统计机器学习

（小班研讨）

主讲教师： 刘峤

第3章 分类算法

An introduction to Classification

Classification

- The goal of classification is to learn to predict a **categorical** outcome from input
  - we have examples of inputs and outcomes : $(\mathbf{x}_i, y_i)$
  - where the input is **usually continuous** features :$\mathbf{x} \in R^m$
  - the outcome y is now **discrete**: $y \in \{1, ..., K\}$
- In binary classification
  - we often call one class **positive** and the other **negative**
  - and encode the outcomes as : $y \in \pm 1$
- Typical classification algorithms
  - **Naive Bayes**, **logistic regression**, support vector machines, and boosting ……

Classification

- Abstractly, a classification algorithm takes as input
  - a set of **n** i.i.d. examples : $D = \{\mathbf{x}_i, y_i\}$
  - where $\mathbf{x} \in \mathcal{X}$ and $y \in \mathcal{Y}$ from an unknown distribution $$P(\mathbf{x}, y)$$
  - and produces a classifier: $h(\mathbf{x}) : \mathcal{X} \mapsto \mathcal{Y}$
  - from a family of possible classifiers
    - called a **hypothesis space**, often denoted as : $\mathcal{H}$
- The goal of a classification algorithm is usually to minimize expected classification error of h(x):

$$\mathbf{E}_{(\mathbf{x},y)}[\mathbf{1}(h(\mathbf{x}) \neq y)]$$

Generative vs.
Discriminative Approaches

Generative approach

- Generative classifiers focus on modeling $P(\mathbf{x}, y)$
- The idea behind generative approaches is
  - estimate $\hat{P}(\mathbf{x}, y)$ from examples (say using MLE):
  - then produce a **classifier**

$$h(\mathbf{x}) = \arg\max_y \hat{P}(y \mid \mathbf{x}) = \arg\max_y \frac{\hat{P}(\mathbf{x},y)}{\sum_{y'} \hat{P}(\mathbf{x},y')}$$

- For binary classification, this reduces to:

$$h(\mathbf{x}) = \begin{cases} 1 & \text{if} \quad \hat{P}(\mathbf{x}, 1) \geq \hat{P}(\mathbf{x}, -1) \\ -1 & \text{otherwise} \end{cases}$$

Bayes optimal

- If we learn a perfect model of the data

$$\hat{P}(\mathbf{x}, y) = P(\mathbf{x}, y)$$

- such a classifier is called **Bayes optimal** in a sense that no other classifier can have lower expected classification error.
- However P(x,y) is usually very complex
  - think of trying to model a distribution over images
- The **Naive Bayes** model makes a very simple approximation to P(x,y) that nevertheless often works well in practice.

Discriminative approach

- Discriminative classifiers focus on modeling $P(y \mid \mathbf{x})$
  - Since the joint distribution P(x,y) could be very complex
  - especially in parts of the space where P(x) is large
- discriminative methods:
  - Logistic regression models P(y|x) as **log-linear** and then attempts to estimate parameters w using MLE or MAP.
  - SVMs and boosting focus directly on learning a function h(x) that **minimizes expected classification error**, **without any probabilistic assumptions about P(x,y)**.

log-linear Model

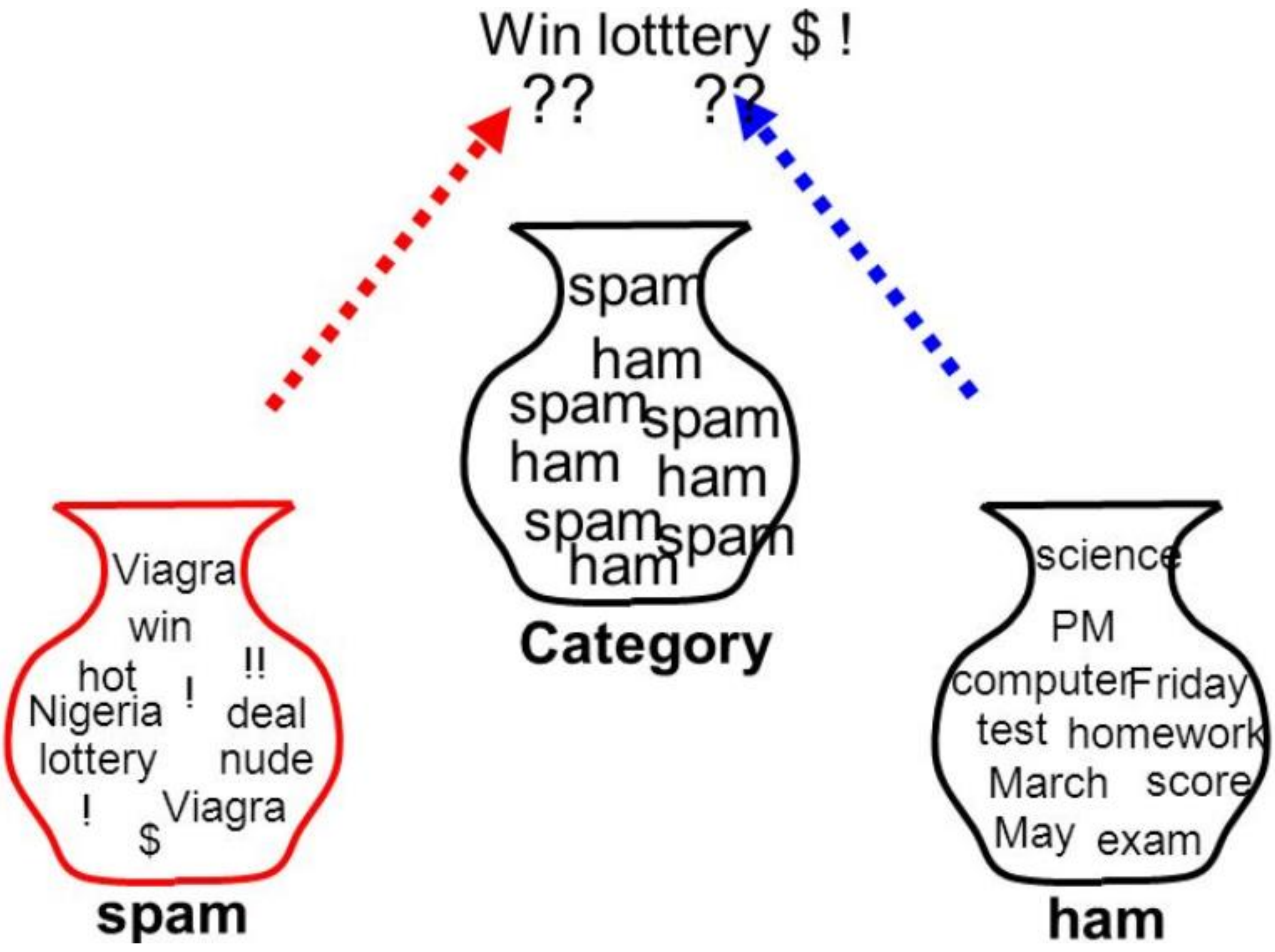
A log-linear model is a mathematical model that takes the form of a function whose logarithm equals a linear combination of the parameters of the model, which makes it possible to apply (possibly multivariate) linear regression. That is, it has the general form:

$$\exp\left(c + \sum_i w_i f_i(X)\right)$$

in which the $f_i(X)$ are quantities that are functions of the variables X, in general a vector of values, while c and the w_i stand for the model parameters.

Naive Bayes

Example : Naive Bayes Text classification



Example : Text classification

- bag-of-words representation
  - text documents are very complex, structured object
  - BOW representation of the input suffices for simple tasks
  - which completely disregard the order of the words in the document and just consider their counts
- The Naive Bayes classifier then learns
  - for each word in our dictionary by using MLE/MAP

$$\hat{P}(spam),\ \hat{P}(word \mid spam) \text{ and } \hat{\mathrm{P}}(\mathrm{word} \mid \mathrm{ham})$$

  - It then predicts prediction spam if:

$$\hat{P}(spam) \prod_{\text{word}\in\text{email}} \hat{P}(word \mid spam) > \hat{P}(ham) \prod_{\text{word}\in\text{email}} \hat{P}(word \mid ham)$$

Naive Bayes Model

- The NB classifier is generative model: we will model P(x,y)
- Consider the toy transportation data below:

x: Inputs/Features/Attributes			y: Class
Distance(miles)	**Raining**	**Flat Tire**	**Mode**
1	no	no	bike
2	yes	no	walk
1	no	yes	bus
1	yes	no	walk
2	yes	no	bus
1	no	no	car
1	yes	yes	bike
10	yes	no	bike
10	no	no	car
4	no	no	bike

Naive Bayes Model

- We will decompose P(x,y) into **class prior** and **class model**:

$$P(\mathbf{x}, y) = \underbrace{P(y)}_{\text{class prior}} \underbrace{P(\mathbf{x} \mid y)}_{\text{class model}}$$

- Then estimate them separately as $\hat{P}(y)$ and $\hat{P}(\mathbf{x} \mid y)$
  - **class prior** should not be confused with **parameter prior**.
- use our estimates to output a classifier using Bayes rule:

$$\begin{aligned} h(\mathbf{x}) &= \arg\max_{y} \hat{P}(y \mid \mathbf{x}) \\ &= \arg\max_{y} \frac{\hat{P}(y)\hat{P}(\mathbf{x}|y)}{\sum_{y'} \hat{P}(y')\hat{P}(\mathbf{x}|y')} \\ &= \arg\max_{y} \hat{P}(y)\hat{P}(\mathbf{x} \mid y) \end{aligned}$$

