Introduction to Machine Learning **Fall Semester, 2013**

Lecture 3: October 27

Lecturer: Eran Halperin *Scribe: ym*

Some of the material was not presented in class (and is marked with a side line) and is given for completeness.

3.1 Topics for Bayesian Inference

We will cover the following topics:

1. Maximum Likelihood - previous lecture
2. Prior and Posterior distribution -previous lecture
3. Naïve Bayes - today
4. Expectation Maximization -today

3.2 Naïve Bayes

Assume we are given a data set of patients that has cholesterol level and whether they had a heart attack. Here is an example:

id	cholesterol	Heart Attack (HA)
1	150	0
⋮	⋮	⋮
999	240	1

Our goal is to design a classifier that given the cholesterol level will predict whether the patient will have a heart attack. (Note that the prediction is binary: $y = 0$ means *no heart attack* and $y = 1$ means *heart attack*.) Our classifier will be given a cholesterol level x and predict y.

We assume that the cholesterol level, given HA is a normal distribution. More specifically,

$$\Pr[X = x|HA = 1] \sim N(\mu_1, \sigma^2), \qquad \Pr[X = x|HA = 0] \sim N(\mu_0, \sigma^2)$$

This implies that the parameters of our model are $\theta = \{\mu_1, \mu_0, \sigma\}$. We need to estimate those parameters from data. One way to do this is using the Maximum Likelihood estimator (from last lecture). We would like to determine whether it is more likely that $HA = 1$ or $HA = 0$, given $X = x$. (This is a simple instantiation of the MAP, selecting the classification with the highest a posteriori probability.) This implies that we predict $y = 1$ if,

$$\frac{\Pr[y=1|x]}{\Pr[y=0|x]} \geq 1$$

By Bayes rule we have that

$$\Pr[y=b|x] = \frac{\Pr[x|y=b]\Pr[y=b]}{\Pr[x]}$$

Plugging back to the inequality we have

$$\frac{\Pr[y=1|x]}{\Pr[y=0|x]} = \frac{\frac{\Pr[x|y=1]\Pr[y=1]}{\Pr[x]}}{\frac{\Pr[x|y=0]\Pr[y=0]}{\Pr[x]}} = \frac{\Pr[x|y=1]\Pr[y=1]}{\Pr[x|y=0]\Pr[y=0]} \geq 1$$

Taking the logarithms we have

$$\log\left(\frac{\Pr[y=1]}{\Pr[y=0]}\right) + \log\left(\frac{\Pr[x|y=1]}{\Pr[x|y=0]}\right) \geq 0$$

Using our assumption that the x given y is a normal distribution we have,

$$\log\left(\frac{\Pr[y=1]}{\Pr[y=0]}\right) + \frac{1}{2\sigma^2}\left((x-\mu_0)^2 - (x-\mu_1)^2\right) = \log\left(\frac{\Pr[y=1]}{\Pr[y=0]}\right) + \frac{\mu_1-\mu_0}{\sigma^2}\left(x - \frac{\mu_1+\mu_0}{2}\right) \geq 0$$

If we assume that $\Pr[y=1] = \Pr[y=0]$ then we are simply testing if x is above or below the average of the two means.

If we use a maximum likelihood estimator for the parameters then we can do the following. Let $S_b = \{i|y_i = b\}$. We set $\widehat{\Pr}[y=b] = |S_b|/n$ and $\hat{\mu}_b = \sum_{i \in S_b} x_i/|S_b|$.

