GI01/M055: Supervised Learning

2. Discriminative and Generative Models

October 12, 2009

John Shawe-Taylor

Today's plan

- Discriminative vs. generative models
- Linear and quadratic discriminant analysis
- Logistic regression
- Naive Bayes classifier

Bibliography: These lecture notes are available at:

http://www.cs.ucl.ac.uk/staff/J.Shawe-Taylor/courses/index-gi01.htm

Lectures are in part based on Chapter 4 of Hastie, Tibshirani, & Friedman

Summary from last class

Last week we have discussed two SL approaches:

- Empirical error minimization also known as Empirical risk minimization (ERM): look for a function in hypothesis space $\mathcal{H}$ (eg, $\mathcal{H} \equiv$ all linear functions) which minimizes the empirical error
- k-NN: classify by majority vote amongst the k nearest neighbors (of the input we wish to classify)

We emphasized differences between the two methods (parametric vs. non parametric, global vs. local, etc.)

Discriminative vs. generative methods

A common aspect of $k-$NN and ERM is that they both **directly** compute a function $f : \mathcal{X} \rightarrow \mathcal{Y}$ (or $P(y|\mathbf{x})$ as we'll see later) from available data **without** estimating the underlying probability model

Generative models approach (aka Statistical Decision Theory):

- first compute class conditional probabilities, $P(\mathbf{x}|y)$ $y \in \mathcal{Y}$ and class probabilities $P(y)$

- then extract $P(y|\mathbf{x})$ by Bayes rule (we'll see how to extract a classifier f in a moment)

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

Generative models

Consider the binary classification problem, $\mathcal{Y} = \{0, 1\}$

- Compute $P(\mathbf{x}|0)$ and $P(\mathbf{x}|1)$ within some model class via maximum likelihood

- Compute $P(0) = \frac{m_0}{m}$, where $m_0 = \#$data in class 0

- Use Bayes rule to compute $P(0|\mathbf{x}) = \frac{P(\mathbf{x}|0)P(0)}{P(\mathbf{x})}$

where $P(\mathbf{x}) = \sum_{y \in \mathcal{Y}} P(\mathbf{x}|y)P(y) = P(\mathbf{x}|0)P(0) + P(\mathbf{x}|1)(1 - P(0))$

Generative models (cont.)

Once we know $P(0|\mathbf{x})$ we classify $\mathbf{x}$ using the Bayes classifier:

$$f(\mathbf{x}) = \begin{cases} 0 & \text{if } P(0|\mathbf{x}) > \frac{1}{2} \\ 1 & \text{otherwise} \end{cases}$$

We can also write this as

$$f(\mathbf{x}) = \text{argmax}_{y\in\mathcal{Y}} \frac{P(\mathbf{x}|y)P(y)}{P(\mathbf{x})} = \text{argmax}_{y\in\mathcal{Y}} P(\mathbf{x}|y)P(y)$$

- Note that $P(\mathbf{x})$ is **not** important for classification

Discriminant function

Equivalently, we can introduce the **discriminant functions**

$$g_k(\mathbf{x}) = \log P(k|\mathbf{x}), \quad k = 0, 1$$

we classify $\mathbf{x}$ as 0 if $g(\mathbf{x}) := g_0(\mathbf{x}) - g_1(\mathbf{x}) > 0$ and 1 otherwise. That is

$$f(\mathbf{x}) = \text{argmax}_{k=0,1}\Big\{g_k(\mathbf{x})\Big\}$$

- Decision regions:

$$R_0 = \Big\{\mathbf{x} : g_0(\mathbf{x}) > g_1(\mathbf{x})\Big\}, \quad R_1 = \Big\{\mathbf{x} : g_1(\mathbf{x}) > g_0(\mathbf{x})\Big\}$$

- Decision boundary: $\Big\{\mathbf{x} : g_0(\mathbf{x}) = g_1(\mathbf{x})\Big\}$