Naive Bayes Model

- To estimate our model using **MLE**
  - we can **separately** estimate the two parts of the model:

$$
\begin{aligned}
\log P(D) &= \sum_i \log P(\mathbf{x}_i, y_i) \\
&= \sum_i \log P(y_i) + \log P(\mathbf{x}_i \mid y_i) \\
&= \log P(D_Y) + \log P(D_X | D_Y)
\end{aligned}
$$

Naive Bayes Model

- How do we estimate P(y)?

x: Inputs/Features/Attributes			y: Class
Distance(miles)	**Raining**	**Flat Tire**	**Mode**
1	no	no	bike
2	yes	no	walk
1	no	yes	bus
1	yes	no	walk
2	yes	no	bus
1	no	no	car
1	yes	yes	bike
10	yes	no	bike
10	no	no	car
4	no	no	bike

Bernoulli distribution

- **伯努利试验是只有两种可能结果的单次随机试验，即对于一个随机变量X而言：**

$$P(X=1)=p,\ P(X=0)=1-p$$

- **进行一次伯努利试验**
  - **成功(X=1)概率为p(0<=p<=1)，失败(X=0)概率为1-p**
  - **则称随机变量X服从伯努利分布**
  - **伯努利分布是离散型概率分布，其概率密度函数为：**

$$f(x)=p^x(1-p)^{1-x}=\begin{cases} p & \text{if } x=1 \\ 1-p & \text{if } x=0 \\ 0 & \text{otherwise} \end{cases}$$

Binomial distribution

- **如果E是一个伯努利试验，将E独立重复地进行n次，则称这一组重复的独立试验为n重伯努利试验。二项分布是n重伯努利试验成功次数的离散概率分布。**
- **二项分布是离散型概率分布，其概率密度函数为：**

$$P\{X=k\}=C_n^k p^k(1-p)^{n-k},\ k=0,1,2,\ldots n$$

- **显然：**

$$\sum_{k=0}^{n} P\{X=k\}=\sum_{k=0}^{n} C_n^k p^k(1-p)^{n-k}=\{p+(1-p)\}^n=1$$

- **二项分布名称的由来，是由于其概率密度函数中使用了二项系数**

$$(x+y)^n=C_n^k x^k y^{n-k}$$

Multinomial Distribution

- **多项式分布是二项式分布的推广。**
- **如果n次试验，每次试验的结果可以有m个，且m个结果发生的概率互斥且和为1，则发生其中一个结果发生X次的概率就是多项式分布。**
- **多项式分布的概率密度函数为：**

$$P(X_1 = x_1, \cdots, X_k = x_k) = \begin{cases} \frac{n!}{x_1!,\cdots,x_k!} p^{x_1} \cdots p^{x_k} & \text{when } \sum_{i=1}^k x_i = n \\ 0 & \text{otherwise.} \end{cases}$$

Estimating P(y)

- How do we estimate P(y)?

y	parameter θ_y	MLE $\hat{\theta}_y$
walk	θ_{walk}	0.2
bike	θ_{bike}	0.4
bus	θ_{bus}	0.2
car	θ_{car}	0.2

- We need 4 parameters to represent this **multinomial** distribution
  - 3 really, since they must sum to 1
- The MLE estimate is

$$\hat{\theta}_y = \frac{1}{n} \sum_i \mathbf{1}(y = y_i)$$

Estimating P(x|y)

- Complexity of the problem:
  - Suppose that all the features are binary
  - If we have m features
  - there are $K * 2^m$ possible values of (x,y)
  - we cannot store or estimate such a distribution explicitly
- The key (**naive**) assumption of the model is
  - **conditional independence** of the **features given the class**
  - Recall that X_k is conditionally independent of X_j given Y if:

$$P(X_j = x_j \mid X_k = x_k, Y = y) = P(X_j = x_j \mid Y = y), \ \forall x_j, x_k, y$$

$$P(X_j = x_j, X_k = x_k \mid Y = y) = P(X_j = x_j \mid Y = y) P(X_k = x_k \mid Y = y)$$

Estimating P(x|y)

- More generally, the Naive Bayes assumption is that:

$$\hat{P}(\mathbf{X} \mid Y) = \prod_j \hat{P}(X_j \mid Y)$$

- Hence the Naive Bayes classifier is simply:

$$\arg\max_y \hat{P}(Y = y \mid \mathbf{X}) = \arg\max_y \hat{P}(Y = y) \prod_j \hat{P}(X_j \mid Y = y)$$

- If the feature X_j is **discrete** (like Raining)
  - then we need to estimate K distributions for it
  - one for each class $P(X_j|Y = k)$
  - We have 4 parameters,

$$\theta_{\mathrm{R|walk}}, \theta_{\mathrm{R|bike}}, \theta_{\mathrm{R|bus}}, \theta_{\mathrm{R|car}}$$

Estimating P(x|y)

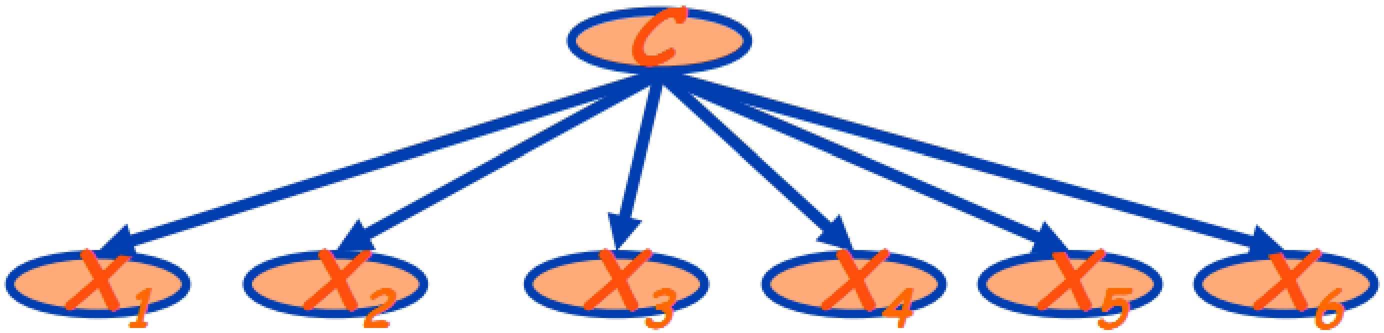
- The MLE estimate is

$$\hat{\theta}_{\mathrm{R}|y} = \frac{\sum_i \mathbf{1}(R{=}yes, y{=}y_i)}{\sum_i \mathbf{1}(y{=}y_i)}$$

- For example, P(R | Y) is

y	parameter $\theta_{R\|y}$	MLE $\hat{\theta}_{R\|y}$
walk	$\theta_{R\|walk}$	1.0
bike	$\theta_{R\|bike}$	0.5
bus	$\theta_{R\|bus}$	0.5
car	$\theta_{R\|car}$	**0.0**

Learning the Model

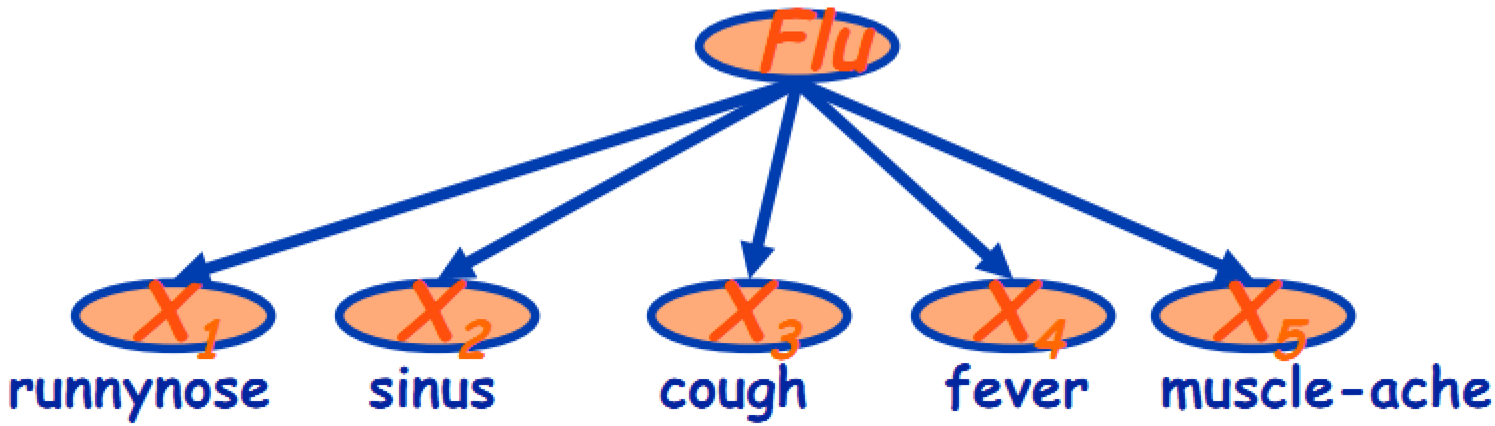


- **maximum likelihood estimates**
  - **simply use the frequencies in the data**

$$\hat{P}(c_j) = \frac{N(C = c_j)}{N}$$

$$\hat{P}(x_i \mid c_j) = \frac{N(X_i = x_i, C = c_j)}{N(C = c_j)}$$

Problem with Max Likelihood



- What if we have seen no training cases where patient had no flu and muscle aches?