It is fairly straightforward to extend it to multiple attributes. Our assumption for the Naïve Bayes would be that the attributes are independent given the classification. This will imply that,

$$\begin{aligned}\Pr[y=1|x_1,\ldots,x_n] =& \frac{\Pr[x_1,\ldots,x_n|y=1]\Pr[y=1]}{\Pr[x_1,\ldots,x_n]}\\ =& \frac{\Pr[x_1|y=1]\cdots\Pr[x_n|y=1]\Pr[y=1]}{\Pr[x_1,\ldots,x_n]}\end{aligned}$$

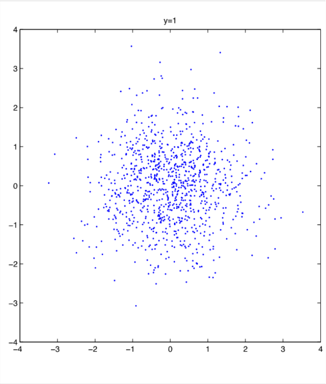

Figure 3.1: $y = 1$: Independent variables

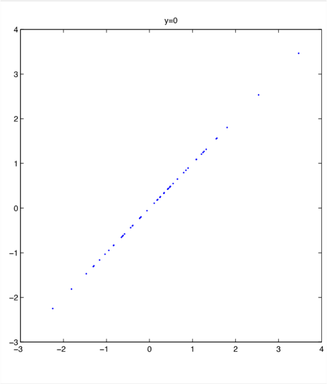

Figure 3.2: $y = 0$: $x_2 = x_1$

Assume that when $y = b$ then x_i is distributed normally $N(\mu_{i,b}, \sigma_i^2)$. Taking the logarithm and using the normality assumption we have,

$$\begin{aligned}
\log\left(\frac{\Pr[y=1|x_1,\ldots,x_n]}{\Pr[y=0|x_1,\ldots,x_n]}\right) &= \log\left(\frac{\Pr[y=1]}{\Pr[y=0]}\right) + \sum_{i=1}^{n} \log\frac{\Pr[x_i|y=1]}{\Pr[x_i|y=0]} \\
&= \log\left(\frac{\Pr[y=1]}{\Pr[y=0]}\right) + \sum_{i=1}^{n} \frac{1}{2\sigma_i^2}\left((x-\mu_{i,0})^2 - (x-\mu_{i,1})^2\right) \\
&= \log\left(\frac{\Pr[y=1]}{\Pr[y=0]}\right) + \sum_{i=1}^{n} \frac{\mu_{i,1}-\mu_{i,0}}{\sigma_i^2}\left(x - \frac{\mu_{i,1}+\mu_{i,0}}{2}\right) \geq 0
\end{aligned}$$

A few remarks about the Naïve Bayes classifier:

1. A Naïve assumption. Rarely it will hold in practice, so we should not take the assumption "seriously".

2. Easy to implement. Probably one of the simplest models.

3. Often works in practice. Definitely gives a reasonable baseline results in many applications.

4. Interpretation: A weighted sum of evidence.

5. Allows for the incorporation of features of different distributions. We can mix normal distribution with a Bernoulli r.v., for example.

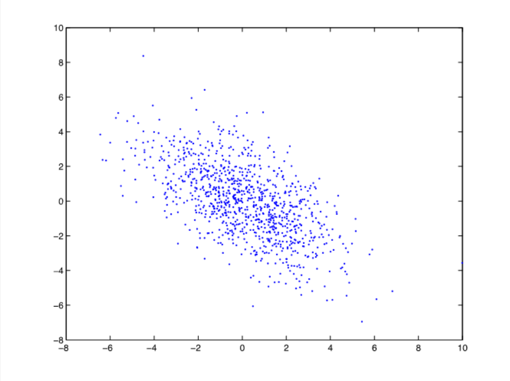

Figure 3.3: Multivariate normal distribution

6. Requires small amounts of data. The number of parameters is equal to the number of classes times the number of attributes. This implies that even with a small sample set, we can get a reasonable estimate of the parameters.

It is important to see when can the independence assumption can break, and what it entails. Consider the distribution in Figures 3.1 and 3.2. The marginal distribution are identical, so Naïve Bayes can not distinguish the two distribution, and the only classification will come from the sample randomization. On the other hand, it is clear that we can get a perfect classifier, by simply testing whether $x_1 = x_2$.

