COMP 562 – Lecture 8

Multivariate Gaussian Distribution -- Dependent Case

Suppose we have p standard random variables (0 mean, unit variance)

$$z_i \sim \mathcal{N}(0,1), \qquad i = 1, \dots p$$

and we are given a vector $\boldsymbol{\mu}$ of length n and a full-rank matrix A of size $p \times p$

Distribution of $\mathbf{x} = A\mathbf{z} + \mu$ is

$$p(\mathbf{x}) = (2\pi)^{-\frac{p}{2}} (\det \Sigma)^{-\frac{1}{2}} \exp\left\{ \frac{1}{2} (\mathbf{x} - \boldsymbol{\mu})^T \Sigma^{-1} (\mathbf{x} - \boldsymbol{\mu}) \right\}$$

where $\Sigma = AA^T$.

- μ is **mean** of the Gaussian
- Σ is **covariance** matrix

Maximum Likelihood Estimates of Mean and Covariance

Given data $\{\mathbf{x}_i \in \mathbb{R}^N | i = 1, \dots, N\}$ maximum likelihood estimates (MLE) of mean and covariance are:

$$\boldsymbol{\mu}^{\text{MLE}} = \frac{1}{N} \sum_{i=1}^{N} \mathbf{x}_i$$

$$\Sigma^{\text{MLE}} = \frac{1}{N} \sum_{i=1}^{N} \underbrace{\left(\mathbf{x}_i - \boldsymbol{\mu}^{\text{MLE}}\right) \left(\mathbf{x}_i - \boldsymbol{\mu}^{\text{MLE}}\right)^T}_{\text{a matrix of size } p \times p}$$

Dimensionality

- $\boldsymbol{\mu}^{\text{MLE}}$ is of same dimension as a single data point $p \times 1$.
- Σ^{MLE} is a matrix of size $p \times p$

Note that $\mathbf{x}\mathbf{x}^T$ and $\mathbf{x}^T\mathbf{x}$ are not the same, former is a matrix, latter is a scalar

Generative Models for Classification

There are two ways to factorize joint probability of labels and features

$$p(y, \mathbf{x}|\theta) = p(y|\mathbf{x}, \theta)p(\mathbf{x}|\theta) = p(\mathbf{x}|y, \theta)p(y|\theta)$$

The second one given us a simple process to *GENERATE* data:

1. First select label according $p(y|\theta)$, say it was c
2. Now generate features $p(\mathbf{x}|y = c, \theta)$

Once we have such a model we can obtain the conditional probability $p(y|\mathbf{x})$ using Bayes rule

$$p(y = c|\mathbf{x}) = \frac{p(y = c|\theta)p(\mathbf{x}|y = c, \theta)}{\sum_k p(y = k|\theta)p(\mathbf{x}|y = k, \theta)}$$

and we can predict label for a new feature vector $\mathbf{x}$

$$p(\mathbf{x}|y, \theta) = \prod_j p(x_j|y, \theta)$$

This assumption **Conditional Independence of Features** underlies the **Naive Bayes** method

Naive Bayes

$$p(y = c|\pi) = \pi_c$$
$$p(\mathbf{x}|y = c, \theta) = \prod_j p(x_j|y = c, \theta_{j,c})$$

Parameters are

- π_c prior probability that a sample comes from the class c
- $\theta_{j,c}$ parameters for the j^{th} feature for class c

In general, there are many variants of Naive Bayes, you can choose different distributions for $p(x_j|y = c)$

- Gaussian -- continuous features
- Bernoulli -- binary features
- Binomial -- count of positive outcomes
- Categorical -- discrete features
- Multinomial -- count of particular discrete outcomes

Naive Bayes with Gaussian Features

We will assume that

$$x_j|y_c, \theta \sim \mathcal{N}\left(\theta_{j,c}, \sigma^2\right)$$

Each feature is Gaussian distributed around class specific mean and with shared spherical variance

Let's take a look at the data we generated earlier and read-off these parameters

joint Log-likelihood

$$\begin{aligned}
\mathcal{LL}(\theta, \pi|\mathbf{y}, X) &= \sum_i \log p(y_i, \mathbf{x}_i|\theta, \pi) && \text{definition of likelihood} \\
&= \sum_i \log p(y_i|\pi) + \log p(\mathbf{x}_i|y_i, \theta) && \text{factorization } p(y, \mathbf{x}) = p(y)p(\mathbf{x}|y) \\
&= \sum_i \log p(y_i|\pi) + \log \prod_j p(x_{j,i}|y_i, \theta_j) && \text{Naive Bayes assumption} \\
&= \sum_i \log p(y_i|\pi) + \sum_j \log p(x_{j,i}|y_i, \theta_j)
\end{aligned}$$