$$\hat{P}(X_5 = t \mid C = flu) = \frac{N(X_5 = t, C = flu)}{N(C = flu)} = 0$$

- Zero probabilities cannot be conditioned away
  - no matter the other evidence!

$$\ell = \arg\max_c \hat{P}(c) \prod_i \hat{P}(x_i \mid c)$$

MLE vs. MAP

- Note the **danger** of using MLE estimates.
  - consider the estimate of conditional distribution of Raining=yes:

$$\hat{P}(\text{Raining} = \text{yes} \mid y = \text{car}) = 0$$

x: Inputs/Features/Attributes			y: Class
Distance(miles)	**Raining**	**Flat Tire**	**Mode**
1	no	no	bike
2	yes	no	walk
1	no	yes	bus
1	yes	no	walk
2	yes	no	bus
1	no	no	car
1	yes	yes	bike
10	yes	no	bike
10	no	no	car
4	no	no	bike

y	parameter $\theta_{R\|y}$	MLE $\hat{\theta}_{R\|y}$
walk	$\theta_{R\|walk}$	1.0
bike	$\theta_{R\|bike}$	0.5
bus	$\theta_{R\|bus}$	0.5
car	$\theta_{R\|car}$	**0.0**

MLE vs. MAP

- Note the **danger** of using MLE estimates.
  - consider the estimate of conditional distribution of Raining=yes:

$$\hat{P}(\text{Raining} = \text{yes} \mid y = \text{car}) = 0$$

  - So if we know it's raining, no matter the distance, the probability of taking the car is 0, which is not a good estimate.
- This is a general problem due to scarcity of data:
  - we never saw an example with car and raining.
  - Using **MAP** estimation with **Beta priors** (with $\alpha,\beta>1$), estimates will never be zero, since additional counts are added.

Smoothing to Avoid Overfitting

$$\hat{P}(x_i \mid c_j) = \frac{N(X_i = x_i, C = c_j) + 1}{N(C = c_j) + v}$$

Text Classification

Using Naive Bayes Classifiers to Classify Text: Bag of Words

- General model: Features are positions in the text (X_1 is first word, X_2 is second word, …), values are words in the vocabulary

$$c_{NB} = \operatorname*{argmax}_{c_j \in C} P(c_j) \prod_i P(x_i \mid c_j)$$

$$= \operatorname*{argmax}_{c_j \in C} P(c_j) P(x_1 = \text{"our"} \mid c_j) \cdots P(x_n = \text{"text"} \mid c_j)$$

- **Too many possibilities, so assume that classification is *independent* of the positions of the words**
  - Result is *bag of words* model
  - Just use the counts of words, or even a variable for each word: is it in the document or not?

Naïve Bayes: Learning

- From training corpus, determine *Vocabulary*

- Estimate $P(c_j)$ and $P(x_k \mid c_j)$
  - For each c_j in C do

    $docs_j \leftarrow$ documents labeled with class c_j

    $$P(c_j) \leftarrow \frac{|docs_j|}{|\text{total \# documents}|}$$

  - For each word x_k in *Vocabulary*

    $n_k \leftarrow$ number of occurrences of x_k in all $docs_j$

    $$P(x_k \mid c_j) \leftarrow \frac{n_k + 1}{|docs_j| + |Vocabulary|}$$

Simple "Laplace" smoothing

Text Classification

Naïve Bayes: Classifying

- For all words x_i in current document
- Return c_{NB}, where

$$c_{NB} = \underset{c_j \in C}{\operatorname{argmax}} P(c_j) \prod_{i \in documant} P(x_i \mid c_j)$$

What is the implicit assumption hidden in this?

- The correct model would have a probability for each word observed and one for each word not observed. --- Naïve Bayes for text assumes that there is no information in words that are not observed – since most words are very rare, their probability of not being seen is close to 1.

Beta distribution

- 在贝叶斯推断中，Beta 分布是二项式和几何分布的共轭先验概率分布。
  - 共轭分布(conjugacy)：后验概率分布函数与先验概率分布函数具有相同形式。
- Beta 分布定义在区间[0,1]上，包含两个形状参数（$\alpha \geq 1, \beta \geq 1$）
  - 其概率密度函数定义为：

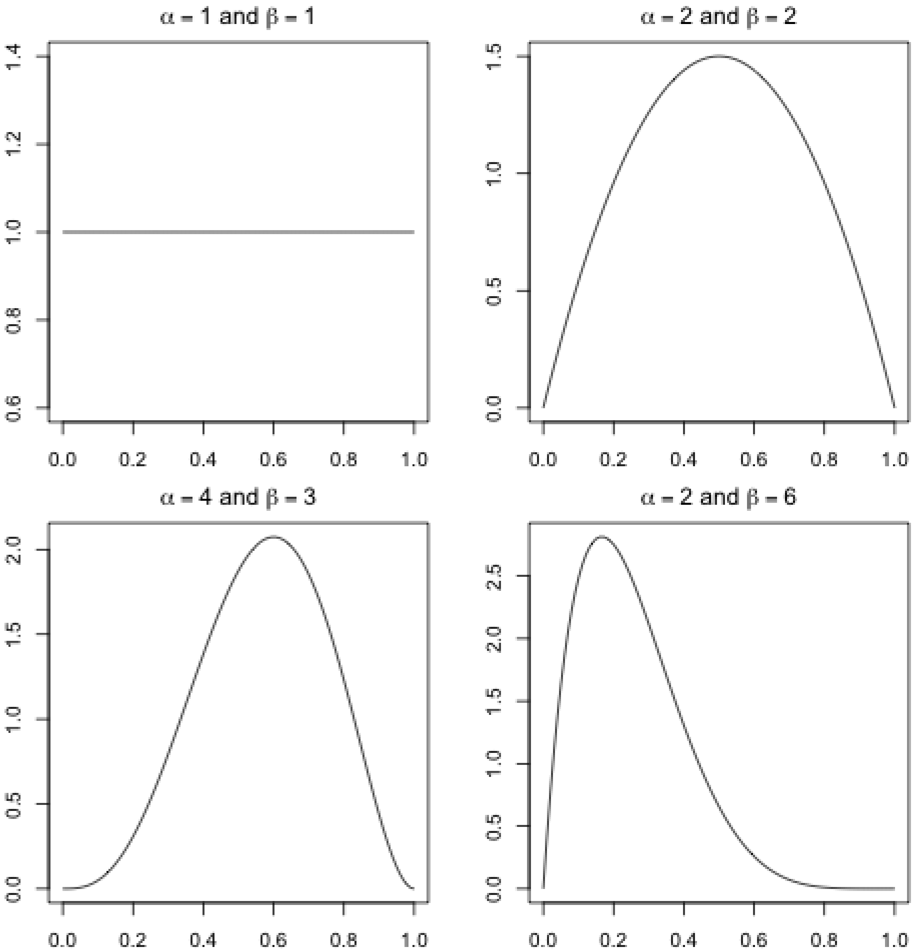
$$Beta(\alpha,\beta): \quad P(\theta \mid \alpha,\beta) = \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)}\theta^{\alpha-1}(1-\theta)^{\beta-1}$$

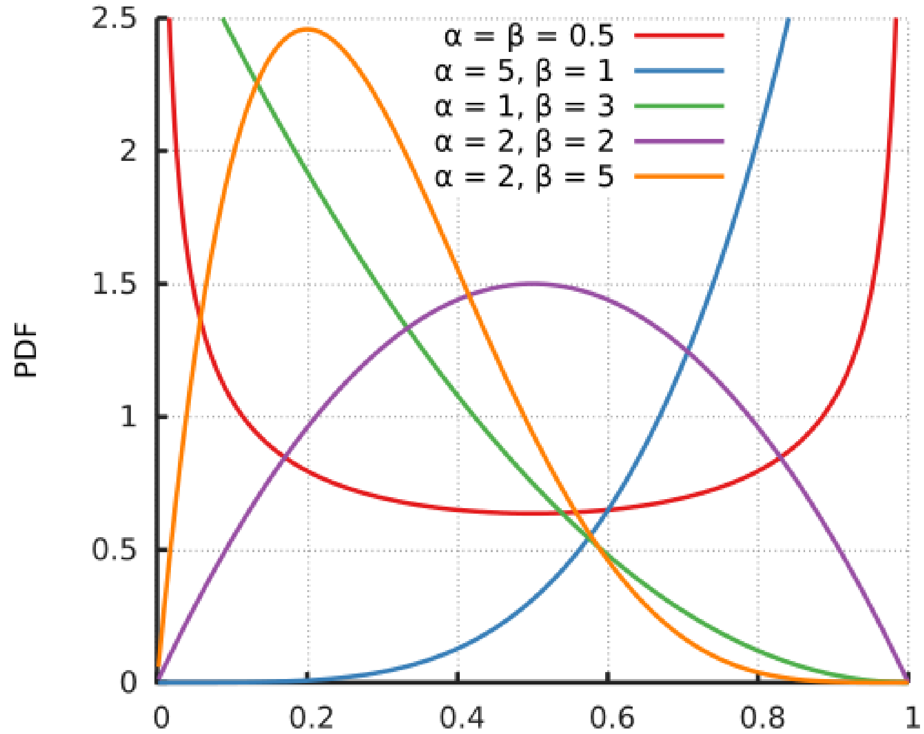
- where Γ is the gamma function, for positive integers n

$$\Gamma(n) = (n-1)!$$

- The hyperparameters
  - their sum controls peakiness and
  - their ratio controls the left-right bias

Beta distribution





The Bayesian way

- In coin flipping problem, when you don't have enough data
  - you rely on your prior knowledge that most coins are pretty fair to estimate the parameter
  - If you're a Bayesian, you will embrace uncertainty but quantify it, by assuming that your prior belief about coin fairness can be described with a distribution $P(\theta)$.
  - The Beta distribution is a very convenient class of priors
- Question: what do we do with the prior?
  - The Bayesian framework uses data to update this prior distribution over the parameters.
  - Using Bayes rule, we obtain a posterior over parameters:

$$P(\theta|D) = \frac{P(D|\theta)P(\theta)}{P(D)} \propto P(D|\theta)P(\theta)$$

Beta distribution

- **Why is Beta so convenient?**

$$P(\theta|D) \propto P(D|\theta)P(\theta) \propto \theta^{n_H+\alpha-1}(1-\theta)^{n_T+\beta-1}$$

- Hence,

$$P(\theta|D) = Beta(n_H+\alpha; n_T+\beta)$$

- This property of fit between a model and its prior is called **conjugacy**, which essentially means the posterior is of the same distributional family as the prior.