3.3 Multivariate Normal Distribution

A multivariate normal distribution is define over vectors, rather than scalars. One simple way to generate a multivariate normal distribution over d attributes it to first sample n normal univariate random variables: $z_1, \ldots, z_d \sim N(0, 1)$. We generate the multivariate distribution by setting $x = Az + \mu$ where A is an $d \times d$ matrix and μ is a vector of length d. The vector μ would be the expected values of the individual attributes. See Figure 3.3 for an example, with $A = \begin{pmatrix} 2 & 1 \\ -2 & 1 \end{pmatrix}$ and $\Sigma = AA^t = \begin{pmatrix} 5 & -3 \\ -3 & 5 \end{pmatrix}$. In general, since $\Sigma = AA^t$ it is both symmetric and positive semi-definite. (You can see that Σ is positive semi-definite, since $x^t \Sigma x = x^t A A^t x = \|A^t x\|^2 \geq 0$.) The matrix Σ is the variance-covariance matrix of the d attributes.

An equivalent definition for a multivariate normal distribution is using its density. $X \sim MVN(\mu, \Sigma)$, where μ is the means and Σ is the variance-covariance matrix.

$$f(x) = \frac{1}{\sqrt{(2\pi)^d |\Sigma|}} e^{-\frac{1}{2}(x-\mu)^t \Sigma^{-1}(x-\mu)}$$

This model has $d(d+1)/2$ parameters in Σ and d parameters in μ.

3.4 k-means

We can now go back to the k-means algorithm from the first lecture and give it a Bayesian interpretation. Recall that we are given n vectors $x_1, \ldots, x_n$ and a number k and our objective is to minimize

$$\min_{\mu_1,\ldots,\mu_k,S_1,\ldots,S_k} \sum_{i=1}^{k} \sum_{j \in S_i} \|x_j - \mu_i\|^2$$

We can now formulate this problem as a likelihood problem. There are k unknown clusters $S_1, \ldots, S_k$. The points in S_i are generated using $MVN(\mu_i, I)$, where I is the identity matrix. Each point x_i originates from a cluster c_i.This implies that our parameters are $\theta = (c_1, \ldots, c_n, \mu_1, \ldots, \mu_k)$. The log-likelihood is

$$\ell(\theta; x_1, \ldots, x_n) = \text{constant} - \sum_{i=1}^{n} \|x_i - \mu_{c_i}\|^2$$

Therefore, maximizing the likelihood is equivalent to minimizing the objective function. (The value of μ_i is selected to minimize the loss of the points in cluster i, and is set to the average x_i due to the minimization.)

3.4.1 Mixture of Gaussians

In k-means we assumed that each point has to be classified to a specific cluster. This is a “hard” decision, since we need to decide for each point a single cluster. We can relax this by having a “soft” decision, where a point will have a distribution over the clusters it originates from. This lead to a mixture of Gaussian model. In the *mixture of Gaussians* we have k Gaussian distribution, and a mixing parameter. The mixing parameter gives a probability to each cluster.To generate a point, we sample a Gaussian given the mixture parameter, and then sample the selected Gaussian to generate the point (See Figures 3.4 and 3.5 for an example. The algorithm is unaware of the origin of the points. The data in the figure was generated with $S_1 \sim MVN(\mu_1, \Sigma_1)$ and $S_2 \sim MVN(\mu_2, \Sigma_2)$ where $\mu_1 = (10, 10)$,

