GI01/4C55: Supervised Learning

2. Discriminative and Generative Models

October 10, 2005

Massimiliano Pontil

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/M.Pontil/courses/index-SL05.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: look for a function in hypothesis space \( \mathcal{H} \) (eg, \( \mathcal{H} \equiv \text{all linear functions} \)) 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 RSS is that they both directly compute a function \( f : \mathcal{X} \rightarrow \mathcal{Y} \) (or \( P(y|x) \) 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, \( P(x|y) \) \( y \in \mathcal{Y} \) and class probabilities \( P(y) \)

- then extract \( P(y|x) \) by Bayes rule (we’ll see how to extract a classifier \( f \) in a moment)

\[
P(y|x) = \frac{P(x|y)P(y)}{P(x)}
\]
Generative models

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

- Compute $P(x|0)$ and $P(x|1)$ 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 $P(0|x) = \frac{P(x|0)P(0)}{P(x)}$

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

Generative models (cont.)

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

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

We can also write this as

$$f(x) = \arg\max_{y \in \mathcal{Y}} \frac{P(x|y)P(y)}{P(x)} = \arg\max_{y \in \mathcal{Y}} P(x|y)P(y)$$

- Note that $P(x)$ is not important for classification
**Discriminant function**

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

\[ g_k(x) = \log P(k|x), \quad k = 0, 1 \]

we classify \( x \) as 0 if \( g(x) := g_0(x) - g_1(x) > 0 \) and 1 otherwise.

That is

\[ f(x) = \arg\max_{k=0,1} \{g_k(x)\} \]

- Decision regions:
  \[ R_0 = \{x : g_0(x) > g_1(x)\}, \quad R_1 = \{x : g_1(x) > g_0(x)\} \]

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

**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 \( P(k|x) = P(y = c_k|x) \)

\[ g_k(x) = \log P(k|x), \quad k = 1, \ldots, K \]

(actually only \( K - 1 \) discriminant f.ons need to be specified because probabilities must sum to one)

\[ f(x) = \arg\max_{k=1}^{K} \{g_k(x)\} \]
Multiclass extension (cont.)

\[ f(x) = \arg\max_{k=1}^{K} g_k(x) \]

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

- Decision boundaries: \( \{ x : g_k(x) = g_{\ell}(x), k \neq \ell, g_k(x) \geq g_q(x) \text{ for all } q \} \)

(roughly speaking, there is a decision boundary between class \( k \) and \( \ell \) if “ties occurs” among those classes)

9

Multiclass example

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

\[ f(x) = \arg\max_{k=1}^{K} g_k(x) \]
Multiclass example (cont.)

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

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

The decision boundaries are the lines (hyperplanes in $\mathbb{R}^d$) of the type $\{x : g_k(x) = g_\ell(x), k \neq \ell\}$ (for some $k$ and $\ell$, not all!)

---

Some well studied generative models

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

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

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

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

where $|\Sigma|$ is the determinant of matrix $\Sigma$.

Recall two important properties of the Gaussian:

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

---

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 $\pi_0 := P(y = 0)$):

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

The minus log likelihood is

$$-\log L = \frac{1}{2} \sum_{i:y_i=0} (x_i - \mu_0)^\top \Sigma_0^{-1}(x_i - \mu_0) + \frac{1}{2} \sum_{i:y_i=1} (x_i - \mu_1)^\top \Sigma_1^{-1}(x_i - \mu_1) + \frac{m_0}{2} \log |\Sigma_0| + \frac{m_1}{2} \log |\Sigma_1| + m_0 \log \pi_0 + m_1 \log (1 - \pi_0) + \text{const.}$$

- $\{\mu_0, \Sigma_0\}$, $\{\mu_1, \Sigma_1\}$ and $\pi_0$ can be separately computed!
- LDA: $\Sigma_0$ and $\Sigma_1$ constrained to be equal, QDA: $\Sigma_0 \neq \Sigma_1$
Univariate case: ML solution

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

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

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

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

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

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 \( \frac{x^2}{2\sigma_k^2} \))

In this case the ML solution for \( \sigma \) 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

Similarly to the univariate case, we have

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

where

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

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

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

\[ g(x) = (\mu_0 - \mu_1)^\top \Sigma^{-1} x - \frac{1}{2} (\mu_1 + \mu_0)^\top \Sigma^{-1} (\mu_0 - \mu_1) + \log \frac{\pi_0}{1 - \pi_0} \]
3 classes example: equal covariances