Multiclass extension

The above can be extended naturally to more than two classes (say $\mathcal{Y} = \{c_1, \ldots, c_K\}$). We use the notation $P(k|\mathbf{x}) = P(y = c_k|\mathbf{x})$

$$g_k(\mathbf{x}) = \log P(k|\mathbf{x}), \quad k = 1, \ldots, K$$

(actually only $K-1$ discriminant functions need to be specified because probabilities must sum to one)

$$f(\mathbf{x}) = \text{argmax}_{k=1}^{K}\Big\{g_k(\mathbf{x})\Big\}$$

Multiclass extension (cont.)

$$f(\mathbf{x}) = \text{argmax}_{k=1}^{K} g_k(\mathbf{x})$$

- Decision regions: $R_k = \left\{\mathbf{x} : g_k(\mathbf{x}) > g_\ell(\mathbf{x}),\ \text{for all } \ell \neq k\right\}$

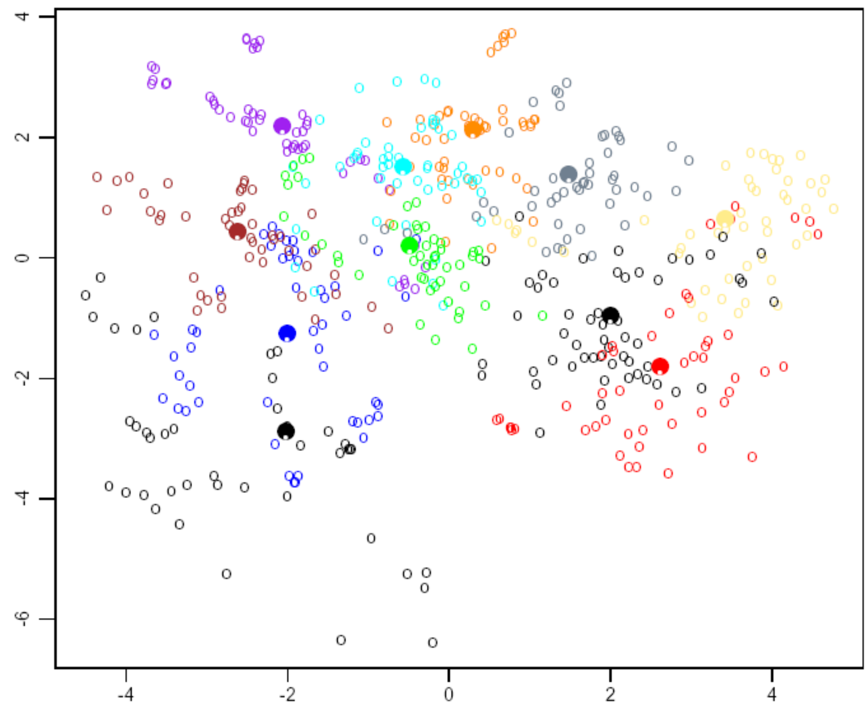
- Decision boundaries: $\left\{\mathbf{x} : g_k(\mathbf{x}) = g_\ell(\mathbf{x}), k \neq \ell, g_q(\mathbf{x}) \leq g_k(\mathbf{x}) \text{ for all } q\right\}$

  (roughly speaking, there is a decision boundary between class k and ℓ if "ties occurs" among those classes)

Multiclass example

We introduce discriminant functions $g_k(\mathbf{x})$ for each class $k = 1, \ldots, K$ and use the classification rule:

$$f(\mathbf{x}) = \text{argmax}_{k=1}^{K} g_k(\mathbf{x})$$

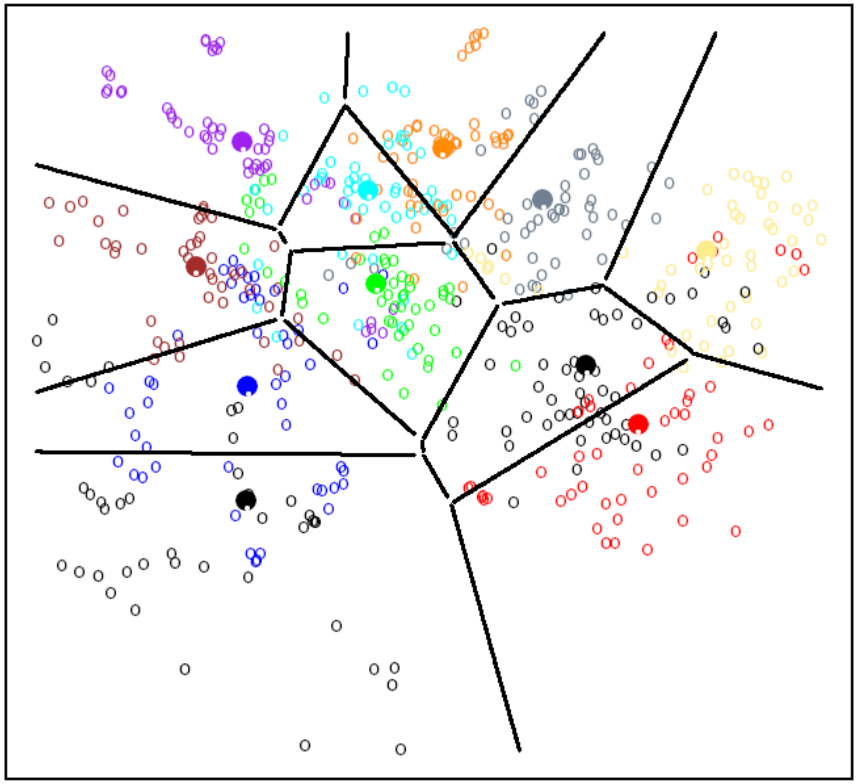


Multiclass example (cont.)

If the discriminant functions are linear, f partitions the input space in piecewise linear regions

$R_k = \{\mathbf{x} : g_k(\mathbf{x}) > g_\ell(\mathbf{x}), k \neq \ell\}$

The decision boundaries are the lines (hyperplanes in $\mathbb{R}^d$) of the type $\{\mathbf{x} : g_k(\mathbf{x}) = g_\ell(\mathbf{x}), k \neq \ell\}$ (for some k and ℓ, not all!) Boundaries also linear if g_k is minus distance to a centre as in diagram. Gives so-called Voronoi diagram.



Some well studied generative models

A generative model is identified by choosing a parameterized family of densities $P(\mathbf{x}|y)$ such as:

- Gaussians
- Mixture of Gaussians
- Naive Bayes: based on assumption $P(\mathbf{x}|y) = \prod_{i=1}^{d} P_i(x_i|y)$
- Some more general non-parametric densities

Gaussian densities

We will assume that $P(\mathbf{x}|0)$, $P(\mathbf{x}, 1)$ are Gaussians with different means and covariances. The Gaussian density is defined as

$$G(\mathbf{x}; \mu, \mathbf{\Sigma}) := \frac{1}{(2\pi)^{\frac{d}{2}}|\mathbf{\Sigma}|^{\frac{1}{2}}} \exp\left\{-\frac{1}{2}(\mathbf{x}-\mu)^\top \mathbf{\Sigma}^{-1}(\mathbf{x}-\mu)\right\}$$

where $|\mathbf{\Sigma}|$ is the determinant of matrix $\mathbf{\Sigma}$

Recall two important properties of the Gaussian:

- μ is the mean of $\mathbf{x}$: $\mathbf{E}[\mathbf{x}] = \mu$
- $\mathbf{\Sigma}$ is the covariance of $\mathbf{x}$: $\mathbf{E}[(\mathbf{x}-\mu)(\mathbf{x}-\mu)^\top] = \mathbf{\Sigma}$

