10-701/15-781, Machine Learning: Homework 1

Eric Xing, Tom Mitchell, Aarti Singh
Carnegie Mellon University
Updated on January 12, 2010

- The assignment is due at 10:30am (beginning of class) on **Wed, Jan 20, 2010**.
- Separate you answers into three parts, one for each TA, and put them into 3 piles at the table in front of the class. Don't forget to put both your name and a TA's name on each part.
- If you have question about any part, please direct your question to the respective TA who design the part.

1 Naive Bayes Classifier [30 pt, Field Cady]

In classification problems, we often have several pieces of "evidence". The problem is that, even if each piece of evidence is highly informative, taken together they may be redundant.

Imagine a problem where we must classify data points as y_1 or y_2, based on evidence random variables $X_1, X_2, ..., X_d$. From Bayes rule, we know that $P(y_i|X_1, ..., X_d) \propto P(y_i)P(X_1, ..., X_d|y_i)$, so for classification we just need to compare $P(y_1)P(X_1, ..., X_d|y_1)$ and $P(y_2)P(X_1, ..., X_d|y_2)$.

But calculating $P(X_1, ..., X_d|y_i)$ requires knowing potentially complicated dependencis between the X_i. In a Naive Bayes classifier, we simplify the problem by assuming $P(X_1, ..., X_d|Y) = P(X_1|Y) \times ... \times P(X_d|Y)$; all the pieces of evidence are conditionally independent. And $P(X_i|Y)$ is easy to estimate; if we use the MLE it's just $\frac{\#(X_i \& Y)}{\#Y}$. This classifier is "naive" because somebody might implement it without realizing there could be complicated dependencies, but generally people are aware of its limitations and use it for its simplicity.

1.1 Why We Use Naive Bayes[15 pt]

A big reason we use Naive Bayes classifiers is that they require less training data than Full Bayes Classifiers. This problem should give you a "feel" for how great the disparity really is.

Imagine that each observation is an independent instance of the multi-variate random variable $\vec{X} = X_1, ..., X_d$, where the X_i are i.i.d and Bernoulli(.5). To train a Full Bayes classifier, we need to see every value of $\vec{X}$ "enough" times; training a Naive Bayes classifier only requires seeing both values of X_i "enough" times. We wonder how many observations are needed until, with probability $1 - \epsilon$, we have seen every variable we need to see at least once. To train the classifier well would require more than this, but for this problem we only require one observation.

1. We start with Full Bayes. Let $\vec{x}$ be a particular value of $\vec{X}$. Show that after N observations, the probability we have never seen $\vec{x}$ is $\leq e^{-N/2^d}$.
2. Show that if more than $N_{FB} = 2^d \ln\left(\frac{2^d}{\epsilon}\right)$ observations have been made, then the probability that *any* value of $\overrightarrow{X}$ has not been seen is $\leq \epsilon$.
3. Now on to Naive Bayes. Show that if N observations have been made, the probability that a given X^i has *not* been seen as both 0 and 1 is $\leq \frac{1}{2^{N-1}}$.
4. Show that if more than $N_{NB} = 1 + \log_2\left(\frac{\epsilon}{d}\right)$ observations have been made, then the probability that *any* X^i has not been observed in both states is $\leq \epsilon$.

5. Let $d = 2$ and $\epsilon = .1$. What are the values of N_{FB} and N_{NB}? What about $d = 5$? And $d = 10$?

1.2 How bad is Naive Bayes?[15 pt]

Clearly Naive Bayes makes what, in many cases, are overly strong assumptions. But even if those assumptions aren't true, is it possible that Naive Bayes is still pretty good? This problem uses a simple example to explore the limitations of Naive Bayes.

Let X_1 and X_2 be i.i.d. Bernoulli(.5) random variables, and let $Y \in \{1, 2\}$ be some deterministic function of the X^i ; hence Y can be represented by a 2x2 grid of 1s and 2s. Imagine that we have trained a Naive Bayes classifier perfectly, so that $P(Y|X_i)$ is known perfectly.

1. Find a function Y for which the Naive Bayes classifier has a 50% error rate. Given the value of Y, how are X_1 and X_2 correlated?

2. Show that for every function Y, the Naive Bayes classifier will perform no worse than theone above.

   Hint: there are many Y functions, but because of symmetries in the problem you only need to analyze a few of them.

2 Multiclass Classification[40pt, Ni Lao]

In this part, you are going to play with the 'The ORL Database of Faces'. Each image is 92 by

Figure 1: 6 sample images from two persons

112 pixels. If we treat the luminance of each pixel as a feature, each sample has $92 * 112 = 10304$ real value features, which can be written as a random vector X. We will treat each person as a class Y ($Y = 1...K, K = 10$). We use X_i to refer the i-th feature. Given a set of training data $D = \{(y^l, x^l)\}$, we will train different classification models to classify images to their person id's. To simplify notation, we will use $P(y|x)$ in place of $P(Y = y|X = x)$.