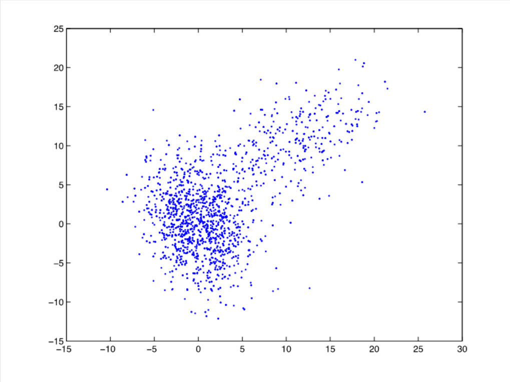

Figure 3.4: unlabeled points

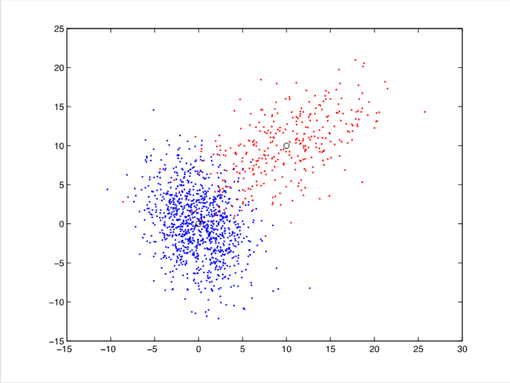

Figure 3.5: two clusters, red and blue and their centers

$\Sigma_1 = \begin{pmatrix} 29.25 & 13.5 \\ 13.5 & 20.25 \end{pmatrix}$ and $\mu_2 = (0,0)$ and $\Sigma_2 = \begin{pmatrix} 9 & -3.3 \\ -3.3 & 18 \end{pmatrix}$. The mixing parameters are $p_1 = 0.25$ and $p_2 = 0.75$.)

For simplicity we derive the analysis for a univariate normal distribution,which would be easier to demonstrate the concepts, and latter we generalize to multivariate normal distributions.

We have k unknown clusters $S-1, \ldots, S_k$, where $S_i \sim N(\mu_i, \sigma_i^2)$. Each point x_i originates from cluster j with probability p_j.

The density function for cluster j is

$$f_j(x) = \frac{1}{\sqrt{2\pi\sigma_j^2}} e^{-\frac{(x-\mu)^2}{2\sigma_j^2}}$$

The likelihood function is,

$$L((\vec{p}, \vec{\mu}, \vec{\sigma}); \vec{x}) = \prod_{i=1}^{n} \sum_{j=1}^{k} p_j f_j(x_i)$$

where we use the fact that the samples are i.i.d.

We can introduce auxiliary variables $a_{i,j}$ for every point i and cluster j, where we have

$a_{i,j} > 0$ and $\sum_{j=1}^{n} a_{i,j} = 1$. We can now lower bound the log-likelihood as follows.

$$\begin{aligned}
\log L((\vec{p}, \vec{\mu}, \vec{\sigma}); \vec{x}) &= \sum_{i=1}^{n} \log \left(\sum_{j=1}^{k} p_j f_j(x_i) \right) \\
&= \sum_{i=1}^{n} \log \left(\sum_{j=1}^{k} a_{ij} \frac{p_j f_j(x_i)}{a_{ij}} \right) \\
&\geq \sum_{i=1}^{n} \sum_{j=1}^{k} a_{ij} \log(p_j f_j(x_i)) - a_{ij} \log(a_{ij})
\end{aligned}$$

The inequality follows from Jensen's inequality, that for a concave function F states that $E[F(x)] \leq F(E[X])$ for a non-negative random variable X. the logarithmic function is concave, and the $\{a_{i,j}\}_{j=1}^{k}$ is a distribution.

> To show the Jensen's inequality, recall that if F is concave then $F(\lambda x_1 + (1-\lambda)x_2) \geq \lambda F(x_1) + (1-\lambda)F(x_2)$. Similarly, $F(\sum_{i=1}^{n} p_i x_i) \geq \sum_{i=1}^{n} p_i F(x_i)$. This essentially proves the Jensen's inequality for discrete random variables.

We can now describe an instance of the expectation-maximization (EM) algorithm for the mixture of Gaussians. The algorithm starts with an initialization $(\vec{\mu}^0, \vec{\sigma}^0, \vec{p}^0)$.

