Note Set 2 – Machine Learning

2.1 - Introduction

In this note set, we introduce a number of methods for "Machine Learning", with a particular emphasis on text analysis. Machine learning is just a substitute term for statistics that a computer scientist would use. However, the types of statistical methods and the goal of applying these methods can differ substantially from conventional statistics. Conventional statistics often focuses on description, theory testing, or causal inference. Machine learning places a greater emphasis on prediction. Conventional statistics relies more heavily on parametric models with a small number of parameters. Machine learning places a greater emphasis on non-parametric methods and methods with a very large number of parameters. Finally, machine learning places a greater emphasis on clustering and multidimensional scaling than conventional statistics.

The main algorithms in machine learning are generally divided into supervised learning and unsupervised learning, which are basically synonymous with regression and clustering/scaling. We treat each of these separately.

2.2 - Unsupervised Learning

2.2.1 - The Dirichlet Distribution

Recall that the beta distribution has the parameters α and β and characterizes a random variable X that ranges between 0 and 1. The density of a beta random variable is given by,

$$f_X(x;\alpha,\beta) = \frac{x^{\alpha-1}(1-x)^{\beta-1}}{B(\alpha,\beta)}$$

$$B(\alpha,\beta) = \frac{\Gamma(\alpha)\Gamma(\beta)}{\Gamma(\alpha+\beta)}$$

The Dirichlet distribution is the multivariate generalization. Specifically, let X have support on the D-dimensional simplex. Let α be a vector strictly positive parameters. The Dirichlet distribution is given by,

$$f_X(x;\alpha) = \frac{1}{B(\alpha)}\prod_{d=1}^{D} x_d^{\alpha_d-1}$$

$$B(\alpha) = \frac{\prod_{d=1}^{D}\Gamma(\alpha_d)}{\sum_{d=1}^{D}\Gamma(\alpha_d)}$$

$$E[X_d] = \frac{\alpha_d}{\sum_{d'=1}^{D}\alpha_{d'}}$$

If you made all the alphas uniformly bigger, you would make the variance smaller

2.2.2 - The LDA Model

We assume that there are K topics, with I words in the vocabulary. There are D documents, with N_d in document d. We let $w_{dn} \in \{1,,...,I\}$ denote the n^{th} word in document d and $z_{d,n} \in \{1,K\}$ the topic that the n^{th} word in document d belong to. We let θ_{dk} be probability that a given word in document d belong to topic k, where $\theta_d \in \Delta^K$, so that $z_{d,n} \sim Categorical_K(\theta_d)$. We let $\varphi_{k,n}$ be the probability that word n belongs to topic k, so that $w_{d,n} \sim Categorical_I(\varphi_{z_{d,n}})$. We employ the priors $\varphi_k \sim Dirichlet_I(\beta)$ and $\theta_d \sim Dirichlet_K(\alpha)$.

In this model, θ is main parameter of interest since it tells us the proportion of each topic within a document. The parameter φ is of auxiliary interest because it tells allows us to interpret what the topics are. We must pre-select the number of topics K as well as the prior parameters α and β. In almost all cases, we would select all elements of α and all elements of β to be equal. In that case, a high value of α would mean that most documents contain most topics and a low value that most documents contain few topics. A high value of β means most topics contain similar words and a low values mean that most topics contain different words.

2.3 - Supervised Learning

2.3.1 - The Classification Problem

Suppose that we observe a sample $(y_1, X_1),(y_2, X_2),...,(y_D, X_D)$ where $y_d \in \{1,...,K\}$. This is called a classification problem. If $K = 2$, we would call this a binary classification

problem. If $K > 2$, we would call this a multi-class classification problem. Through this section, we will develop the algorithms with particular emphasis on the text classification problem.

2.3.2 - Naive Bayes

Suppose that for each document, indexed by d, we observe the vector w_d, with components $w_{dn} \in \{1,...,I\}$, where I is the number of terms in the vocabulary, where n indexes words in the document, and where N_d is the number of words in the document. We can write,

$$\Pr(y_d = k \mid w_d) = \frac{\Pr(y_d = k, w_d)}{\Pr(w_d)} = \frac{\Pr(w_d \mid y_d = k)\Pr(y_d = k)}{\Pr(w_d)} \propto \Pr(w_d \mid y_d = k)\Pr(y_d = k)$$