We will select our models by 10-fold cross validation: partition the data for each face into 10 mutually exclusive sets (folds). In our case, exactly one image for each fold. Then, for k=1...10, leave out the data from fold k for all faces, train on the rest, and test on the left out data. Average the results of these 10 tests to estimate the training accuracy of your classifier.

Beware that we are actually not evaluating the generalization errors of the classifier here. When evaluating generalization error, we would need an independent test set that is not at all touched during the whole developing and tuning process.

For your convenience, a piece of code "loadFaces.m" is provided to help loading images as feature vectors.

1. **KNN [10 pt]** Implement the KNN algorithm we learnt from the class. Use L2-norm as the distance metric. Show your evaluation result here, and compare different values of K.

2. **Conditional Gaussian Estimation [5 pt]** For a Gaussian model we have

$$P(y|x) = \frac{P(x|y)P(y)}{P(x)}$$

where

$$P(x|y) = \frac{1}{(2\pi)^{d/2}|\Sigma_y|^{1/2}} \exp\{-(x-\mu_y)'\Sigma^{-1}(x-\mu_y)/2\},$$

and $P(y) = \pi_y$. Please write down the MLE estimation of model parameters Σ_y, μ_y, and π_y. Here we do not assume that X_i are independent given Y

3. **Gaussian Naive Bayes Model [5 pt]** Gaussian NB is a form of Guassian model with assumption that X_i are independent given Y. Please implement the Gaussian NB model. Show your evaluation result here.

4. **Multinomial Logistic Regression [5 pt]** From the reading material (Tom's chapter draft) you will see a generalization of logistic regression, which allow Y to have more than two possible values. Write down the objective function, and the first order derivatives of the multinomial logistic regression model (which is a binary classifier). Here we consider a L2-norm regularized objective function (with a term $\lambda|\theta|_2$).

5. **Gradient Ascent [5 pt]** Implement the logistic regression model with gradient ascent. Show your evaluation result here. Use regularization parameter $\lambda = 0$. **Hint:** The gradient ascent method (also known as "steepest ascent") is a first-order optimization algorithm. It optimizes a function $f(x)$ by

$$x_{t+1} = x_t + \alpha_t f'(x_t),$$

where α_t is called the *step size*, which is often picked by *line search*. For example, we can initialize $\alpha_t = 1.0$. Then set $\alpha_t = \alpha_t/2$ while $f(x_t + \alpha_t f'(x_t)) < f(x_t)$. The iteration stops when the change of x or $f(x)$ is smaller than a threshold (the optimization is converged). **Hint:** if the training time of your model is too long, you can consider use just a subset of the features (e.g. in Matlab X=X(:,1:100:d))

6. **Overfitting and Regularization [5 pt]** Now we test how regularization can help prevent overfitting. During cross validation, let's use m images from each person for training, and the reset for testing. Report your cross-validated result with varying $m = 1...9$ and varying regularization parameter λ.

7. **[5 pt]** Compare the above methods by training/testing time, and accuracy. Which do you prefer?

3 Linear Regression[30pt, Amr]

In linear regression, we are given training data of the form, $\mathcal{D} = (\mathbf{X}, \mathrm{y}) = \{(\mathrm{x}_i, y_i)\}, i = 1, 2, ..., N$, where $\mathrm{x}_i \in \mathcal{R}^{1\times M}$, i.e. $\mathrm{x}_i = (x_{i,1}, \cdots, x_{i,M})^{\mathrm{T}}$, $y_i \in \mathcal{R}$, $\mathbf{X} \in \mathcal{R}^{N\times M}$, where row i of $\mathbf{X}$ is ${\mathrm{x}_\mathrm{i}}^{\mathrm{T}}$, and $\mathrm{y} = (y_1, \cdots, y_N)^{\mathrm{T}}$. Assuming a parametric model of the form: $y_i = {\mathrm{x}_\mathrm{i}}^{\mathrm{T}}\beta + \epsilon_i$, where ϵ_i are noise terms from a given distribution, linear regression seeks to find the parameter vector β that provides the best of *fit* of the above regression model. One criteria to measure fitness, is to find β

that minimizes a given loss function $\mathrm{J}(\beta)$. In class, we have shown that if we take the loss function to be the square-error, i.e.:

$$\mathrm{J}_1(\beta) = \sum_i (y_i - \mathrm{x_i}^{\mathrm{T}}\beta)^2 = (\mathbf{X}\beta - \mathrm{y})^{\mathrm{T}}(\mathbf{X}\beta - \mathrm{y})$$

Then

$$\beta^* = (\mathbf{X}^{\mathrm{T}}\mathbf{X})^{-1}\mathbf{X}^{\mathrm{T}}\mathrm{y} \tag{1}$$

Moreover, we have also shown that if we assume that $\epsilon_1, ..., \epsilon_N$ are IID and sampled from the same zero mean Gaussian that is, $\epsilon_i \sim \mathcal{N}(0, \sigma^2)$, then the least square estimate is also the MLE estimate for $p(\mathrm{y}|\mathbf{X}; \beta)$.