- For the Beta(α;β) prior, the MAP estimate is:

$$\hat{\theta}_{MAP} = \arg\max_\theta P(\theta \mid D) = \arg\max_\theta \left(\log P(D \mid \theta) + \log P(\theta)\right)$$

$$\hat{\theta}_{MAP} = \frac{n_H+\alpha-1}{n_H+n_T+\alpha+\beta-2}$$

Naive Bayes for Continuous Inputs
Gaussian Naive Bayes

Gaussian Naive Bayes

- How about the MLE estimate of **continuous** variable like Distance?
  - there are many possible choices of models, with **Gaussian** being the simplest
  - We need to estimate K distributions for each feature, one for each class

$$P(X_j|Y=k)$$

- For example, P(D|Y) is

y	parameter $\mu_{D|y}$ and $\sigma_{D|y}$	MLE $\hat{\mu}_{D|y}$	MLE $\hat{\sigma}_{D|y}$
walk	$\mu_{D|walk}$, $\sigma_{D|walk}$	1.5	0.5
bike	$\mu_{D|bike}$, $\sigma_{D|bike}$	4.0	3.7
bus	$\mu_{D|bus}$, $\sigma_{D|bus}$	1.5	0.5
car	$\mu_{D|car}$, $\sigma_{D|car}$	5.5	4.5

Gaussian Naive Bayes

- In order to train such a Naive Bayes classifier we must therefore estimate the mean and standard deviation of each of these Gaussians for each attribute Xi and each possible value yk of Y .

$$\mu_{ik} = E[X_i | Y = y_k]$$
$$\sigma_{ik}^2 = E[(X_i - \mu_{ik})^2 | Y = y_k]$$

- Note there are 2^nK of these parameters, all of which must be estimated independently

- Of course we must also estimate the priors on Y as well

$$\pi_k = P(Y = y_k)$$

Gaussian Naive Bayes

- The maximum likelihood estimator for μ_{ik} is

$$\hat{\mu}_{ik} = \frac{1}{\sum_j \delta(Y^j = y_k)} \sum_j X_i^j \delta(Y^j = y_k)$$

- The maximum likelihood estimator for σ^2_{ik} is

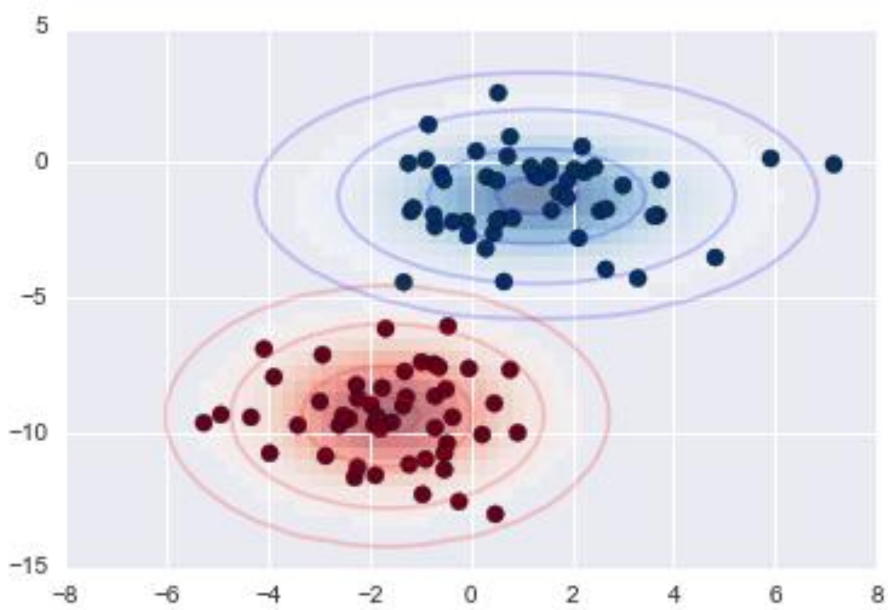
$$\hat{\sigma}^2_{ik} = \frac{1}{\sum_j \delta(Y^j = y_k)} \sum_j (X_i^j - \hat{\mu}_{ik})^2 \delta(Y^j = y_k)$$

- This maximum likelihood estimator is biased, so the m
- d estimator (MVUE) is sometimes used instead. It is

$$\hat{\sigma}^2_{ik} = \frac{1}{(\sum_j \delta(Y^j = y_k)) - 1} \sum_j (X_i^j - \hat{\mu}_{ik})^2 \delta(Y^j = y_k)$$

Gaussian Naive Bayes

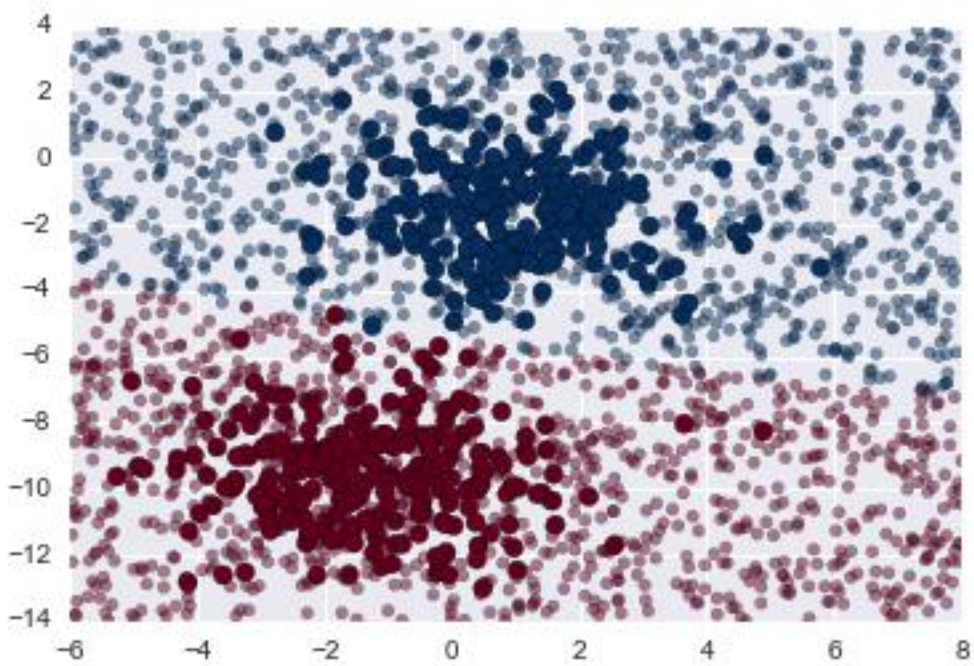
- Assumption of the Gaussian naive Bayes classifier
  - data from each label is drawn from a simple Gaussian distribution
  - Suppose our data is from two classes, each described by a Gaussian distribution with no covariance between dimensions.



The ellipses here represent the Gaussian generative model for each label, with larger probability toward the center of the ellipses.