Notice that,

$$\Pr(w_d \mid y_d = k) = \Pr(w_{d1} \mid y_d = k, w_{d2},..., w_{d,N_d})\Pr(w_{d2},..., w_{d,N_d} \mid y_d = k)$$

$$= ... = \Pr(w_{d1} \mid y_d = k, w_{d2},..., w_{dN_d})\Pr(w_{d2} \mid y_d = k, w_{d3},..., w_{dN_d})...\Pr(w_{dN_d} \mid y_d = k)$$

We will make a conditional independence assumption to obtain,

$$\Pr(w_d \mid y_d = k) = \prod_{n=1}^{N_d} \Pr(w_{dn} \mid y_d = k)$$

which implies that,

$$\Pr(y_d = k \mid x_d) \propto \left(\prod_{n=1}^{N_d} \Pr(w_{dn} \mid y_d = k) \right) \Pr(y_d = k)$$

Define $N = \sum_{d=1}^{D} N_d$, $N_k = \sum_{d=1}^{D}\sum_{n=1}^{N_d} 1\{y_d = k\}$ and $N_{ki} = \sum_{d=1}^{D}\sum_{n=1}^{N_d} 1\{y_d = k, w_{dn} = i\}$. The maximum likelihood estimates for these are given by $\Pr(w_{dn} = i \mid y_d = k) \approx \frac{N_{ki}}{N_k}$ and $\Pr(y_d = k) \approx \frac{N_k}{N}$, or

alternatively, with smoothing parameter λ, we obtain, $\Pr(w_{dn} = i \mid y_d = k) \approx \frac{N_{ki}+\lambda}{N_k+I\lambda}$ and $\Pr(y_d = k) \approx \frac{N_k+\lambda}{N+K\lambda}$. This allows us to estimate,

$$\Pr(y_d = k \mid w_d) \propto \left(\prod_{n=1}^{N_d} \prod_{i=1}^{I} \frac{N_{ki}+\lambda}{N_k+I\lambda}^{1\{w_{dn}=i\}} \right) \frac{N_k}{N}$$

or,

$$\log \Pr(y_d = k \mid w_d) \propto \sum_{n=1}^{N_d} \sum_{i=1}^{I} 1\{w_{dn} = i\} \log \frac{N_{ki}+\lambda}{N_k+I\lambda} + \log\left(\frac{N_k}{N}\right) = \sum_{i=1}^{I} \log \frac{N_{ki}+\lambda}{N_k+I\lambda} \sum_{n=1}^{N_d} 1\{w_{dn} = i\} + \log\left(\frac{N_k}{N}\right)$$

More conveniently, define $X_{di} = \sum_{n=1}^{N_d} 1\{w_{dn} = i\}$, which is the number of times word i appears in document d. This would often be termed the document feature matrix. We have,

$$\log \Pr(y_d = k \mid w_d) \propto \sum_{i=1}^{I} X_{di} \log \frac{N_{ki}+\lambda}{N_k+I\lambda} + \log\left(\frac{N_k}{N}\right)$$

To classify a new document with features w_d, we would use,

$$\hat{y}_d = \arg\max_k \log \Pr(y_d = k \mid w_d) = \arg\max_k \sum_{i=1}^{I} X_{di} \log \frac{N_{ki}+\lambda}{N_k+I\lambda} + \log\left(\frac{N_k}{N}\right)$$

We can also calculate the predicted probability that $y_d = k$ using the fact that $\Pr(y_d = k \mid w_d) = Ce^{\sum_{i=1}^{I} X_{di} \log \frac{N_{ki}+\lambda}{N_k+I\lambda} + \log\left(\frac{N_k}{N}\right)}$ for some unknown constant C. We can calculate,

$$\Pr(y_d = k \mid w_d) = \frac{e^{\sum_{i=1}^{I} X_{di} \log \frac{N_{ki}+\lambda}{N_k+I\lambda} + \log\left(\frac{N_k}{N}\right)}}{\sum_{l=1}^{K} e^{\sum_{i=1}^{I} X_{di} \log \frac{N_{li}+\lambda}{N_l+I\lambda} + \log\left(\frac{N_l}{N}\right)}} = \frac{1}{1 + \sum_{l \neq k} e^{\sum_{i=1}^{I} X_{di} \log \frac{N_{li}+\lambda}{N_l+I\lambda} + \log\left(\frac{N_l}{N}\right)}}$$

A common variety is to consider the feature matrix $X_{di} = 1\left\{\sum_{n=1}^{N_d} 1\{w_{dn} = i\} > 0\right\}$ which counts if the document contains the feature. This is called the binary Naive Bayes, whereas the method we developed above is called the multinomial Naive Bayes.