In iteration $t+1$ we have:

$$\begin{aligned}
a_{ij}^{t+1} =& Pr(x_i \in S_j \mid \vec{p}^t, \vec{\mu}^t, \vec{\sigma}^t) = \frac{p_j^t f_j^t(x_i)}{\sum_{m=1}^{k} p_m^t f_m^t(x_i)} \\
(\vec{p}^{t+1}, \vec{\mu}^{t+1}, \vec{\sigma}^{t+1}) =& \arg\max_{\vec{\mu}, \vec{\sigma}, \vec{p}} \sum_{i=1}^{n} \sum_{j=1}^{k} a_{ij}^{t+1} \log(p_j f_j(x_i))
\end{aligned}$$

In the maximization we dropped the terms $a_{ij} \log(a_{i,j})$ since they do not influence the maximization.

The maximization factors out nicely. For the part involving p_j we have

$$\begin{aligned}
\vec{p}^{t+1} =& \arg\max_{\vec{p}} \sum_{i=1}^{n} \sum_{j=1}^{k} a_{ij}^{t+1} \log(p_j) \\
p_j^{t+1} =& \frac{\sum_{i=1}^{n} a_{ij}}{n}
\end{aligned}$$

where the maximization solution is identical to that in lecture 2, for multinomial distribution.

For the μ and σ maximization we have,

$$\begin{aligned}(\vec{\mu}^{t+1}, \vec{\sigma}^{t+1}) =& \arg\max_{\vec{\mu},\vec{\sigma},\vec{p}} \sum_{i=1}^{n}\sum_{j=1}^{k} a_{ij}^{t+1}\log(f_j(x_i)) \\ \mu_j^{t+1} =& \frac{\sum_{i=1}^{n} a_{ij}x_i}{\sum_{i=1}^{n} a_{i,j}} \\ \sigma_j^{t+1} =& \frac{\sum_{i=1}^{n} a_{ij}(x_i-\mu_j^{t+1})^2}{\sum_{i=1}^{n} a_{ij}}\end{aligned}$$

where the maximization is form the derivation of the maximum likelihood for normal distributions (see lecture 2).

We can define a g function:

$$g_a(\vec{p},\vec{\mu},\vec{\sigma}) := \sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij}\log(p_j f_j(x_i)) - a_{ij}\log(a_{ij})$$

Note that g depends on a, but we consider as as constants in g.

For the log-likelihood, by construction, for any a we have,

$$\log L((\vec{p},\vec{\mu},\vec{\sigma});\vec{x}) \geq g_a(\vec{p},\vec{\mu},\vec{\sigma})$$

We like to find an a such that

$$\log L((\vec{p},\vec{\mu},\vec{\sigma});\vec{x}) = g_a(\vec{p},\vec{\mu},\vec{\sigma})$$

This would hold if all the terms are identical in the log likelihood. Namely,

$$\frac{p_j^t f_j^t(x_i)}{a_{i,j}^{t+1}} = \text{constant}$$

For this we need that $a_{i,j}^{t+1} \propto p_j^t f_j^t(x_i)$. Since we need it to be a distribution, we set

$$a_{i,j}^{t+1} = \frac{p_j^t f_j^t(x_i)}{\sum_{r=1}^{k} p_r^t f_r^t(x_i)} = \Pr[c_i = j|\theta^t, x_i]$$

Given that this is the way we select a, we have the following,

$$\begin{aligned}\log L((\vec{p}^{t+1},\vec{\mu}^{t+1},\vec{\sigma}^{t+1});\vec{x}) \geq& g_a(\vec{p}^{t+1},\vec{\mu}^{t+1},\vec{\sigma}^{t+1}) \\ \geq& g_a(\vec{p}^{t},\vec{\mu}^{t},\vec{\sigma}^{t}) \\ =& \log L((\vec{p}^{t},\vec{\mu}^{t},\vec{\sigma}^{t});\vec{x})\end{aligned}$$

where the first inequality holds for any a. The second inequality follows since $(\vec{p}^{t+1},\vec{\mu}^{t+1},\vec{\sigma}^{t+1})$ is the solution to the maximization, given a. The last equality follows from the fact that we selected a the way we did.

