regression

The Bottom Line

- Consider learning f: X → Y, where
  - X is a vector of real-valued features, $< X_1 \ldots X_n >$
  - Y is boolean
  - assume all X_i are conditionally independent given Y
  - model $P(X_i \mid Y = y_k)$ as Gaussian $N(\mu_{ik},\sigma_i)$
  - model P(Y) as Bernoulli (π)

- Then P(Y|X) is of this form, and we can directly estimate W

$$P(Y = 1|X =< X_1, \ldots X_n >) = \frac{1}{1 + exp(w_0 + \sum_i w_i X_i)}$$

Logistic regression more generally

- Logistic regression when Y not boolean (but still discrete-valued).
- Now $y \in \{y_1 \ldots y_R\}$: learn R-1 sets of weights

for *k<R* $$P(Y = y_k|X) = \frac{\exp(w_{k0} + \sum_{i=1}^{n} w_{ki} X_i)}{1 + \sum_{j=1}^{R-1} \exp(w_{j0} + \sum_{i=1}^{n} w_{ji} X_i)}$$

for *k=R* $$P(Y = y_R|X) = \frac{1}{1 + \sum_{j=1}^{R-1} \exp(w_{j0} + \sum_{i=1}^{n} w_{ji} X_i)}$$

How does logistic regression handle missing values?

- Replace missing values with column averages (i.e. replace missing values in feature 1 with the average for feature 1).
- Replace missing values with column medians.
- Impute missing values using the other features.
- Remove records that are missing features.
- Use a machine learning technique that uses classification trees, e.g. Decision tree

Logistic Regression Applications

- **Credit Card Fraud** : Predicting if a given credit card transaction is fraud or not
- **Health** : Predicting if a given mass of tissue is benign or malignant
- **Marketing** : Predicting if a given user will buy an insurance product or not
- **Banking** : Predicting if a customer will default on a loan.

Generative vs. Discriminative Classifiers

Training classifiers involves estimating f: X → Y, or P(Y|X)

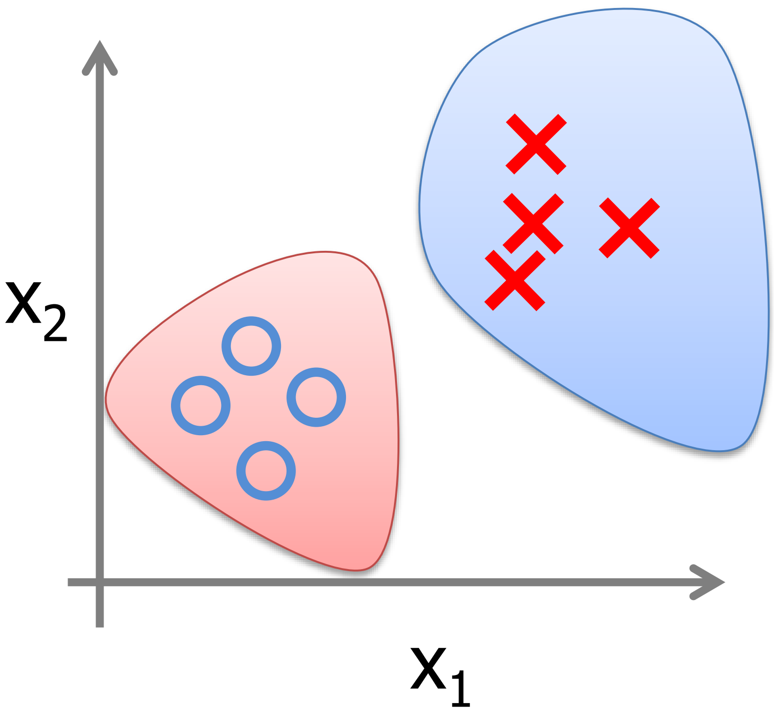
Generative classifiers (e.g., Naïve Bayes)
- Assume some functional form for P(X|Y), P(X)
- Estimate parameters of P(X|Y), P(X) directly from training data
- Use Bayes rule to calculate $P(Y|X = x_i)$

Discriminative classifiers (e.g., Logistic regression)

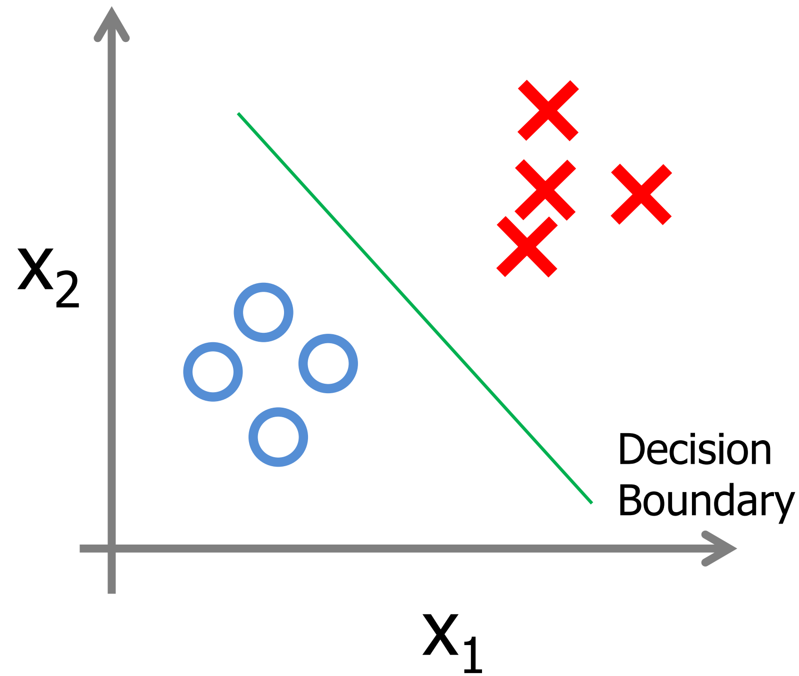
- Assume some functional form for P(Y|X)
- Estimate parameters of P(Y|X) directly from training data

Probabilistic Generative Model versus Probabilistic Discriminative Model

Generative:



Discriminative:



Probabilistic Generative Model versus Probabilistic Discriminative Model

Generative	Discriminative
Ex: Naïve Bayes	Ex: Logistic Regression
Estimate $P(Y)$ and $P(X\|Y)$	Finds class label directly $P(Y\|X)$
Prediction $\hat{y} = \text{argmax}_y\, P(Y = y)P(X = x\|Y = y)$	Prediction $\hat{y} = P(Y = y\|X = x)$

Naïve Bayes versus Logistic Regression

- Naïve Bayes are Generative Models
- Logistic Regression are Discriminative Models
- Naïve Bayes easy to construct
- Naive Bayes also assumes that the features are conditionally independent. Real data sets are never perfectly independent
- When the training size reaches infinity, logistic regression performs better than the generative model Naive Bayes.
  - Optional reading by Ng and Jordan has proofs and experiments

Good references

http://www.cs.cmu.edu/~tom/NewChapters.html

- http://ai.stanford.edu/~ang/papers/nips01-discriminativegenerative.pdf
- https://medium.com/@sangha_deb/naive-bayes-vs-logistic-regression-a319b07a5d4c

In our next session

We will cover:

Linear basis function models

Bias-variance decomposition

Bayesian linear regression