Note that we have not yet used our assumptions about distribution of $x_{j,i}$

Learning parameters for Naive Bayes with Gaussian features

joint Log-likelihood

$$\begin{aligned}
\mathcal{LL}(\theta, \pi|\mathbf{y}, X) &= \sum_i \left[\log p(y_i|\pi) + \sum_j \log p(x_{j,i}|y_i, \theta_j)\right] \\
&= \sum_i \left[\log \pi_{y_i} + \sum_j \log \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left\{-\frac{1}{2\sigma^2}(x_{j,i} - \theta_{j,y_i})^2\right\}\right] \\
&= \sum_i \left[\log \pi_{y_i} - \frac{1}{2}\sum_j \log(2\pi\sigma^2) - \frac{1}{2\sigma^2}\sum_j (x_{j,i} - \theta_{j,y_i})^2\right]
\end{aligned}$$

Note that parameters π_c and $\theta_{i,c}$ are only used for samples that belong to class c ($y_i = c$)

Hence, we can learn of parameters for each class separately

Learning Parameters for Naive Bayes with Gaussian Features

Closed form estimates for parameters are

$$\pi_c = \frac{\sum_i [y_i = c]}{\sum_i 1} \qquad \text{frequency of class } c \text{ in training data}$$

$$\theta_{j,c} = \frac{\sum_i [y_i = c] x_{i,j}}{\sum_i [y_i = c]} \qquad \text{average of feature } j \text{ among samples in class } c$$

$$\sigma = \frac{\sum_i (x_{j,i} - \theta_{j,y_i})^2}{\sum_i 1} \qquad \text{variance across all features}$$

Note $[x]$ is an indicator function, defined as

$$[x] = \begin{cases} 1 & \text{if } x \text{ is true} \\ 0 & \text{otherwise} \end{cases}$$

Class Prediction using Naive Bayes with Gaussian Features

Recall that

$$\underset{c}{\operatorname{argmax}}\, p(y = c|\mathbf{x}) = \underset{c}{\operatorname{argmax}}\, \log p(y = c|\theta) + \log p(\mathbf{x}|y = c, \theta)$$

After a little bit more manipulation

$$\log p(y = c|\mathbf{x}, \theta, \pi) = \log \pi_c - \sum_j \frac{1}{2\sigma^2}(x_{j,i} - \theta_{j,c})^2 + \text{const.}$$

Predicted class

$$y^* = \underset{c}{\operatorname{argmax}}\, \log \pi_c - \underbrace{\sum_j (x_{j,i} - \theta_{j,c})^2}_{\text{distance to class center}} + \text{const.}$$

- **Larger** K means **less bias** towards overestimating the true expected error (as training folds will be closer to the total dataset) but **typically higher variance and higher running time** (as you are getting closer to the limit case: Leave-One-Out CV)
- If cross-validation were averaging independent estimates, then with large K, one should see relatively lower variance between models; however, this is not true when training sets are highly correlated, which is what we typically deal with

Classification Performance -- Prediction Rate

Sensitivity, Recall, or True Positive Rate (TPR)

$$TPR = \frac{TP}{P} = \frac{TP}{TP + FN} = 1 - FNR$$

Specificity, Selectivity or True Negative Rate (TNR)

$$TNR = \frac{TN}{N} = \frac{TN}{TN + FP} = 1 - FPR$$

False Positive Rate (FPR)

$$FPR = \frac{FP}{N} = \frac{FP}{TN + FP} = 1 - TNR$$

False Negative Rate (FNR)

$$FNR = \frac{FN}{P} = \frac{FN}{TP + FN} = 1 - TPR$$

These measures are computed from either +ve or -ve group, hence they don't depend on classes balance (prevalence = $\frac{P}{P+N}$)

Precision or Positive Predictive Value (PPV)

$$PPV = \frac{TP}{TP + FP}$$

Negative Predictive Value (NPV)

$$NPV = \frac{TN}{TN + FN}$$

Accuracy (ACC)

$$ACC = \frac{TP + TN}{P + N} = \frac{TP + TN}{TP + FN + TN + FP}$$

These measures are computed from both +ve and -ve groups, hence they depend on classes balance (prevalence = $\frac{P}{P+N}$)

Classification Performance -- ROC Curves

Predictions are based on a cutoff

$$p(y = 1|\mathbf{x}) > \tau$$

where τ is typically 0.5

This particular cutoff will result in a specific prediction rates; however, you may prefer to tradeoff false positives for false negatives -- health industry does

Classification Performance -- ROC Curves

 A good ROC curve – hugs top left corner: high TPR, low FPR

- A bad ROC curve – runs along diagonal: TPR equals the FPR

Probabilistic Interpretation of AUC

AUC Interpretation: AUC is the probability of correct ranking of a random "positive"-"negative" pair

- So, given a randomly chosen observation x_1 belonging to class 1, and a randomly chosen observation x belonging to class 0, the AUC is the probability that the evaluated classification algorithm will assign a higher score to x than to x_2, i.e., the conditional probability of $p(y = 1|x_1) > p(y = 1|x_2)$

AUC Computation: Among all "positive"-"negative" pairs in the dataset compute the proportion of those which are ranked correctly by the evaluated classification algorithm

$$\hat{AUC} = \frac{1}{P \times N} \sum_{i=1}^{P} \sum_{j=1}^{N} [p(y = 1|x_i) > p(y = 1|x_j)]$$