Gaussian Naive Bayes

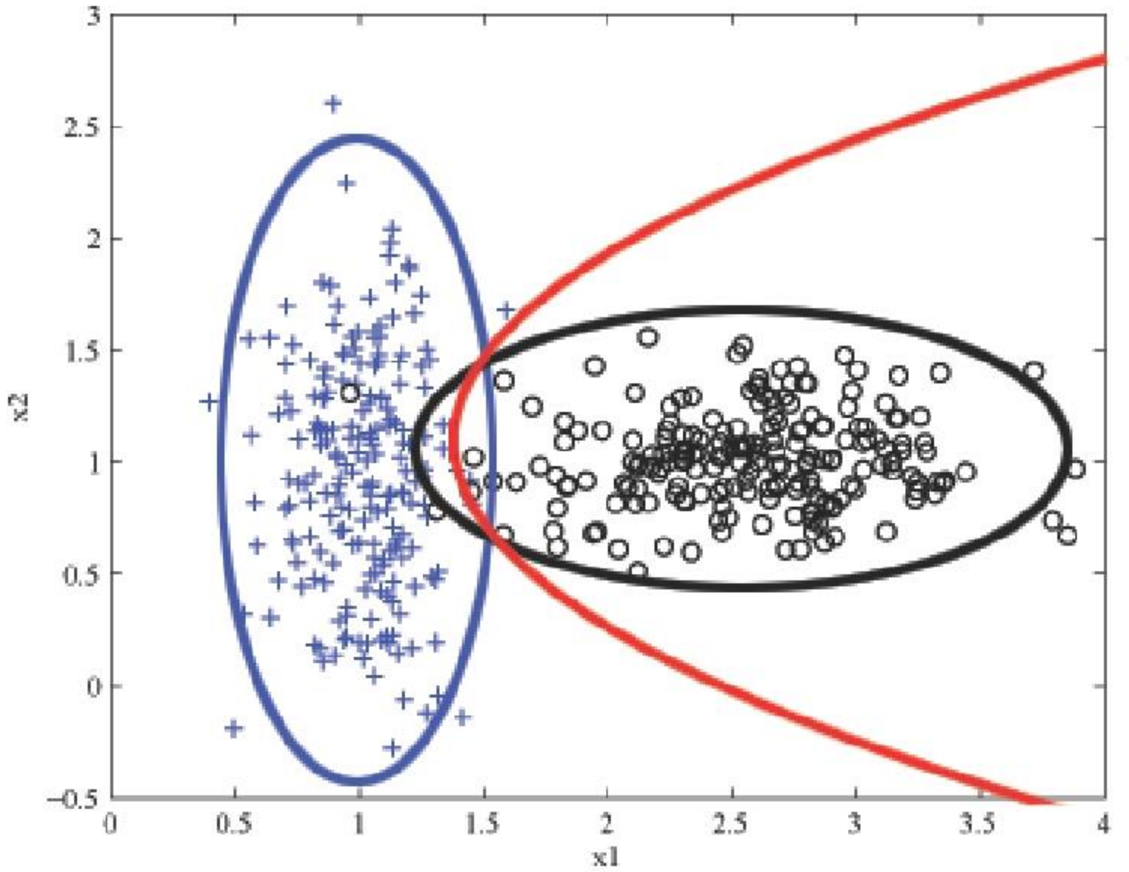
- Decision boundary of the Gaussian naive Bayes classifier
  - The Naive Bayes classifier will estimate a Gaussian for each class and each dimension.



we see a slightly curved boundary in the classifications

Gaussian Naive Bayes

- In general, the boundary in Gaussian naive Bayes is quadratic.



2D binary classification with Naive Bayes. A density contour is drawn for the Gaussian model of each class and the decision boundary is shown in red

More Facts About Bayes Classifiers

Naive Bayes is not so dumb

- A good baseline for text classification
- Optimal if the Independence Assumptions hold:
- Very Fast:
  - Learn with one pass over the data
  - Testing linear in the number of attributes and of documents
  - Low Storage requirements

Naive Bayes is not so dumb

- Naïve Bayes is wonderfully cheap
  - And handles 1,000,000 features cheerfully!
- Bayes Classifiers don't try to be maximally discriminative
  - They merely try to honestly model what's going on
- Zero probabilities give stupid results
- Bayes Classifiers can be built with real-valued inputs
  - Or many other distributions

Logistic Regression

Recap : Naive Bayes

- Naive Bayes assumes some functional form for

$$\hat{P}(X|Y) \quad \text{and} \quad \hat{P}(Y)$$

  – and estimates parameters of P from training data

- Bayes rule:

$$\hat{P}(Y|X=x) = \frac{\hat{P}(X=x|Y)\hat{P}(Y)}{\hat{P}(X=x)}$$

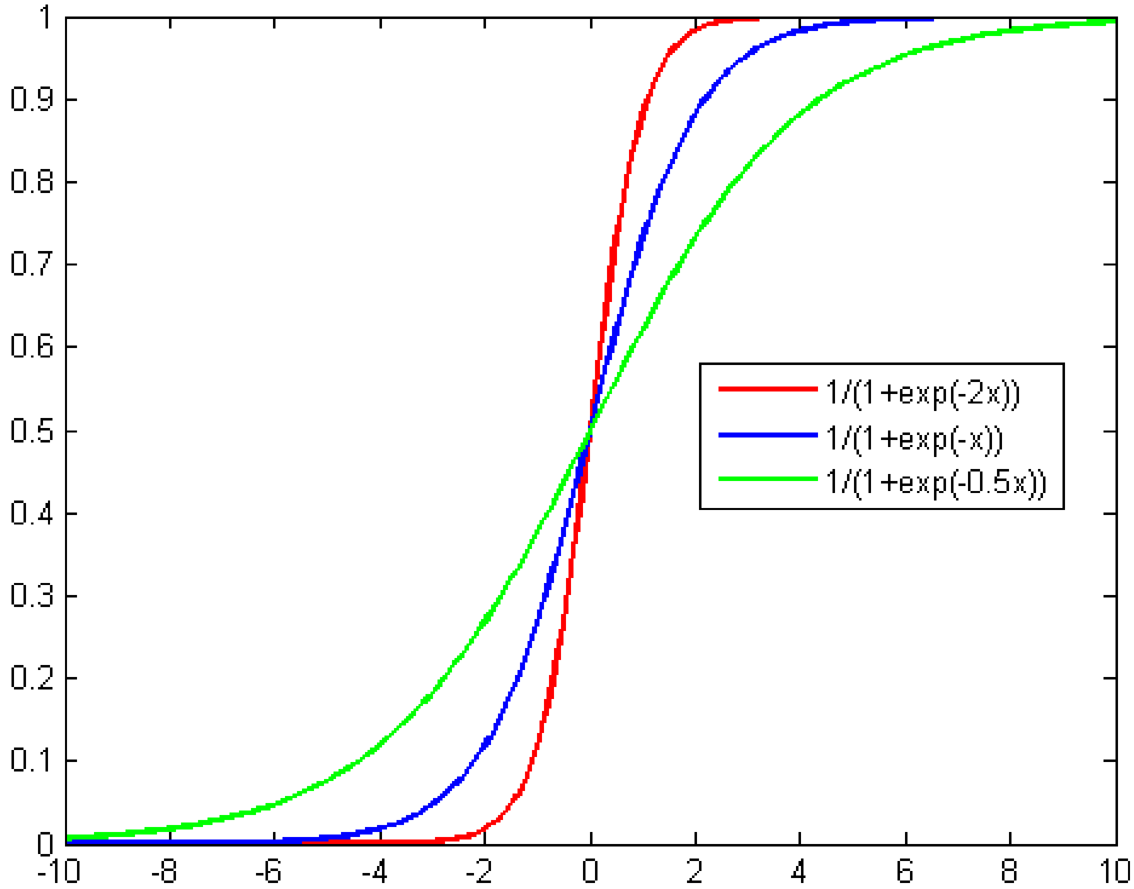
- Note: in generative models, the computation of P(Y|X) is always indirect, through Bayes rule.

- While discriminative classifiers, such as Logistic Regression, instead assume some functional form for P(Y|X) and estimate parameters of P(Y|X) from training data.

(simple) binary classification

- The model uses the **logistic** (**sigmoid**) function:

$$f(x) = 1/(1 + e^{-x})$$



Binary Classification

- Logistic regression for binary classification $y \in \{-1, 1\}$

$$P(Y = 1|\mathbf{x}, \mathbf{w}) = \frac{1}{1+\exp\{-\mathbf{w}^\top \mathbf{x}\}} = \frac{1}{1+\exp\{-y\mathbf{w}^\top \mathbf{x}\}}$$

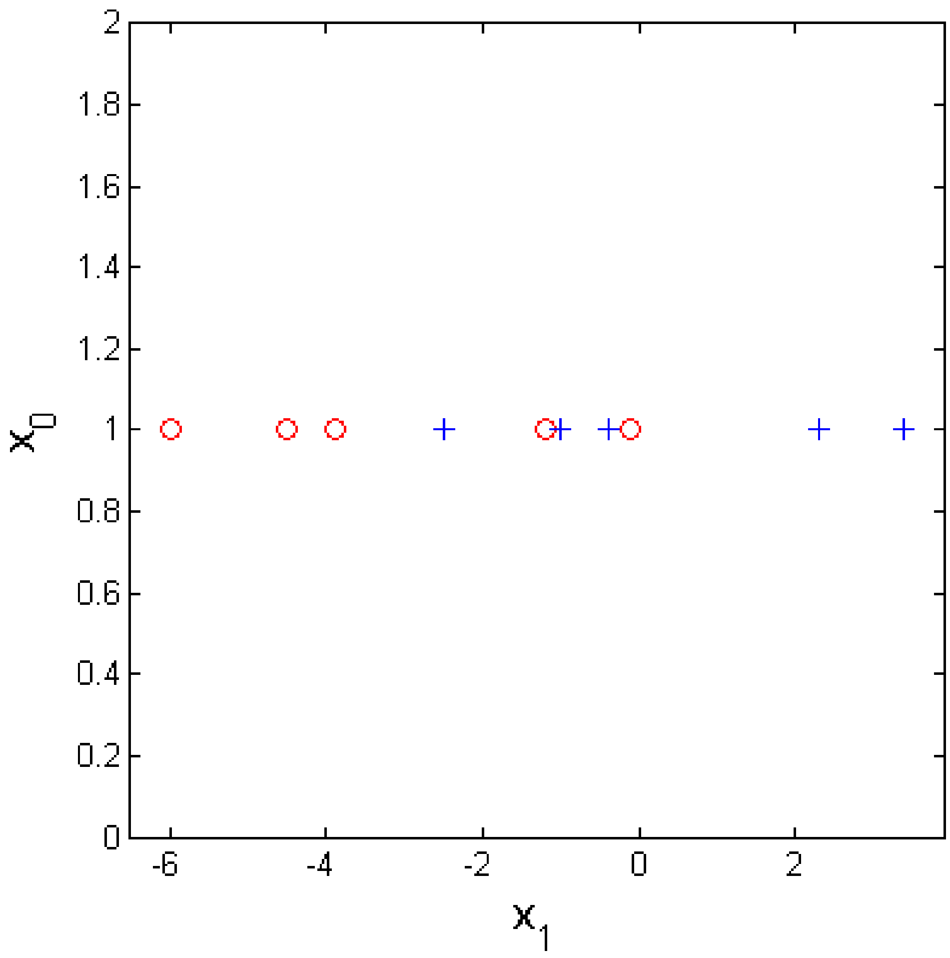
$$P(Y = -1|\mathbf{x}, \mathbf{w}) = 1 - P(Y = 1|\mathbf{x}, \mathbf{w})$$

