CS 181 Spring 2017 Section 4
Solution

1 Naive Bayes

A Naive Bayes classifier is a generative classifier, meaning that we are building a model for the joint distribution $p(\mathbf{x}, y)$. What we are actually interested in during classification is $p(y|\mathbf{x})$, which is related to the joint $p(\mathbf{x}, y)$ through Bayes' Rule:

$$p(y|\mathbf{x}) = \frac{p(y)p(\mathbf{x}|y)}{p(\mathbf{x})} \tag{1}$$

$$\propto p(y)p(\mathbf{x}|y) \tag{2}$$

The $p(y)$ is the prior over the classes. In the case of binary classification, $p(y;\theta) \sim \text{Bern}(\theta)$. $p(\mathbf{x})$ is constant and therefore we can disregard it as a normalizing constant, allowing us to go from (1) to (2).

The "naive" of Naive Bayes means that we assume that each feature in $\mathbf{x}$ is independently distributed, conditioned on the class. In the case of Multinomial Naive Bayes, we have $p(\mathbf{x}|y;\boldsymbol{\pi}_0, \boldsymbol{\pi}_1) \propto \prod_{j=1}^m \pi_{yj}^{x_j}$.

If we are doing MLE, we then find the values of our parameters $\theta, \boldsymbol{\pi}_0, \boldsymbol{\pi}_1$ that maximize the log-likelihood:

$$\max_{\theta,\boldsymbol{\pi}_0,\boldsymbol{\pi}_1} \sum_{i=1}^n \ln p(\mathbf{x}_i, y_i) = \max_{\boldsymbol{\pi}_1,\boldsymbol{\pi}_2} \sum_{i=1}^n \ln p(\mathbf{x}_i|y_i;\boldsymbol{\pi}_0,\boldsymbol{\pi}_1) + \max_{\theta} \sum_{i=1}^n \ln p(y_i;\theta) \tag{3}$$

Why is it that Naive Bayes is a linear classifier? When we classify a new example $\mathbf{x}$, we choose the class that is more likely by comparing $p(y|\mathbf{x})$ for every possible class value of y. In particular, for a binary classifier, we can compute the value of $h(\mathbf{x})$, which is the difference of the log of $p(y=1|\mathbf{x})$ and the log of $p(y=0|\mathbf{x})$:

$$\begin{aligned}
h(\mathbf{x}) &= [\ln p(\mathbf{x}|y=1) + \ln p(y=1)] - [\ln p(\mathbf{x}|y=0) + \ln p(y=0)] && (4)\\
&= \big[\ln \prod_{j=1}^m \pi_{1j}^{x_j} - \ln \prod_{j=1}^m \pi_{0j}^{x_j}\big] + \big[\ln\theta - \ln(1-\theta)\big] && (5)\\
&= \sum_{j=1}^m x_j \ln\frac{\pi_{1j}}{\pi_{0j}} + \ln\left(\frac{\theta}{1-\theta}\right) && (6)\\
&= \mathbf{x}^\top \mathbf{ln}(\frac{\boldsymbol{\pi}_1}{\boldsymbol{\pi}_0}) + \ln\left(\frac{\theta}{1-\theta}\right), && (7)
\end{aligned}$$

where $\mathbf{ln}(\mathbf{q})$ for vector $\mathbf{q}$ applies ln elementwise, and we write $\boldsymbol{\pi}_1/\boldsymbol{\pi}_0$ to mean elementwise division. The resulting equation is in the same form as linear regression, and in particular the decision boundary $h(\mathbf{x}) = 0$ is linear in $\mathbf{x}$. With different Naive Bayes models, the only difference would be the corresponding weight and bias terms.