This implies that in every iteration the log-likelihood can only increase.

3.5 Expectation Maximization (EM)

We now generalize the EM algorithm in general. Let D be the given data, θ the parameters to be estimated, Z the missing (latent) variables. (In the mixture of Gaussians the Z is the probabilities that x_i was generated by each cluster c_j.)

The EM algorithm alternates between an E-step and an M-step. In the E-step we compute an expectation over the latent variable Z.

$$\textbf{E-step} \quad Q(\theta|\theta^t) = E_{Z|D,\theta^t}[\log \Pr(D, Z|\theta)]$$

The input to the Q function is θ, a complete model. The output is the log-likelihood, whether the expectation is taken over the latent variables. Many times the Q function factors nicely between the different parameters, as in the mixture if Gaussians.

The M-step, computes a maximization of the Q function.

$$\textbf{M-step} \quad \theta^{t+1} = \arg\max_{\theta} Q(\theta|\theta^t)$$

We can now compute the change in the log likelihood,

$$\begin{aligned}
\log Pr(D \mid \theta) &= \log\left(\sum_z Pr(D, z \mid \theta)\right) \\
&= \log\left(\sum_z a_z \frac{Pr(D, z \mid \theta)}{a_z}\right) \\
&\geq \sum_z a_z \log\left(Pr(D, z \mid \theta)\right) - \sum_z a_z \log(a_z) \\
&= Q(\theta \mid \theta^t) - \text{constant}
\end{aligned}$$

As before,

$$\begin{aligned}
\log Pr(D \mid \theta^{t+1}) &\geq Q(\theta^{t+1} \mid \theta^t) - \text{constant} \\
&\geq Q(\theta^t \mid \theta^t) - \text{constant} = \log Pr(D \mid \theta^t)
\end{aligned}$$

where the first inequality holds in general. The second inequality follows since θ^{t+1} is the solution to the maximization. The last equality follows from the fact that we selected a the way we did.

As before, we can set

$$g(\theta) = Q(\theta|\theta^t) - \sum_z a_z \log(a_z)$$

Again we have

$$\log \Pr(D|\theta^{t+1}) \geq g(\theta^{t+1}) \geq g(\theta^t) = \log \Pr(D|\theta^t)$$

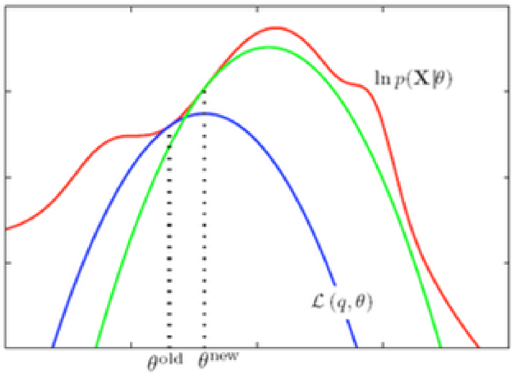

Figure 3.6: EM algorithm

This implies that the likelihood can not decrease.

In Figure 3.6 we can see an iteration of the algorithm. The red curve is the log-likelihood. Given a parameter θ^t we set a function $g(\theta)$ which equals the log-likelihood at θ^t. Maximizing $g(\theta)$ gives θ^{t+1}, which leads to an increase in the log-likelihood.

Remarks on the EM algorithm:

- No guarantee of optimization to local maximum. We are guarantee no to decrease, but we might get stuck at a saddle point.
- No guarantee of running times. The improvements might be very slow.also, the magnitude of the improvements need not be monotone.
- Often it takes many iterations to converge.
- Efficiency: no matrix inversion is needed (e.g., in Newton). Generalized EM - no need to find the max in the M-step.
- Easy to implement. Especially in cases where there are close form solution for the E and M steps.
- Numerical stability.
- Monotone - it is easy to ensure correctness in EM. Simply check that the likelihood increases.
- Interpretation - provides interpretation for the latent variables.