$$= \frac{\exp\{-\mathbf{w}^\top \mathbf{x}\}}{1+\exp\{-\mathbf{w}^\top \mathbf{x}\}} = \frac{1}{1+\exp\{-y\mathbf{w}^\top \mathbf{x}\}}$$

$$log(\frac{P(Y=1|\mathbf{x},\mathbf{w})}{P(Y=-1|\mathbf{x},\mathbf{w})}) = \mathbf{w}^\top \mathbf{x}$$

1-D Example

- Consider a 1D problem, with positive class marked as pluses and negative as circles.



1-D Example

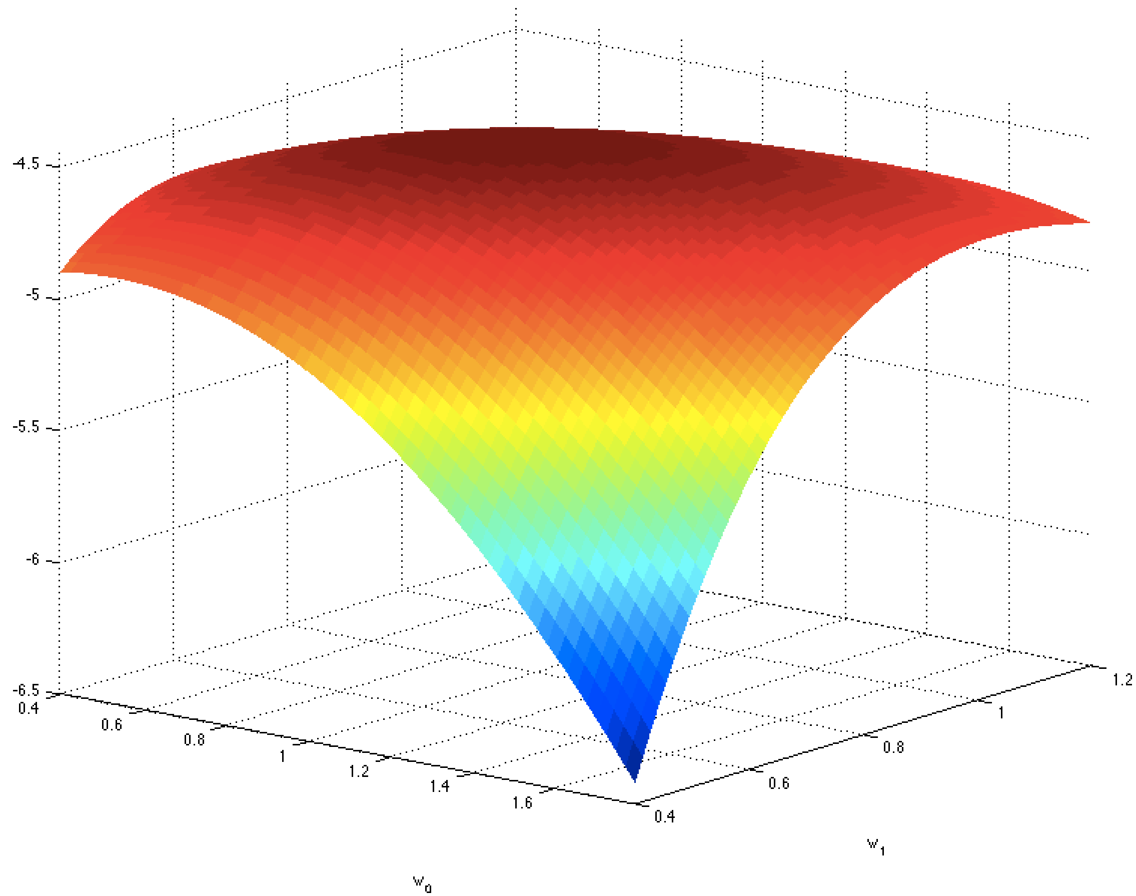
- The log likelihood of the data ℓ(w) is:

$$\log(P(D_Y|D_X, \mathbf{w})) = \log\left(\prod_i \frac{1}{1+\exp\{-y_i \mathbf{w}^\top \mathbf{x}_i\}}\right)$$

$$= -\sum_i \log(1 + \exp\{-y_i \mathbf{w}^\top \mathbf{x}_i\})$$

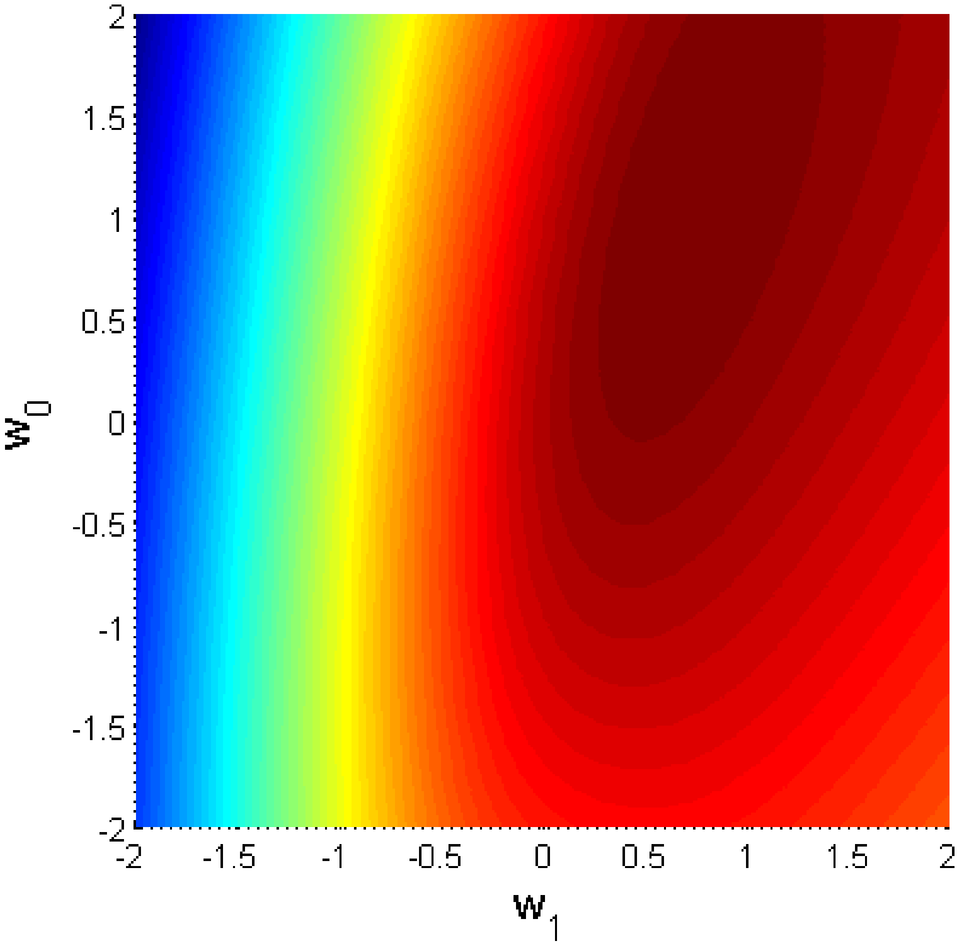
1-D Example

- This log likelihood function is shown below in the space of the two parameters w0 and w1, with the maximum at (w0=1, w1=0.7).