1. **Redundant Features in Naive Bayes**

Suppose, as in Bishop 4.2.3, that we use a Naive Bayes classifier to classify binary feature vectors $\mathbf{x}_i \in \mathbb{R}^m$ into two classes. The class conditional distributions will then be of the form

$$p(\mathbf{x} \,|\, y = C_k) = \prod_{i=1}^{m} \pi_{ki}^{x_i} \, (1 - \pi_{ki})^{(1-x_i)}, \qquad \text{(Bishop 4.81)}$$

where $x_i \in \{0, 1\}$, and $\pi_{ki} = p(x_i = 1 \,|\, y = C_k)$. This is a Bernoulli Naive Bayes, where all the features are binary instead of representing count data, as in the Multinomial case. Assume also that the class priors are $p(y = C_1) = p(y = C_2) = \frac{1}{2}$.

a. How is the quantity $\ln(p(y = C_1 \,|\, \mathbf{x})/p(y = C_2 \,|\, \mathbf{x}))$ used for classification of a new example $\mathbf{x}$?

b. If $m = 1$ (i.e., there is only one feature), use the equations above to write out $\ln \frac{p(y=C_1 \,|\, x)}{p(y=C_2 \,|\, x)}$ for a single binary feature x.

c. Now suppose we change our feature representation so that instead of using just a single feature, we use two redundant features (i.e., two features that always have the same value). With this feature representation, instead of x we will use $\mathbf{x} = [x, x]^\top$. What is $\ln \frac{p(y=C_1 \,|\, \mathbf{x})}{p(y=C_2 \,|\, \mathbf{x})}$ in terms of the value for $\ln \frac{p(y=C_1 \,|\, x)}{p(y=C_2 \,|\, x)}$ you calculated in part (a.)?

d. Is this a bug or a feature?

Solution

a. We will predict class C_1 if $p(y = C_1 \,|\, \mathbf{x}) \geq p(y = C_2 \,|\, \mathbf{x})$, and class C_2 otherwise. Equivalently, we will predict C_1 if $\ln(p(y = C_1 \,|\, \mathbf{x})/p(y = C_2 \,|\, \mathbf{x})) \geq 0$, and C_2 otherwise.

b. Because the class priors are the same and the denominators cancel, we have $p(y = C_1 \,|\, x)/p(y = C_2 \,|\, x) = p(y = C_1)p(x \,|\, y = C_1)/p(y = C_2)p(x \,|\, y = C_2) = p(y = C_1 \,|\, x)/p(y = C_2 \,|\, x)$, and we have:

$$\begin{aligned}\ln \frac{p(y = C_1 \,|\, x)}{p(y = C_2 \,|\, x)} &= \ln \frac{\pi_{11}^{x}\,(1-\pi_{11})^{(1-x)}}{\pi_{21}^{x}\,(1-\pi_{21})^{(1-x)}} \\ &= x \ln \pi_{11} + (1-x)\ln(1-\pi_{11}) - x\ln\pi_{21} - (1-x)\ln(1-\pi_{21})\end{aligned}$$

c. Because the two features are identical, we will have

$$\begin{aligned}\ln \frac{p(y = C_1 \,|\, \mathbf{x})}{p(y = C_2 \,|\, \mathbf{x})} &= \ln \frac{\left(\pi_{11}^{x}\,(1-\pi_{11})^{(1-x)}\right)^2}{\left(\pi_{21}^{x}\,(1-\pi_{21})^{(1-x)}\right)^2} \\ &= \ln \left[\left(\frac{\pi_{11}^{x}\,(1-\pi_{11})^{(1-x)}}{\pi_{21}^{x}\,(1-\pi_{21})^{(1-x)}}\right)^2\right] \\ &= 2\ln \frac{p(y = C_1 \,|\, x)}{p(y = C_2 \,|\, x)}\end{aligned}$$

d. This is a feature! We see that the classifier with the two identical features has exactly the same behavior as the classifier with just a single feature. In particular,

$$\ln \frac{p(y = C_1 \,|\, \mathbf{x})}{p(y = C_2 \,|\, \mathbf{x})} \geq 0 \quad \Leftrightarrow \quad \ln \frac{p(y = C_1 \,|\, x)}{p(y = C_2 \,|\, x} \geq 0 \tag{8}$$

Note: it does not matter that there is a new constant 2 in front of the expression. Only the sign is important for classification.