Linear and quadratic discriminant analysis

We compute the parameters $\theta = \{\mu_0, \mu_1, \Sigma_0, \Sigma_1, \pi_0\}$ via maximum likelihood (we use the notation $\pi_0 := P(y = 0)$):

$$L(\theta; S) = \prod_{i=1}^{m} P(\mathbf{x}_i, y_i; \theta) = \prod_{i=1}^{m} P(\mathbf{x}_i | y_i; \theta) P(y_i)$$

The minus log likelihood is

$$\begin{aligned} -\log L \ &= \ \frac{1}{2} \sum_{i:y_i=0} (\mathbf{x}_i - \mu_0)^\top \mathbf{\Sigma}_0^{-1} (\mathbf{x}_i - \mu_0) + \frac{1}{2} \sum_{i:y_i=1} (\mathbf{x}_i - \mu_1)^\top \mathbf{\Sigma}_1^{-1} (\mathbf{x}_i - \mu_1) \\ &\quad + \frac{m_0}{2} \log |\Sigma_0| + \frac{m_1}{2} \log |\Sigma_1| + m_0 \log \pi_0 + m_1 \log(1 - \pi_0) + const. \end{aligned}$$

- $\{\mu_0, \Sigma_0\}$, $\{\mu_1, \Sigma_1\}$ and π_0 can be separately computed!
- LDA: Σ_0 and Σ_1 constrained to be equal,   QDA: $\Sigma_0 \neq \Sigma_1$

Univariate case: ML solution

In this case we have (we use the notation $\Sigma = \sigma^2$)

$$-\log L \;=\; \frac{1}{2}\sum_{i \in C(0)} \frac{(x_i-\mu_0)^2}{\sigma_0^2} + \frac{1}{2}\sum_{i \in C(1)} \frac{(x_i-\mu_1)^2}{\sigma_1^2}$$

$$+m_0 \log|\sigma_0| + m_1 \log|\sigma_1| + m_0 \log \pi_0 + m_1 \log(1-\pi_0) + const.$$

Solving for $\nabla \log L = 0$ we obtain (please verify this):

- $\pi_0 = \frac{m_0}{m}$
- $\mu_0 = \frac{1}{m_0}\sum_{i:y_i=0} x_i, \quad \sigma_0^2 = \frac{1}{m_0}\sum_{i:y_i=0}(x_i-\mu_0)^2$
- $\mu_1 = \frac{1}{m_1}\sum_{i:y_i=1} x_i, \quad \sigma_1^2 = \frac{1}{m_1}\sum_{i\in y_i=1}(x_i-\mu_1)^2$

Univariate case: discriminant function

$$P(x|0) = \frac{1}{\sqrt{2\pi}\sigma_0} \exp\left\{-\frac{(x-\mu_0)^2}{2\sigma_0^2}\right\}, \quad P(x|1) = \frac{1}{\sqrt{2\pi}\sigma_1} \exp\left\{-\frac{(x-\mu_1)^2}{2\sigma_1^2}\right\},$$

Recalling that $g_k(x) = \log P(k|x) = \log P(x|k)P(k)$ (minus an unimportant $\log P(x)$), we obtain

$$g_k(x) = -\frac{x^2}{2\sigma_k^2} + \frac{\mu_k x}{\sigma_k^2} - \frac{\mu_k^2}{2\sigma_k^2} + \log \frac{\pi_k}{\sqrt{2\pi}\sigma_k}, \quad k = 0, 1$$

Univariate case: discriminant function

$$g_k(x) = -\frac{x^2}{2\sigma_k^2} + \frac{\mu_k x}{\sigma_k^2} - \frac{\mu_k^2}{2\sigma_k^2} + \log \frac{\pi_k}{\sqrt{2\pi}\sigma_k}$$

Hence, in general, the discriminant functions need to be quadratic

However, if $\sigma_0 = \sigma_1 = \sigma$ we can choose them to be linear (can drop term $\frac{x^2}{2\sigma_k}$)