2.3.3 - Logistic Regression

The logistic regression model fits the model,

$$\Pr(y_d = k) = \frac{e^{\alpha_k + \beta_k' X_d}}{\sum_{l=1}^{K} e^{\alpha_l + \beta_l' X_d}}$$

where we constrain one of (α_k, β_k) to be zero. We can write the likelihood as,

$$l(y, X; \alpha, \beta) = \sum_{n=1}^{N} \sum_{k=1}^{K} 1\{y_d = k\} \log \frac{e^{\alpha_k + \beta_k' X_d}}{\sum_{l=1}^{K} e^{\alpha_l + \beta_l' X_d}}$$

Particularly in applications to text, this approach will not work since $\dim((\alpha, \beta))$ could easily bigger than the sample size. To solve this problem, regularization is used. In particular, we consider either an L2 regularization (as in ridge regression) or an L1 regularization (as in the LASSO). For ridge regression, we have,

$$(\hat{\alpha}, \hat{\beta}) = \arg\max_{(\alpha,\beta)} Q(\alpha, \beta; y, X) = l(y, X; \alpha, \beta) - \lambda \beta' \beta$$

for a fixed value of $\lambda > 0$.

Suppose that we assume a prior of $\alpha \sim Unif(-\infty, \infty)$ and $\beta \sim N(0, \frac{1}{2}\lambda^{-1} I)$. We have that,

$$\Pr(\alpha, \beta \mid y, x) \propto L(y, x; \alpha, \beta)\pi(\alpha)\pi(\beta) \propto L(y, x; \alpha, \beta) e^{-\lambda \beta' \beta}$$

$$\log \Pr(\alpha, \beta \mid y, x) \propto l(y, x; \alpha, \beta) - \lambda \beta' \beta$$

This implies that we can think ridge regression as maximizing a Bayesian posterior where the prior is a diagonal normal and the penalty parameters λ is inversely related to the variance of the prior (a bigger penalty mean a more certain prior, so we penalize deviations more).

The other type of regularization leads to the objective function,

$$Q(\alpha,\beta) = l(\alpha,\beta) + \lambda \| \beta \|_1 = l(\alpha,\beta) + \lambda \sum_{j=1}^{J} | \beta_j |$$

The Laplace (or double-exponential) distribution is given by,

$$f_X(x;\mu,b) = \tfrac{1}{2b} e^{-\frac{1}{b}|x-\mu|}$$

Consider the prior which assume that β_j are independent $\beta_j \sim DE(0,\lambda^{-1})$. We have,

$$\Pr(\alpha,\beta \mid y,x) \propto L(y,x;\alpha,\beta)\pi(\alpha)\pi(\beta) \propto L(y,x;\alpha,\beta) e^{-\lambda \sum_{j=1}^{J} |\beta_j|}$$

$$\log \Pr(\alpha,\beta \mid y,x) \propto l(y,x;\alpha,\beta) - \lambda \| \beta \|_1$$

The interesting thing about this estimator is that $\| \beta \|_1$ is not differentiable in β_j when $\beta_j = 0$. For this reason, it is possible that optimization of the objective function may lead many of β_j will be set to zero. This is arguably a desirable property.

Combining the ridge and LASSO constraints leads to *elastic nets*. This method also easily extends to the multiclass classification problem (using a multinomial logit model), an ordered classification problem (using an ordered logit or ordered probit model), or a continuous classification problem (using an Ordinary Least Squares model).

2.3.4 - Support Vector Machines

Support vector machines depart quite a bit from conventional statistics. The vectors X_d denote points in multidimensional space and y_d denotes a predicted outcome. We would like to select a line that divides the points in X_d such that all points on one side have $y_d = 1$ and all points on the other side have $y_d = -1$. Suppose that this is possible. In this case, there will be multiple such lines. To choose a unique line for performing the classification, we create two lines that are parallel to each other. We denote these lines by the equations $w'X + b = 1$ and $w'X + b = -1$. We select w and b such that all points on one side of $w'X + b = 1$ have $y_d = 1$ and all points on the other side of $w'X + b = -1$ have $y_d = -1$. We then make these lines as far apart as possible. We can show that the distance between these two lines is $\frac{2}{\| w \|_2} = \frac{2}{\sqrt{w'w}}$ which means that to maximize this distance, we can minimize $w'w$.