End Solution

2 Gaussian Probabilistic Classification

Shapes of Decision Boundaries - Part I

Consider now a generative model with $c > 2$ classes, and output label $\mathbf{y}$ encoded as a "one hot" vector of length c.
We adopt class prior $p(\mathbf{y} = C_k; \boldsymbol{\pi}) = \pi_k$ for all $k \in \{1, \ldots, c\}$ (where π_k is a parameter of the prior).
Let $p(\mathbf{x} \mid \mathbf{y} = C_k)$ denote the class-conditional density of features $\mathbf{x}$ (in this case for class C_k). Let the class-conditional probabilities be Gaussian distributions

$$p(\mathbf{x} \mid \mathbf{y} = C_k) = \mathcal{N}(\mathbf{x} \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k), \text{ for } k \in \{1, \ldots, c\} \tag{9}$$

We will predict the class of a new example $\mathbf{x}$ as the class with the highest conditional probability, $p(\mathbf{y} = C_k \mid \mathbf{x})$.
Luckily, a little bird came to the window of your dorm, and claimed that you can classify an example $\mathbf{x}$ by finding the class that maximizes the following function:

$$f_k(\mathbf{x}) = \ln(\pi_k) - \frac{1}{2}\ln(|\boldsymbol{\Sigma}_k|) - \frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_k)^T \boldsymbol{\Sigma}_k^{-1}(\mathbf{x} - \boldsymbol{\mu}_k).$$

Derive this formula by comparing two different classes' conditional probabilities. What can we claim about the shape of the decision boundary given this formula?

Solution

Let's start with the conditional probability:

$$p(\mathbf{y} = C_k \mid \mathbf{x}) = \frac{p(\mathbf{x} \mid \mathbf{y} = C_k)p(\mathbf{y} = C_k)}{\sum_\ell^c p(\mathbf{x} \mid \mathbf{y} = C_\ell)p(\mathbf{y} = C_\ell)} \tag{10}$$

{We can take out the denominator since it will be the same for each class}

$$\propto p(\mathbf{x} \mid \mathbf{y} = C_k)p(\mathbf{y} = C_k) \tag{11}$$

$$= \frac{1}{(2\pi)^{\frac{m}{2}}|\boldsymbol{\Sigma}_k|^{\frac{1}{2}}} \exp\left(-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_k)^T \boldsymbol{\Sigma}_k^{-1}(\mathbf{x} - \boldsymbol{\mu}_k)\right)\pi_k \tag{12}$$

{We can factor out constants that will be shared across classes}

$$\propto \frac{\pi_k}{|\boldsymbol{\Sigma}_k|^{\frac{1}{2}}} \exp\left(-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_k)^T \boldsymbol{\Sigma}_k^{-1}(\mathbf{x} - \boldsymbol{\mu}_k)\right) \tag{13}$$

{We take the logarithm, which is a monotonically increasing function and won't change the maximum across classes}

$$\propto \ln(\pi_k) - \frac{1}{2}\ln(|\boldsymbol{\Sigma}_k|) - \frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_k)^T \boldsymbol{\Sigma}_k^{-1}(\mathbf{x} - \boldsymbol{\mu}_k) \tag{14}$$

Now, what can we say about the shape of our decision boundary? $(\mathbf{x} - \boldsymbol{\mu}_k)^T \boldsymbol{\Sigma}_k^{-1}(\mathbf{x} - \boldsymbol{\mu}_k)$ gives us intuition about the shape of our decision boundary. We see it is quadratic.

For further intuition, we can look at the other two terms to see what affects the probability of an example being classified to a given class. As the prior π_k increases, we see that the probability increases, which fits our intuition. As the covariance matrix $\mathbf{\Sigma}_k$ increases, we see that the probability decreases, which also fits our intuition.

End Solution

Shapes of Decision Boundaries - Part II

Let's say the little bird comes back and now tells you that every class has the same covariance matrix, and so $\mathbf{\Sigma}_\ell = \mathbf{\Sigma}'_\ell$ for all classes C_ℓ and $C_{\ell'}$. Simplify this formula down further. What can we claim about the shape of the decision boundaries now?