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

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

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

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

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

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

**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

\[
X = \begin{bmatrix} x_1^\top \\ \vdots \\ x_m^\top \end{bmatrix}, \quad y = \begin{bmatrix} y_1 \\ \vdots \\ y_m \end{bmatrix}
\]

We compute \( w \) by maximizing the conditional likelihood:

\[
L(w; y|X) = P(y|X; w) = \prod_{i=1}^{m} P(y_i|x_i; w)
\]
Logistic regression (III)

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

\[ \ell(w) : = \log L(w; y|X) = \sum_{i=1}^{m} \left\{ y_i \log p(x_i; w) + (1 - y_i) \log (1 - p(x_i; w)) \right\} \]

The quantity

\[ y \log p(x; w) + (1 - y) \log (1 - p(x; w)) \]

is called the cross entropy function. This is exactly the relative entropy (studied in information theory) between the binary probability functions \((y, 1 - y)\) and \((p(x; w), 1 - p(x; w))\)

Loss function

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

\[ \mathcal{E}_{\text{emp}} = \sum_{i=1}^{m} V(y_i, f(x)), \quad f(x) = w^T x \]

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

- Least squares: \(V(y, f(x)) = (y - f(x))^2\)

- Logistic regression:

\[ V(y, f(x)) = y \log (1 + e^{-f(x)}) + (1 - y) \log (1 + e^{f(x)}) \]
Logistic regression (IV)

\[ \ell(w) = \sum_{i=1}^{m} \left\{ y_i \log p(x_i; w) + (1 - y_i) \log \left( 1 - p(x_i; w) \right) \right\} \]

Setting the derivatives to zero we obtain the nonlinear equations:

\[ \nabla \ell(w) = \sum_{i=1}^{m} x_i (y_i - p(x_i; w)) = 0 \]

Compare to normal equations for least squares:

\[ \sum_{i=1}^{m} x_i x_i^T w = \sum_{i=1}^{m} x_i y_i \quad \text{or} \quad \sum_{i=1}^{m} x_i (y_i - x_i^T w) = 0 \]

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 \( P(x|0) \) and \( P(x|1) \) are Gaussians with the same covariance and estimated their mean and covariance (as well as the class probabilities) by ML.

It follows that \( P(x) \) is a **mixture of Gaussians**

More interestingly, it is easy to verify that

\[ P(1|x) = \frac{1}{1 + e^{-w^T x}} \]

like in logistic regression!
Logistic regression vs. LDA (cont.)

However, in logistic regression, \( P(x) \) will not in general be a mixture of Gaussians!

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

Naive Bayes classifier

Based on the following simple assumption:

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

Meaning: the components of \( x \) are conditionally independent given \( y \):

\[
P(x = (x_1, \ldots, x_d)|y) = P(x_1|y)P(x_2|y, x_1) \cdots P(x_d|y, x_1, \ldots, x_{d-1})
\]

\[
= P(x_1|y)P(x_2|y) \cdots P(x_d|y) = \prod_{j=1}^{d} P(x_j|y)
\]
Naive Bayes (cont.)

Individual class conditional probabilities can be estimated independently!

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

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

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 \( x \) and \( x_j = 0 \) otherwise

Define \( p_{jk} := P(x_j = 1|y = k) \) and \( \pi_k := P(y = k) \)

One can show (exercise) that the maximum likelihood estimate of \( p_{jk} \) and \( \pi_k \) is

\[
p_{jk} = \frac{\#\{(x, y) \in S : x_j = 1 \text{ and } y = k\}}{\#\{(x, y) \in S : y = k\}}
\]

\[
\pi_k = \frac{\#\{(x, y) \in S : y = k\}}{m}
\]
Dealing with rare words

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

\[
p_{hk} = \frac{\#\{(x, y) \in S : x_h = 1 \text{ and } y = k\}}{\#\{(x, y) \in S : y = k\}} = \frac{0}{m_k} = 0, \quad \text{for all } k
\]

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

\[
P(k|x) = \frac{P(x|k)\pi_k}{P(x)} = \frac{0}{0}
\]

To avoid this pathological situation we introduce the following modified estimator

\[
p_{hk} = \frac{\#\{(x, y) \in S : x_h = 1 \text{ and } y = k\} + 1}{\#\{(x, y) \in S : y = k\} + K}
\]