In this case the ML solution for σ is

$$\sigma^2 = \frac{1}{m}\left\{\sum_{i:y_i=0}(x_i - \mu_0)^2 + \sum_{i:y_i=1}(x_i - \mu_1)^2\right\}$$

Multivariate case

Estimating parameters in multivariate case: solving for $\nabla \log L = 0$ we obtain:

- $\pi_0 = \frac{m_0}{m}$
- $\mu_0 = \frac{1}{m_0} \sum_{i:y_i=0} \mathbf{x}_i, \quad \Sigma_0 = \frac{1}{m_0} \sum_{i:y_i=0} (\mathbf{x}_i - \mu_0)(\mathbf{x}_i - \mu_0)^\top$
- $\mu_1 = \frac{1}{m_1} \sum_{i:y_i=1} \mathbf{x}_i, \quad \Sigma_1 = \frac{1}{m_1} \sum_{i \in y_i=1} (\mathbf{x}_i - \mu_1)(\mathbf{x}_i - \mu_1)^\top$
- if constrain $\Sigma_0 = \Sigma_1$: $\Sigma = \frac{1}{m} \sum_{k=0}^{1} \sum_{i \in y_i=k} (\mathbf{x}_i - \mu_k)(\mathbf{x}_i - \mu_1)^\top$

Verifying this involves use of equations for matrix differentials: the relevant results are given on the web page:

http://www.ee.ic.ac.uk/hp/staff/dmb/matrix/calculus.html#deriv_quad

Multivariate case

Similarly to the univariate case, we have

$$g(\mathbf{x}) := \log \frac{P(0|\mathbf{x})}{P(1|\mathbf{x})} = \log \frac{P(\mathbf{x}|0)P(0)}{P(\mathbf{x}|1)P(1)} = g_0(\mathbf{x}) - g_1(\mathbf{x})$$

where

$$g_k(\mathbf{x}) = -\frac{1}{2}\mathbf{x}^\top\Sigma_k^{-1}\mathbf{x} + \mu_k^\top\Sigma_k^{-1}\mathbf{x} + b_k, \quad b_k := -\frac{1}{2}\mu_k^\top\Sigma_k^{-1}\mu_k + \log\left(\frac{\pi_k}{(2\pi)^{\frac{d}{2}}|\Sigma_k|^{\frac{1}{2}}}\right)$$

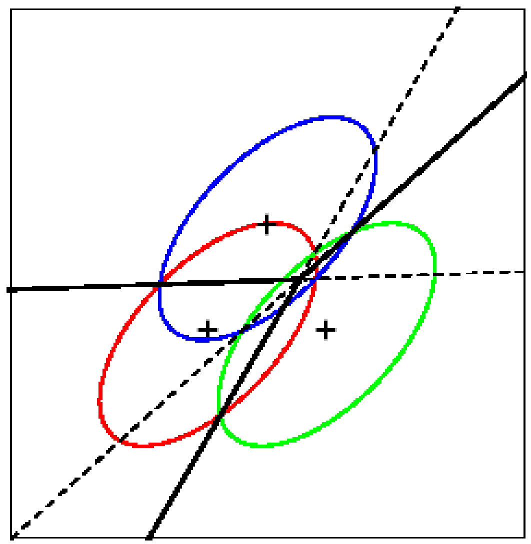
In general, g is a multiquadric (we call this QDA)

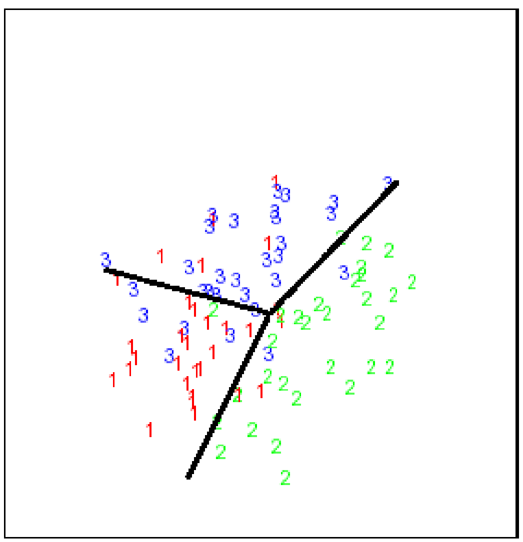
However, if $\Sigma_0 = \Sigma_1 = \Sigma$ then $g(\mathbf{x})$ is linear in $\mathbf{x}$: (we call this LDA)