Solution

Here was our original formula.

$$\ln(\pi_k) - \frac{1}{2}\ln(|\mathbf{\Sigma}_k|) - \frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_k)^T\mathbf{\Sigma}_k^{-1}(\mathbf{x}-\boldsymbol{\mu}_k) \tag{15}$$

Given that all of the classes have the same covariance matrix, which we will write as $\mathbf{\Sigma}$, we have:

$$\begin{aligned}
f_k(\mathbf{x}) &= \ln(\pi_k) - \frac{1}{2}\ln(|\mathbf{\Sigma}|) - \frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_k)^T\mathbf{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu}_k) &(16)\\
&= \ln(\pi_k) - \frac{1}{2}\ln(|\mathbf{\Sigma}|) - \frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_k)^T(\mathbf{\Sigma}^{-1}\mathbf{x}-\mathbf{\Sigma}^{-1}\boldsymbol{\mu}_k) &(17)\\
&= \ln(\pi_k) - \frac{1}{2}\ln(|\mathbf{\Sigma}|) - \frac{1}{2}(\mathbf{x}^T\mathbf{\Sigma}^{-1}\mathbf{x}-\mathbf{x}^T\mathbf{\Sigma}^{-1}\boldsymbol{\mu}_k-\boldsymbol{\mu}^T\mathbf{\Sigma}^{-1}\mathbf{x}+\boldsymbol{\mu}^T\mathbf{\Sigma}^{-1}\boldsymbol{\mu}_k) &(18)\\
&= \ln(\pi_k) - \frac{1}{2}\ln(|\mathbf{\Sigma}|) - \frac{1}{2}(\mathbf{x}\mathbf{\Sigma}^{-1}\mathbf{x}^T-2\mathbf{x}^T\mathbf{\Sigma}^{-1}\boldsymbol{\mu}_k+\boldsymbol{\mu}_k^T\mathbf{\Sigma}^{-1}\boldsymbol{\mu}_k) &(19)\\
&= \ln(\pi_k) - \frac{1}{2}\ln(|\mathbf{\Sigma}|) - \frac{1}{2}\mathbf{x}\mathbf{\Sigma}^{-1}\mathbf{x}^T+\mathbf{x}^T\mathbf{\Sigma}^{-1}\boldsymbol{\mu}_k-\frac{1}{2}\boldsymbol{\mu}_k^T\mathbf{\Sigma}^{-1}\boldsymbol{\mu}_k &(20)
\end{aligned}$$

{We can drop $-\frac{1}{2}\ln(|\mathbf{\Sigma}|)$ and $-\frac{1}{2}\mathbf{x}\mathbf{\Sigma}^{-1}\mathbf{x}^T$ since they are class independent}

$$\propto \ln(\pi_k) + \mathbf{x}^T\mathbf{\Sigma}^{-1}\boldsymbol{\mu}_k - \frac{1}{2}\boldsymbol{\mu}_k^T\mathbf{\Sigma}^{-1}\boldsymbol{\mu}_k \tag{21}$$

Thus, we conclude that we can adopt function

$$\tilde{f}_k(\mathbf{x}) = \mathbf{x}^T\mathbf{\Sigma}^{-1}\boldsymbol{\mu}_k - \frac{1}{2}\boldsymbol{\mu}_k^T\Sigma^{-1}\boldsymbol{\mu}_k + \ln(\pi_k) \tag{22}$$

Looking at this formula, we can see that our decision boundaries are no longer quadratic but linear. We've proven that if the classes share the same covariance matrix, our decision boundaries will be linear!

End Solution

3 Neural Networks

Recall that in the case of binary classification, we can think about neural network as being equivalent to logistic regression with parameterized, adaptive basis functions.

Let's think about a neural network binary classifier with $\mathbf{x} \in \mathbb{R}^2$ (and thus $m = 2$ features) and with two a single hidden layer.