We can set this up as an optimization problem,

$$\min_{(w,b):y_d(w'X_d+b)\geq 1 \forall d} w'w$$

Our classifier then becomes $y_d = sign(w'X_d + b)$. Note that if we push the hyperplanes as far apart as possible, they will touch up against some of the X_d s. This leads to the name of the classifier, since X_d are support vectors. This is called a hard margin SVM.

This approach by itself is not very useful. Instead we consider a soft margin SVM. We consider a linear classifier $w'X_d + b$. As mistake will occur when $y_d(w'X_d + b) < 0$. The more negative the number, the bigger the mistake since this means that the point was far away from the line. Let $1 - \xi_d$ quantify the size of the mistake. We can require that $y_d(w'X_d + b) \geq 1 - \xi_d$ which limits the size of the mistake. We can then use the objective function $\frac{1}{2}w'w + C\sum_{d=1}^{D}\xi_d$.

Here, we penalize both planes that are too close together and lines that allow bigger mistakes. We consider the constraints $y_d(w'X_d + b) \geq 1 - \xi_d$ and $\xi_d \geq 0$ (where the second constraint disallows credit for negative mistakes).

We have, $\min_{(w,b,\xi): y_d(w'X_d+b) \geq 1-\xi_d, \xi_d \geq 0 \forall d} \frac{1}{2} w'w + C\sum_{d=1}^{D} \xi_d$. This can again be solved using quadratic programming.

Next, we can consider nonlinear transformations to obtain the features. We can consider $\varphi(X_d)$. For example, $\varphi(X_d) = (1, X_{d1}, X_{d2}, X_{d1}^2, X_{d2}^2, X_{d1}X_{d2})$. We can apply the same setup to obtain,

$$\min_{(w,b,\xi): y_d(w'\varphi(X_d)+b) \geq 1-\xi_d, \xi_d \geq 0 \forall d} w'w + C\sum_{d=1}^{D} \xi_d$$

There is a useful trick we can apply however. We can write the Lagrangian as,

$$L(w,b,\xi,\beta,\delta) = \frac{1}{2} w'w + C\sum_{d=1}^{D} \xi_d - \sum_{d=1}^{D} \alpha_d (y_d(w'\varphi(X_d) + b) - 1 + \xi_d) - \sum_{d=1}^{D} \delta_d \xi_d$$

Taking first-order conditions with respect to w, b, and ξ_d yields,

$$w = \sum_{d=1}^{D} \alpha_d y_d \varphi(X_d)$$

$$\sum_{d=1}^{D} \alpha_d y_d = 0$$

$$C = \alpha_d + \delta_d$$

$$\xi_d = 1 - y_d(w'\varphi(X_d) + b) \text{ or } \xi_d = 0.$$

Plugging this in yields,

$$\max_{(\alpha,C)} \frac{1}{2} \sum_{d=1}^{D} \sum_{d'=1}^{D} \alpha_d \alpha_{d'} y_d y_{d'} \varphi(x_d)'\varphi(x_{d'}) + C\sum_{d=1}^{D} \alpha_d$$

$$\text{s.t. } \sum_{d=1}^{D} \alpha_d y_d = 0 \text{ and } 0 \le \alpha_d \le C .$$

which is the dual of the original problem.

The last part of trick is that $\varphi(x_d)'\varphi(x_{d'})$ can be easily computed. Specifically, for the linear kernel, $\varphi(x_d)'\varphi(x_{d'}) = x_d' x_{d'}$. For a polynomial kernel, $\varphi(x_d)'\varphi(x_{d'}) = (1 + x_d' x_{d'})^q$ where $\varphi(x_d)$ include all polynomials up to degree q. For the Gaussian (or radial) kernel, $\varphi(x_d)'\varphi(x_{d'}) = e^{-(x_d - x_{d'})'(x_d - x_{d'})/2\sigma^2}$, which corresponds to an infinite dimension feature space.

There exist extensions to SVMs that deal with the multiclass classification problem (by reducing the multiclass problem to multiple binary classification problems) as well as the case of continuous classification (using Support Vector Regression).

2.3.5 - k-Nearest Neighbors

With k-Nearest Neighbors, our predicted values is based on the k closest point in the training set. Specifically, we have,

$$\hat{y} = \tfrac{1}{k} \sum_{d \in K_k(X)} y_d$$

where $K_k(X)$ are the k closest values of $\{X_1, X_2, ..., X_T\}$ to X.

A value of k can be chosen on a training set, using K-fold cross validation, or even leave one out cross validation.