$$g(\mathbf{x}) = (\mu_0 - \mu_1)^\top\Sigma^{-1}\mathbf{x} - \frac{1}{2}(\mu_1 + \mu_0)^\top\Sigma^{-1}(\mu_0 - \mu_1) + \log\frac{\pi_0}{1 - \pi_0}$$

3 classes example: equal covariances

If $\Sigma_0 = \Sigma_1 = \Sigma_2$
then $g_k(\mathbf{x})$ are linear

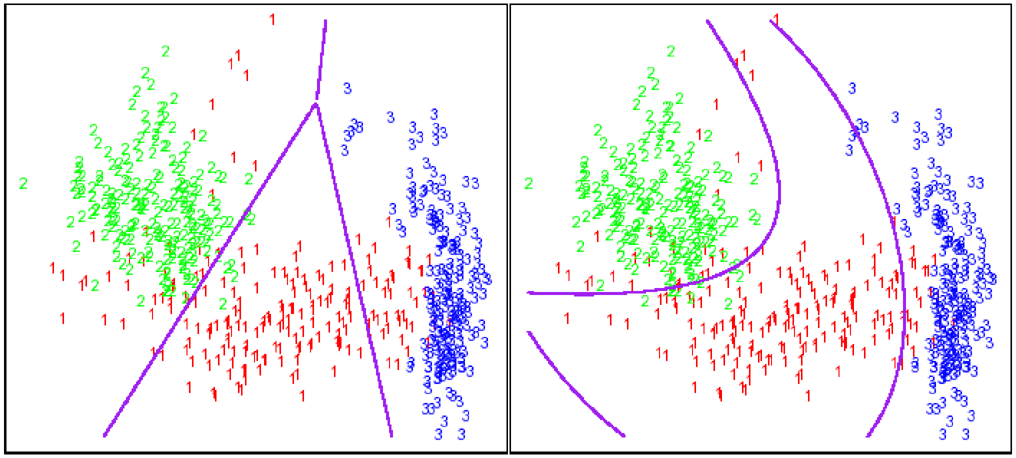




$$g_k(\mathbf{x}) = \mu_k^\top \Sigma^{-1} \mathbf{x} - \frac{1}{2} \mu_k^\top \Sigma^{-1} \mu_k + \log \pi_k$$

3 classes example: linear vs. non-linear

Here is an example where using different covariances gives a better model...



...However:

- LDA: need to fit $(K-1)(d+1)$ parameters (since we need to compute $K-1$ differences $g_k - g_\ell$ and each has $d+1$ parameters)
- QDA: need to fit $(K-1)\frac{d(d+2)}{2}$ parameters, so if d is high QDA may more easily overfit our data

Logistic regression (I)

Let's go back to the discriminative model approach. Assume that

$$\log \frac{P(0|\mathbf{x})}{P(1|\mathbf{x})} = -(\mathbf{w}^\top \mathbf{x} + b) \quad \text{(incorporate } b \text{ in } \mathbf{w}\text{...)}$$

Using $P(0|\mathbf{x}) + P(1|\mathbf{x}) = 1$, a simple computation gives

$$P(1|\mathbf{x}) \equiv p(\mathbf{x}; \mathbf{w}) = \frac{1}{1 + e^{-\mathbf{w}^\top \mathbf{x}}}$$

Note: for simplicity, we discuss only binary classification but all of what we say naturally extends to the multiclass case

Logistic regression (II)