For the activation function, we use the *ReLU* function defined by the following:

$$\text{ReLU}(z) = \begin{cases} z & \text{if } z > 0 \\ 0 & \text{otherwise} \end{cases} \tag{23}$$

Let's consider a function h defined by the following:

$$\begin{aligned} h(\mathbf{x}) &= \mathbf{w}^\mathsf{T}\phi(\mathbf{x}) + w_0 && (24) \\ &= \mathbf{w}^\mathsf{T}\mathbf{ReLU}\left(\mathbf{W}^1\mathbf{x} + \mathbf{w}_0^1\right) + w_0, && (25) \end{aligned}$$

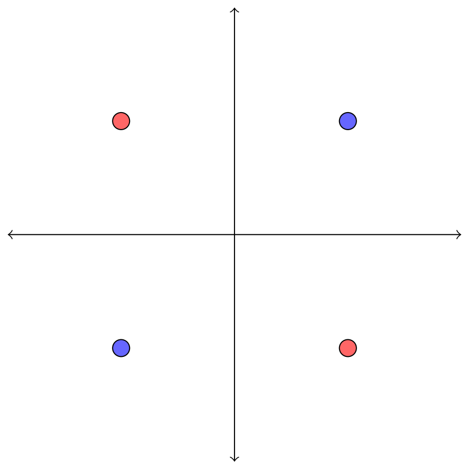
where example $\mathbf{x} \in \mathbb{R}^{2\times1}$, weights in the hidden layer $\mathbf{W}^1 \in \mathbb{R}^{2\times2}$, bias in the hidden layer $\mathbf{w}_0^1 \in \mathbb{R}^{2\times1}$, output weight vector $\mathbf{w} \in \mathbb{R}^{2\times1}$, and output bias $w_0 \in \mathbb{R}$.

Suppose we want to fit the following data:

$$\mathbf{x}_1 = \begin{pmatrix} 1 \\ 1 \end{pmatrix} \; y_1 = 1, \quad \mathbf{x}_2 = \begin{pmatrix} 1 \\ -1 \end{pmatrix} \; y_2 = -1$$

$$\mathbf{x}_3 = \begin{pmatrix} -1 \\ 1 \end{pmatrix} \; y_3 = -1, \quad \mathbf{x}_4 = \begin{pmatrix} -1 \\ -1 \end{pmatrix} \; y_4 = 1$$

This looks as follows (blue = 1, red = -1):



Why can't we solve this problem with a linear classifier? What values of parameters $\mathbf{W}^1, \mathbf{w}_0^1, \mathbf{w}$, and w_0 will allow the neural network to solve the problem?

Hint: Think carefully about why we need a nonlinearity in finding a basis function for the examples. What can the ReLU do for us? What does it do to various kinds of vectors?

Solution

We can solve the problem with

$$\mathbf{W}^1 = \begin{pmatrix} 1 & 1 \\ -1 & -1 \end{pmatrix}, \quad \mathbf{w}_0^1 = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

$$\mathbf{w} = \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \quad w_0 = -1$$

How does this solve the problem? Note that

$$h(\mathbf{x}_1) = \begin{pmatrix} 1 & 1 \end{pmatrix} \mathbf{ReLU} \begin{pmatrix} 2 \\ -2 \end{pmatrix} - 1 = 1$$

$$h(\mathbf{x}_2) = \begin{pmatrix} 1 & 1 \end{pmatrix} \mathbf{ReLU} \begin{pmatrix} 0 \\ 0 \end{pmatrix} - 1 = -1$$

$$h(\mathbf{x}_3) = \begin{pmatrix} -1 & 1 \end{pmatrix} \mathbf{ReLU} \begin{pmatrix} 0 \\ 0 \end{pmatrix} - 1 = -1$$

$$h(\mathbf{x}_4) = \begin{pmatrix} 1 & 1 \end{pmatrix} \mathbf{ReLU} \begin{pmatrix} -2 \\ 2 \end{pmatrix} - 1 = 1$$

This works because we have a ReLU! Having a nonlinearity in the activation function allows us to find a classifier for these examples.

End Solution