10-701/15-781, Machine Learning: Homework 4

Eric Xing, Tom Mitchell, Aarti Singh
Carnegie Mellon University
Updated on March 24, 2010

- The assignment is due at 10:30am (beginning of class) on **Wen, April 7, 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.
- Submit your code to HW4 in the blackboard *http : //www.cmu.edu/blackboard*

1 Bayesian Networks [25 pts, Field Cady]

Bayesian networks are a convenient way to represent disitributions where the random variables are only partly dependent on each other.

1.1 Part a : Creating a Network [8 pts]

One useful aspect of Bayesian networks is that they can make use of expert knowledge about a domain. Let's take a (highly) simplified view of land-based ecosystems, and try to characterize all ecosystems in the world. Two of the most important aspects of an ecosystem are temperature(T) and precipition(P); an area can be either hot or cold, and either wet, moist or dry. The foundation of an ecosystem is its vegetation(V), and it is either sparse or dense, and either trees or shrubs/grasses (so 4 possibilities). More water and higher temperatures both make trees more likely and vegetation more dense. The presence grazing animals (G, boolean for whether or not there are grazing animals), like bison, depends on there being a lot of grass, but they don't really need rain or moderate temperatures. Frogs (F, boolean for whether there are frogs), on the other hand, need lots of water and prefer to live near trees. Construct a Bayesian network to describe the relationship between these random variables.

1.2 Part b : Memory Efficiency [5 pts]

Perhaps the most important thing about Bayesian networks is that they require vastly less memory to store; in general it takes exponential space to store the dependencies among random variables, but a Bayes net can often reduce the complexity to linear. Calculate

- How many real numbers are required to store the complete probability distribution for the random variables in part a, not making any conditional independence assumptions.
- The number of floats required to store the distribution using our conditional independence assumptions.

1.3 Part c : Inference [8 pts]

Bayes nets are fabulous for storing large distributions, but actually using them can be tricky. To get a feel for this, show graphically an elimination sequence for calculating $P(T|F)$.

1.4 Part d : Generalizing Complexity [4 pts]

Imagine a Bayes net with binary random variables X_1, X_2,... , X_n. Let X_i be conditionally dependent on all X_j with $j < i$. Find a simple formula, as a function of n, for the number of real numbers needed to store this distribution.

This distribution is problematic because X_n and its neighbors have so many conditional dependencies. What if each X_i was conditionally dependent only on the one before it? What would the formula be then?

2 Learning Theory [25pt, Ni Lao]

Sample complexity and model complexity are two important concepts in machine learning. We will explore them in this question, and you will also practice you skill in estimating VC dimensions.

2.1 Gaussian Bayes Model [15 pt]

Let's consider several Gaussian Bayes classification Models for the classification problem $P(Y|X)$. All models here assume there are two classes ($Y \in \{0, 1\}$), that X is a vector of real-valued features, and $P(X|Y = 1)$ and $P(X|Y = 0)$ are modeled by two different Gaussian distributions. Please complete the following table by filling in the number of parameters and VC dimension of different Gaussian Bayes learning models under different settings. Each column describes a different Bayesian classification model in terms of (1) the number of features in X, (2) whether the two Gaussians for $P(X|Y = 1)$ and $P(X|Y = 0)$ share the same co-variance matrix, and (3) whether the model makes the naive Bayes assumption of conditional independence among features. (note the first two columns have '-' for Naive Bayes because these columns assume X has just one feature).

ID	**a**	**b**	**c**	**d**	**e**	**f**
No. features	1	1	2	2	2	2
Shared Covariance Matrix?	Y	N	Y	Y	N	N
Naive Bayes?	-	-	Y	N	Y	N
No. parameters						
VC dimension						

Policy: number of parameters is 0.5pt each. VC dimension is 2pt each, and you get $\max(0, 2 - d_{best} + d_{your})$pt, where d_{best} is the best bound I know, and d_{your} is your answer. You need to convince me in order to get credit for the VC dimension, but you need not give a formal proof.

Hint: think about what kind of decision boundary we get in each of the models.

2.2 VC dimension and Effective Number of Hypothesis [3 pt]

For models where the set of instances X involve continuous variables, the hypothesis space H defined over X may contain an infinite number of hypothesis. In order to quantify the richness of our hypothesis space, we now define the Effective Number of Hypotheses of a model with respect to a set of unlabeled data points $D = (x_1, \cdots, x_m)$ as $N_{eff}(H, D)$, which is the number of different ways samples in D can be divided into positive and negative ones by hypotheses in H.

What is the relationship between $N_{eff}(H, D)$ and m, the number of samples in D?

Hint: discuss separately for $m \leq VC(H)$ and $m > VC(H)$

2.3 [2 pt]

Assume no noise in the labels. At least how many labeled samples do we need in order to identify one out of the $N_{eff}(H, \mathcal{D})$ hypothesis groups?

Hint: think about entropy (or information)

2.4 Linear Regression Model [3 pt]

Now let's explore the relations between sample complexity, model complexity, and number of parameters on a model we all know so well about.