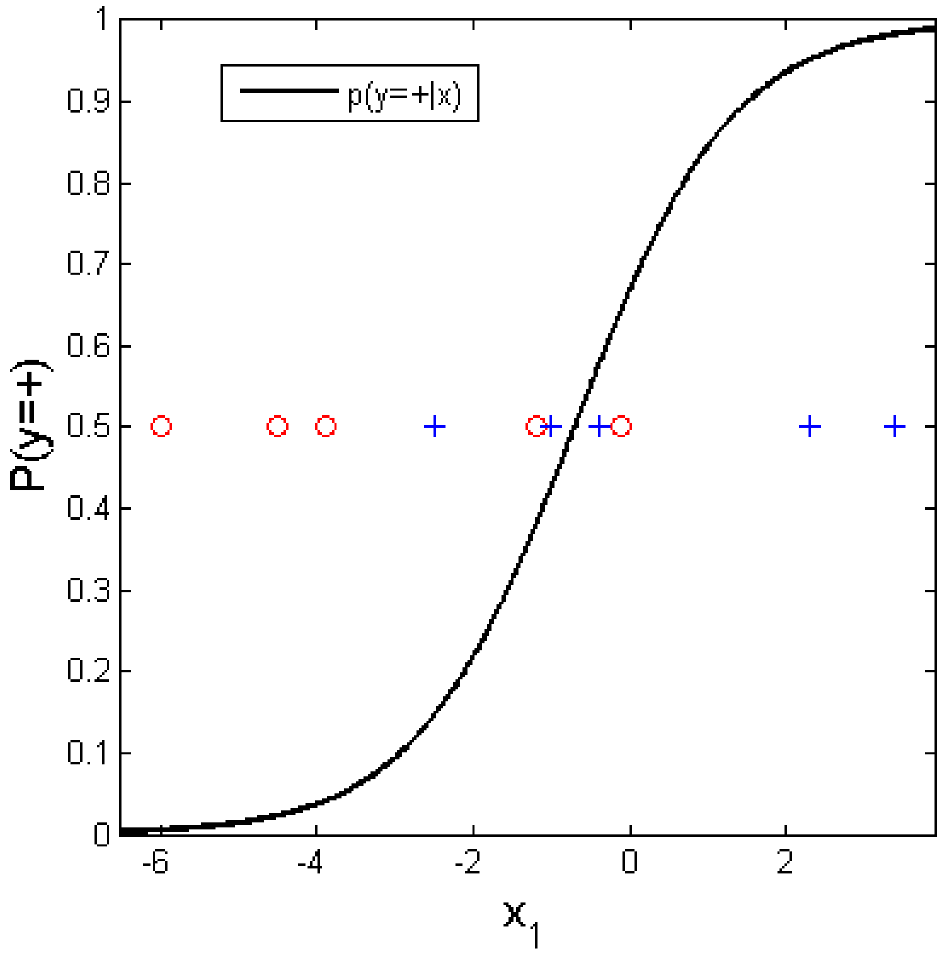
1-D Example

- heat map of the log likelihood function.



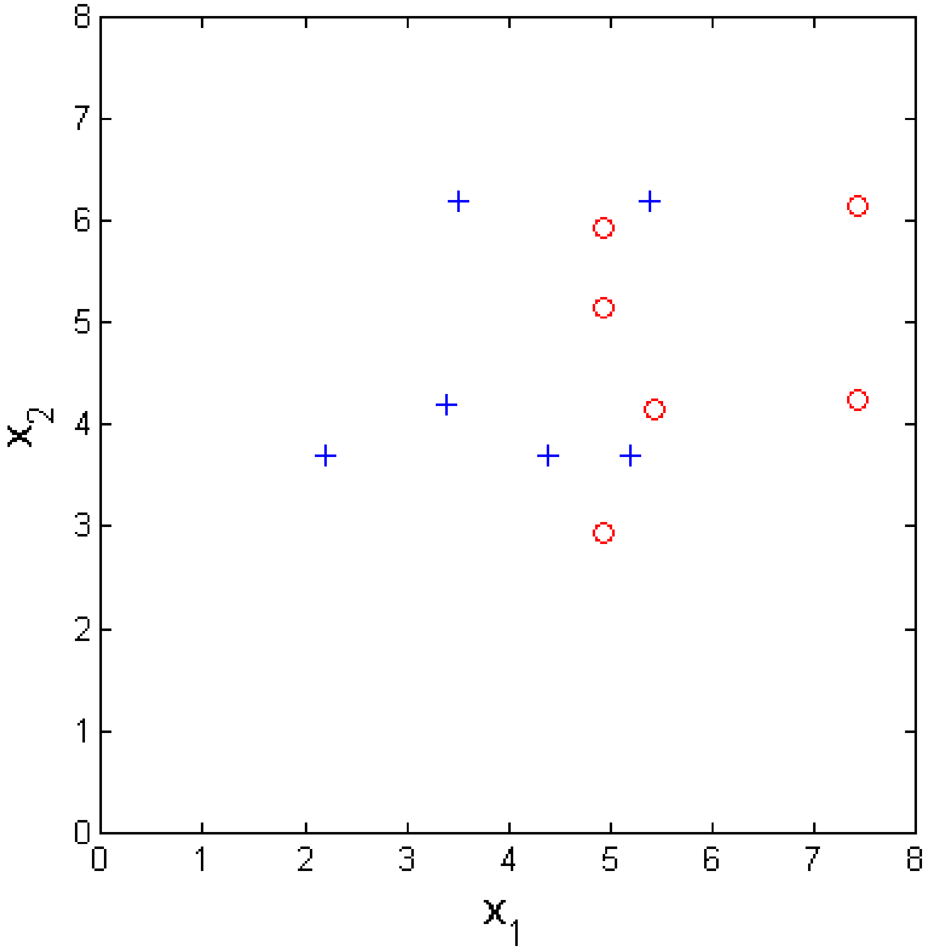
1-D Example

- The MLE solution is displayed below.



2-D Example

- Consider a 2D problem, with positive class marked as pluses and negative as circles.



2-D Example

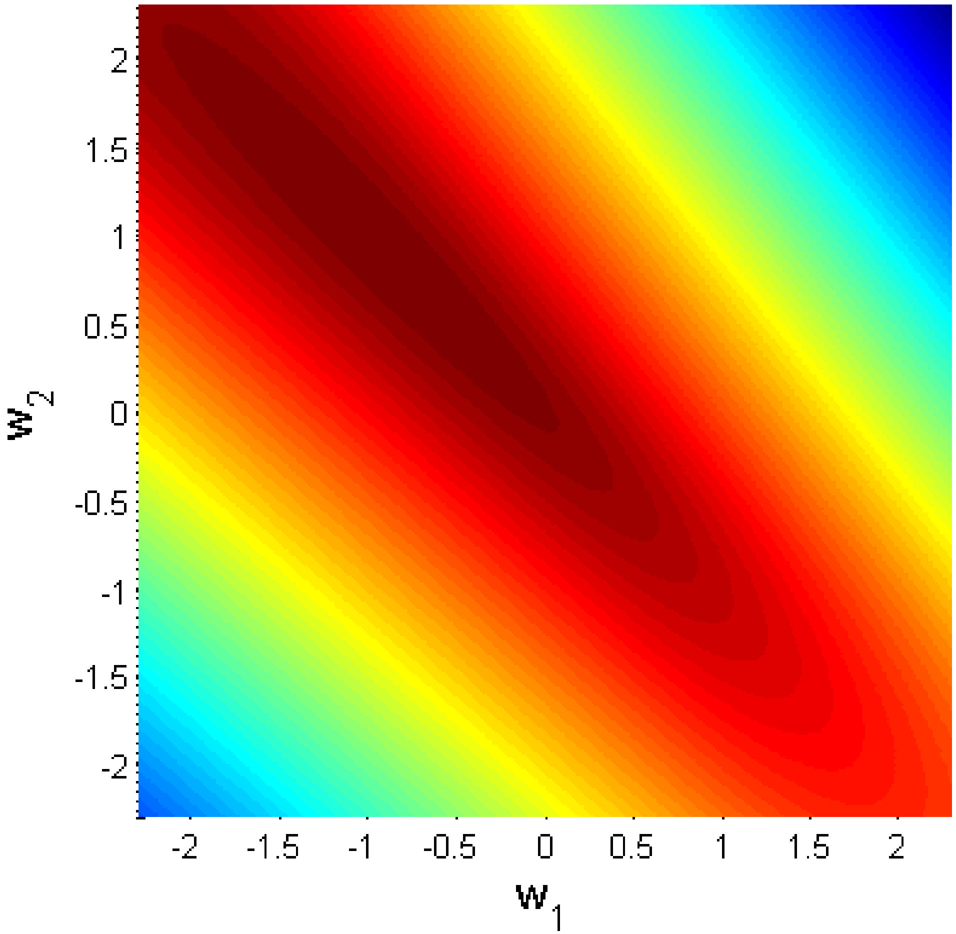
- The log likelihood of the data $\ell(w)$ is:

$$\log(P(D_Y|D_X,\mathbf{w})) = \log\left(\prod_i \frac{1}{1+\exp\{-y_i\mathbf{w}^\top\mathbf{x}_i\}}\right)$$

$$= -\sum_i \log(1+\exp\{-y_i\mathbf{w}^\top\mathbf{x}_i\})$$

2-D Example

- This log likelihood function is shown below in the space of the two parameters w1 and w2, with the maximum at (−0.81,0.81)



Linear Decision Boundary

- Why is the boundary linear?
  - Consider the condition that holds at the boundary:

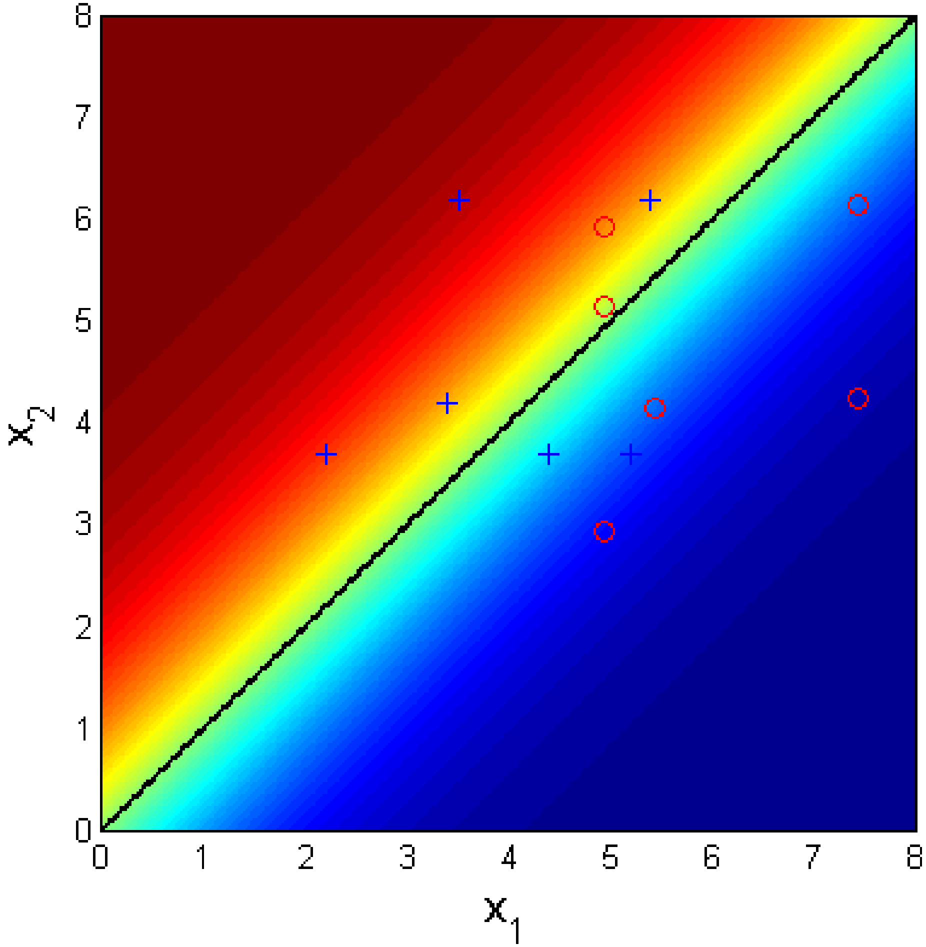
$$P(Y=1|\mathbf{x},\mathbf{w}) = P(Y=-1|\mathbf{x},\mathbf{w}) \;\rightarrow$$

$$\frac{1}{1+\exp\{-\mathbf{w}^\top \mathbf{x}\}} = \frac{\exp\{-\mathbf{w}^\top \mathbf{x}\}}{1+\exp\{-\mathbf{w}^\top \mathbf{x}\}} \quad \rightarrow \;\mathbf{w}^\top \mathbf{x} = 0$$

- For the toy problem above, the optimal w is (−0.81,0.81)
- so solving : $\mathbf{w}^\top \mathbf{x} = -0.81x_1 + 0.81x_2 = 0$
- we get the line : $x_1 = x_2$

2-D Example

- The MLE solution is displayed below, where the red color indicates high probability of positive class. The black line shows the decision boundary learned by MLE.



Computing MLE

- Unfortunately, solving for the parameters is not "**closed form**"
  - To find the MLE we will resort to iterative optimization.
  - Luckily, the function we're trying to optimize is convex, which means it has one global optimum (caveat: it may be at infinity).
- The simplest form of optimization is gradient ascent
  - start at some point and climb up in steepest direction
  - The gradient of our objective is given by:

$$\nabla_{\mathbf{w}} \ell(\mathbf{w}) = \frac{\partial \log(P(D_Y|D_X, \mathbf{w}))}{\partial \mathbf{w}} = \frac{\partial - \sum_i \log(1+\exp\{-y_i \mathbf{w}^\top \mathbf{x}_i\})}{\partial \mathbf{w}}$$

$$= \sum_i \frac{y_i \mathbf{x}_i \exp\{-y_i \mathbf{w}^\top \mathbf{x}_i\}}{1+\exp\{-y_i \mathbf{w}^\top \mathbf{x}_i\}} = \sum_i y_i \mathbf{x}_i (1 - P(y_i|\mathbf{x}_i, \mathbf{w}))$$

Computing MLE

- Gradient ascent update is

$$\mathbf{w}^{t+1} = \mathbf{w}^t + \eta_t \nabla_{\mathbf{w}} \ell(\mathbf{w})$$

- Where $\eta_t > 0$ is the update rate.

- iterate until change is in parameters is smaller than some tolerance

Computing MAP

- we can assume a prior on the parameters w
  - We will pick the simplest zero-mean Gaussian with a standard deviation λ for every parameter:

$$w_j \sim \mathcal{N}(0, \lambda^2) \;\; \text{so} \;\; P(\mathbf{w}) = \prod_j \frac{1}{\lambda\sqrt{2\pi}} \exp\left\{\frac{-w_j^2}{2\lambda^2}\right\}$$

- So to get the MAP we need:

$$\arg\max_{\mathbf{w}} \log P(\mathbf{w} \mid D, \lambda) = \arg\max_{\mathbf{w}} \left(\ell(\mathbf{w}) + \log P(\mathbf{w} \mid \lambda)\right)$$

$$= \arg\max_{\mathbf{w}} \left(\ell(\mathbf{w}) - \frac{1}{2\lambda^2}\mathbf{w}^\top \mathbf{w}\right)$$

Computing MAP

- Recall that

$$\nabla_{\mathbf{w}} \ell(\mathbf{w}) = \sum_i y_i \mathbf{x}_i (1 - P(y_i | \mathbf{x}_i, \mathbf{w}))$$

- The gradient of the MAP objective is given by:

$$\nabla_{\mathbf{w}} \, \log P(\mathbf{w} \mid D, \lambda) = \sum_i y_i \mathbf{x}_i (1 - P(y_i | \mathbf{x}_i, \mathbf{w})) - \frac{1}{\lambda^2} \mathbf{w}$$

- Note the effect of the prior: it penalizes large values of w.

Multinomial Logistic Regression

- Generalizing to the case when the outcome has K classes
  - We simply have K-1 sets of weights $\mathbf{w}_1, \ldots \mathbf{w}_{K-1}$
  - one for each class minus one (since the last one is redundant)

$$P(Y = k|\mathbf{x}, \mathbf{w}) = \frac{\exp\{\mathbf{w}_k^\top \mathbf{x}\}}{1+\sum_{k'=1}^{K-1} \exp\{\mathbf{w}_{k'}^\top \mathbf{x}\}}, \quad \text{for} \quad k = 1, \ldots, K-1$$

$$P(Y = K|\mathbf{x}, \mathbf{w}) = \frac{1}{1+\sum_{k'=1}^{K-1} \exp\{\mathbf{w}_{k'}^\top \mathbf{x}\}}$$

$$P(Y = k|\mathbf{x}, \mathbf{w}) = \frac{\exp\{\mathbf{w}_k^\top \mathbf{x}\}}{\sum_{k'=1}^{K} \exp\{\mathbf{w}_{k'}^\top \mathbf{x}\}}, \quad \text{for} \quad k = 1, \ldots, K$$

Linear boundary for Gaussian Naive Bayes

- For Gaussian Naive Bayes σ_{jk}
  - we typically estimate a separate variance for each feature j and each class k
  - consider a simpler model where we assume the variances are shared σ_j
  - so there is one parameter per feature
  - What this means is that the shape (the density contour ellipse) of the multivariate Gaussian for each class is the same.
  - In this case the equation for Naive Bayes is exactly the same as for logistic regression, and the decision boundary is linear:

$$P(Y=1|X) = \frac{1}{1+\exp\{w_0 + \mathbf{w}^\top X\}}, \text{ for some } w_0, \mathbf{w}$$

Let's derive it

Linear boundary for Gaussian NB with shared variances

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

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

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

Now let's plug in our definitions:

$$P(Y=1) = \theta$$

$$P(X_j|Y=1) = \frac{1}{\sigma_j\sqrt{2\pi}}\exp\left\{\frac{-(X_j-\mu_{j1})^2}{2\sigma_j^2}\right\}$$

$$P(X_j|Y=0) = \frac{1}{\sigma_j\sqrt{2\pi}}\exp\left\{\frac{-(X_j-\mu_{j0})^2}{2\sigma_j^2}\right\}$$

Linear boundary for Gaussian NB with shared variances

We have: $\log \frac{P(Y=0)P(X|Y=0)}{P(Y=1)P(X|Y=1)}$

$$= \log P(Y=0) + \log P(X|Y=0) - \log P(Y=1) - \log P(X|Y=1)$$

$$= \log \frac{P(Y=0)}{P(Y=1)} + \sum_j \log \frac{P(X_j|Y=0)}{P(X_j|Y=1)}$$

$$= \log \frac{1-\theta}{\theta} + \sum_j \left(\frac{(\mu_{j0}-\mu_{j1})}{\sigma_j^2} X_j + \frac{\mu_{j1}^2-\mu_{j0}^2}{2\sigma_j^2} \right)$$

$$= w_0 + \mathbf{w}^\top X$$

What you should know

- **Applications of document classification**
  - **Spam detection, email routing, author ID,**
  - **topic prediction, sentiment analysis**
- **Naïve Bayes**
  - **As MAP estimator (uses prior for smoothing)**
    - **Contrast MLE**
  - **For document classification**
    - **Use bag of words**
    - **Could use richer feature set**

电子科技大学

UESTC

1956