Recall our notation from last class

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}_1^\top \\ \vdots \\ \mathbf{x}_m^\top \end{bmatrix}, \quad \mathbf{y} = \begin{bmatrix} y_1 \\ \vdots \\ y_m \end{bmatrix}$$

We compute $\mathbf{w}$ by maximizing the conditional likelihood:

$$L(\mathbf{w}; \mathbf{y}|\mathbf{X}) = P(\mathbf{y}|\mathbf{X}; \mathbf{w}) = \prod_{i=1}^{m} P(y_i|\mathbf{x}_i; \mathbf{w})$$

Logistic regression (III)

The log-likelihood is given by (modulo an additive constant term)

$$\ell(\mathbf{w}) := \log L(\mathbf{w}; \mathbf{y}|\mathbf{X}) \;=\; \sum_{i=1}^{m} \Big\{ y_i \log p(\mathbf{x}_i; \mathbf{w}) + (1 - y_i) \log \Big(1 - p(\mathbf{x}_i; \mathbf{w})\Big)\Big\}$$

The quantity

$$-y \log p(\mathbf{x}; \mathbf{w}) - (1 - y) \log(1 - p(\mathbf{x}; \mathbf{w}))$$

is the **cross entropy function** between the binary probability functions $(y, 1 - y)$ and $(p(\mathbf{x}; \mathbf{w}), 1 - p(\mathbf{x}; \mathbf{w}))$.

For distributions p and q the cross-entropy between p and q is defined as

$$H(p, q) = -\sum_{x} p(x) \log q(x) = H(p) + D_{\mathsf{KL}}(p\|q).$$

Loss function

Thus maximizing the likelihood is equivalent to minimizing a generalized type of empirical error:

$$\mathcal{E}_{\text{emp}} = \sum_{i=1}^{m} V\Big(y_i, f(\mathbf{x})\Big), \qquad f(\mathbf{x}) = \mathbf{w}^\top \mathbf{x}$$

where $V : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$ is called the **loss function**

- Least squares: $V\Big(y, f(\mathbf{x})\Big) = \Big(y - f(\mathbf{x})\Big)^2$

- Logistic regression:

$$V\Big(y, f(\mathbf{x})\Big) = y \log\Big(1 + e^{-f(\mathbf{x})}\Big) + (1 - y) \log\Big(1 + e^{f(\mathbf{x})}\Big)$$

Logistic regression (IV)

$$\ell(\mathbf{w}) = \sum_{i=1}^{m} \Big\{ y_i \log p(\mathbf{x}_i; \mathbf{w}) + (1 - y_i) \log \Big(1 - p(\mathbf{x}_i; \mathbf{w})\Big) \Big\}$$

Setting the derivatives to zero we obtain the nonlinear equations:

$$\nabla \ell(\mathbf{w}) = \sum_{i=1}^{m} \mathbf{x}_i \Big(y_i - p(\mathbf{x}_i; \mathbf{w}) \Big) = 0$$

Compare to normal equations for least squares:

$$\sum_{i=1}^{m} \mathbf{x}_i \mathbf{x}_i^\top \mathbf{w} = \sum_{i=1}^{m} \mathbf{x}_i y_i \qquad \text{or} \qquad \sum_{i=1}^{m} \mathbf{x}_i \Big(y_i - \mathbf{x}_i^\top \mathbf{w} \Big) = 0$$

They look very similar! We'll see next week how to solve those

Log-Reg versus LDA

Let's go back to LDA. We assumed that $P(\mathbf{x}|0)$ and $P(\mathbf{x}|1)$ are Gaussians with the same covariance and estimated their mean and covariance (as well as the class probabilities) by ML

It follows that $P(\mathbf{x})$ is a **mixture of Gaussians**

More interestingly, it is easy to verify that

$$P(1|\mathbf{x}) = \frac{1}{1 + e^{-\mathbf{w}^\top \mathbf{x}}} \qquad \text{like in logistic regression!}$$

Logistic regression vs. LDA (cont.)