In this problem we will explore several extensions to this basic regression model. The following facts might be useful for some parts of problem 2.3:

- The column (row) rank of a matrix A is the maximal number of linearly independent columns (rows) of A.
- If A is $m \times n$ then rank(A)$\leq$ min(n,m)
- An $n \times n$ matrix A is invertible iff it is full-rank, i.e rank(A)=n.
- if A = C^{T}C, then rank(A)=rank(C).
- if A = B + C, then rank(A) $\leq$ rank(B)+ rank(C)

Note: for this problem, you really need to show your work in clear steps to get **full credit**.

3.1 Weighted Least-square

Assume that $\epsilon_1, ..., \epsilon_N$ are independent but each $\epsilon_i \sim \mathcal{N}(0, \sigma_i^2)$.

(a) [**2 points**] Write down the formula for calculating the MLE of β.

(b) [**4 points**]Calculate the MLE of β. [Show your work]

(c) [**2 points**] Show that the MLE you just calculated is the minimizer of the weighted least square loss function $\mathrm{J}_2(\beta) = \sum_i a_i(y_i - \mathrm{x_i}^{\mathrm{T}}\beta)^2$. Express each a_i in terms of the variance of each example.

(d) [**2 points**] Explain why this weighted least-square estimator is preferred to the non-weighted version. (hint: Consider the case when σ_i^2 is large and when it is small).

3.2 Laplace noise-model

Assume that $\epsilon_1, ..., \epsilon_N$ are independent and identically distributed according to a Laplace distribution. That is each $\epsilon_i \sim \mathrm{Laplace}(0, b) = \frac{1}{2b}\exp(-\frac{|\epsilon_i|}{b})$.

(a) [**3 points**]Provide the loss function $\mathrm{J}_3(\beta)$ whose minimization is equivalent to finding the MLE of β under the above noise model.

(b) [**2 points**] What is the advantage of this model compared to the standard Gaussian assumption? (hint: think about outliers)

3.3 Regularization: Ridge Regression

For this part assume that the noise terms are IID distributed according to $\mathcal{N}(0, \sigma^2)$. Also assume that the number of features M is much larger than the number of training instances N (i.e., $M \gg N$).

(a) [**1 point**] Explain why in this situation, we can NOT compute β according to (1).

(b) [**5 points**] Instead of minimizing $\mathrm{J}_1(\beta)$, we minimize the following loss function:

$$\mathrm{J}_R(\beta) = \sum_i (y_i - \mathrm{x_i}^{\mathrm{T}}\beta)^2 + \lambda \sum_{j=1}^{M} \beta_j^2 = (\mathbf{X}\beta - \mathrm{y})^{\mathrm{T}}(\mathbf{X}\beta - \mathrm{y}) + \lambda \parallel \beta \parallel^2 \tag{2}$$

Derive the value of β^* that minimizes (2) in closed form and show that it is given by $\beta^* = (\mathbf{X}^{\mathrm{T}}\mathbf{X} + \lambda I)^{-1}\mathbf{X}^{\mathrm{T}}\mathrm{y}$. [please, show you work **in details** to get full credit]

(c) [**2 points**] Now revisit your answer to (a) and explain the effect of adding this extra term to the loss function.

(d) We have shown in class that the minimizer of $\mathrm{J}_1(\beta)$ is the same as the **MLE** estimate under the IID Gaussian noise assumption. An alternative view is to consider β as a random variable and specify a **prior distribution** $p(\beta)$ on β that expresses our prior belief about the parameters. Then we estimate β using the **MAP (maximum a posteriori)** estimate as:

$$\beta_{\mathrm{MAP}} = \arg\max_{\beta} \prod_{i=1}^{N} p(y_i|\mathrm{x}_i;\beta)\ p(\beta) \tag{3}$$

Assume that $\beta \sim \mathcal{N}(0, \tau^2 I)$:

(i) [**4 points**] Show that maximizing (3) results in the same value of β obtained by minimizing (2) for some value of λ. In other words, show that you can express (3) as (2).

(ii) [**1 point**] Express λ as a function of σ and τ.

(iii) [**2 points**] If we set $\tau = \infty$, what prior belief do we have over β, and how does this affect the loss function?