However, in logistic regression, $P(\mathbf{x})$ will **not** in general be a mixture of Gaussians!

- LDA based on stronger assumptions than Log-Reg
- Log-Reg leaves the marginal density of $\mathbf{x}$ arbitrary and fits parameter $\mathbf{w}$ by maximizing the conditional likelihood
- If $P(\mathbf{x}|0)$ and $P(\mathbf{x}|1)$ are indeed Gaussians then we should use LDA
- Otherwise Log-Reg should work better (more robust to the underlying $P(\mathbf{x})$)

Naive Bayes classifier

Based on the following simple assumption:

$$P(\mathbf{x}|y) = \prod_{j=1}^{d} P(x_j|y)$$

Meaning: the components of $\mathbf{x}$ are conditionally independent given y:

$$\begin{aligned} P(\mathbf{x} = (x_1, \ldots, x_d)|y) &= P(x_1|y)P(x_2|y, x_1) \cdots P(x_d|y, x_1, \ldots, x_{d-1}) \\ &= P(x_1|y)P(x_2|y) \cdots P(x_d|y) = \prod_{j=1}^{d} P(x_j|y) \end{aligned}$$

Naive Bayes (cont.)

Individual class conditional probabilities can be estimated independently!

Discriminant functions (recall $\pi_k := P(y = c_k)$)

$$g_k(\mathbf{x}) = \log P(\mathbf{x}|k)\pi_k = \sum_{j=1}^{d} \log P(x_j|k) + \log \pi_k$$

As before if $P(x_j|k)$ are Gaussians the discriminant functions are linear

- Naive Bayes is a very simple model! Yet, if d is very large it is a good choice to try

Naive Bayes: binary features

Example ("bag of words" representation for text documents) Assume x_j are binary variables and $x_j = 1$ if $j-$th word in our dictionary appears in document $\mathbf{x}$ and $x_j = 0$ otherwise

Define $p_{jk} := P(x_j = 1|y = k)$ and $\pi_k := P(y = k)$ – here we are thinking of p_{jk} as being a distribution over words.

One can show (exercise) that the maximum likelihood estimate of p_{jk} and π_k (constraining $\sum_j p_{jk} = 1 = \sum_k \pi_k$) is

$$
\begin{aligned}
p_{jk} &= \frac{\#\Big\{(\mathbf{x}, y) \in S : x_j = 1 \text{ and } y = k\Big\}}{\sum_{j'} \#\Big\{(\mathbf{x}, y) \in S : x_{j'} = 1 \text{ and } y = k\Big\}} \\
\pi_k &= \frac{\#\{(\mathbf{x}, y) \in S : y = k\}}{m}
\end{aligned}
$$

Dealing with rare words

Note that if, say, the $h-$th word is not in any training input data,

$$p_{hk} = \frac{\#\Big\{(\mathbf{x}, y) \in S : x_h = 1 \text{ and } y = k\Big\}}{\sum_{h'} \#\Big\{(\mathbf{x}, y) \in S : x_{h'} = 1 \text{ and } y = k\Big\}} = \frac{0}{m_k} = 0, \quad \text{for all } k$$

However, if a new document $\mathbf{x}$ contains the $h-$th word, we have: $p_{hk} = 0 \Rightarrow P(\mathbf{x}|k) = 0 \Rightarrow P(\mathbf{x}) = 0$. Hence

$$P(k|\mathbf{x}) = \frac{P(\mathbf{x}|k)\pi_k}{P(\mathbf{x})} = \frac{0}{0}$$

To avoid this pathological situation we introduce the following modified estimator (N is the number of words – including those not in the training set)

$$p_{hk} = \frac{\#\Big\{(\mathbf{x}, y) \in S : x_h = 1 \text{ and } y = k\Big\} + 1}{N + \sum_{h'} \#\Big\{(\mathbf{x}, y) \in S : x_{h'} = 1 \text{ and } y = k\Big\}}$$