In homework 3 we derived the expected error of linear (and ridge) regression. Basically, we are given training data of the form, $\mathcal{D} = (\mathbf{X}, \mathrm{y}) = \{(\mathrm{x}_i, y_i)\}, i = 1, 2, ..., n$, where $\mathrm{x}_i \in \mathcal{R}^{1\times p}$, i.e. $\mathrm{x}_i = (x_{i,1}, \cdots, x_{i,p})^{\mathrm{T}}$, $y_i \in \mathcal{R}$, $\mathbf{X} \in \mathcal{R}^{n\times p}$, where n is number of samples, p is number of features. Each row i of $\mathbf{X}$ is $\mathrm{x_i}^{\mathrm{T}}$, and $\mathrm{y} = (y_1, \cdots, y_n)^{\mathrm{T}}$. We assumed that $p < n$, and $\mathbf{X}^T\mathbf{X}$ is invertible. Also assume that our data is generated from a true model of the form: $y_i = \mathrm{x_i}^{\mathrm{T}}\beta + \epsilon_i$ (or in matrix form $\mathrm{y} = \mathbf{X}\beta + \epsilon$), where $\epsilon_1, ..., \epsilon_n$ are IID and sampled from a Gaussian with 0 mean and constant standard deviation, that is $\epsilon_i \sim \mathcal{N}(0, \sigma^2)$ (or $\epsilon \sim \mathcal{N}(0, \sigma^2 I)$).

We also assumed that the true parameter β itself is a random variable $\sim \mathcal{N}(0, \alpha^2 I)$. Apart from our training data $\mathcal{D} = (\mathbf{X}, \mathrm{y})$, we generate a set of testing data $\mathcal{D}^* = (\mathbf{X}, \mathrm{y}^*)$. It has exactly the same x values $\mathbf{X}$ as training data, but the y values are regenerated independently. Again we can decompose them as $\mathrm{y}^* = \mathbf{X}\beta + \epsilon^*$. Relations among these quantities can be summarized by the figure below.

$$\begin{tikzcd}
\varepsilon \arrow[r] & y \arrow[r] & \hat{\beta} \arrow[dd] & \lambda \arrow[l] \\
\beta \arrow[ur] \arrow[dr] & X \arrow[u] \arrow[d] \arrow[ur] \arrow[dr] & & \\
\varepsilon^* \arrow[r] & y^* & \hat{y}^* &
\end{tikzcd}$$

Finally, we showed that the risk of ridge regression can be expressed in terms of the regularization parameter λ as

$$R(\lambda) = E[e(\lambda)^T e(\lambda)] \tag{1}$$

$$= \sum_{i=1..p} \left[\left(\frac{\lambda d_i}{d_i^2 + \lambda} \right)^2 \alpha^2 + \left(\frac{d_i^2}{d_i^2 + \lambda} \right)^2 \sigma^2 \right] + \sum_{i=1..n} \sigma^2, \tag{2}$$

Here we decompose X as $X = UDV^T$ by using SVD, where D is a $p \times p$ diagonal matrix, V is a $p \times p$ unitary matrix, U is a $n \times p$ matrix, which is the first p columns of a unitary matrix. We define $d_i = D_{i,i}$.

Please express the average risk $R(\lambda)/n$ for linear regression (not ridge regression) as a function of p and n. It has two terms, one corresponds to the irreducible error, one corresponds to the variance of the model.

Hint: the above solution to HW3 Q1 is slightly revised (see the online documents). Basically U should be $n \times p$ instead of $n \times n$

2.5 [2 pt]

How many training examples n suffice to assure that the error term which corresponds to the model variance will be no larger than the irreducible error? What is its relationship to the VC dimension of linear regression model?

3 HMM[55pt, Amr]

Please upload your code to the blackboard and hand a print-out of your writing (but not the code) including all plots.

As we discussed in the class and recitation, an HMM is used to model a sequence of observed variables as generated from a sequence of hidden states. For instance, the observed sequence might be a sentence and the hidden sequence might be the part of speech tags for each word in the sentence. However, HMM can be also used in modeling continuous observations. For instance, in speech , the observed sequence is the speech signal, and the hidden sequence represents the words. The data for this problem is available over the website `data.mat`.

In this problem you will implement an HMM with continuous observations. You will implement solutions to the three basic problems in HMM: evaluation, decoding and learning.

Recall that an HMM defines a joint probability distribution over the observed and hidden sequences $p(x_1, x_2, \cdots, x_T, y_1, y_2, \cdots, y_T)$, where x and y represent the observed and hidden sequences respectively. Each hidden state $y_i \in \{1, \cdots, K\}$, and each observation $x_i \in \mathcal{R}$. The parameters of this continuous-observation HMM are: the distribution over the initial state π, the transition probability $a_{ij} = P(Y_t = j|Y_{t-1} = i)$, and the emission *density* $p(x_t|y_t = i) = \mathcal{N}(\mu_i, \sigma_i^2)$.

Our motivating application is detecting copy number variation in the genome. The human genome is comprised of 6 billion chemical bases (or nucleotides) of DNA packaged into two sets of 23 chromosomes, one set inherited from each parent. The DNA encodes 30,000 genes. Usually genes are present in two copies (copy number) in a genome. A variation in this copy number can be linked to diseases. aCGH (Array comparative genomic hybridization) is an experimental technique used to measure copy number in a test sequence (like a genome sequence from a patient with cancer) relative to a normal sequence. The output of this procedure is represented in terms of a log2 ratio between the copy number of the test sequence and the reference (normal) sequence. For simplicity, we assume that there are three state of the copy number: 0 (normal), log2(1/2) (missing copy) , and log2(3/2) (gain). However, the aCGH procedure is noisy and thus you might get a noisy measurement as shown in Fig. 1. In this figure, the Y-axis gives the log2 ratio and the X-axis represents location over the genome. Our biological knowledge tells us that the copy number of nearby locations should be the same which suggests that an HMM model might be a good option in recovering the correct copy number from this noisy measurements (Fig. 1 has 11 segments).

Unless otherwise stated, during this problem we take $K = 3$. Moreover, in this problem we will deal only with one-sequence for simplicity, however, in practice, your EM-algorithm can be easily changed to work with multiple sequences.

1. [**2 points**] Write down the forward and backward recurrences for this continuous-observation HMM.

2. Implement the forward-backward algorithm. (you might want to refer to the book or recitation slides for how to avoid underflow using re-scaling)

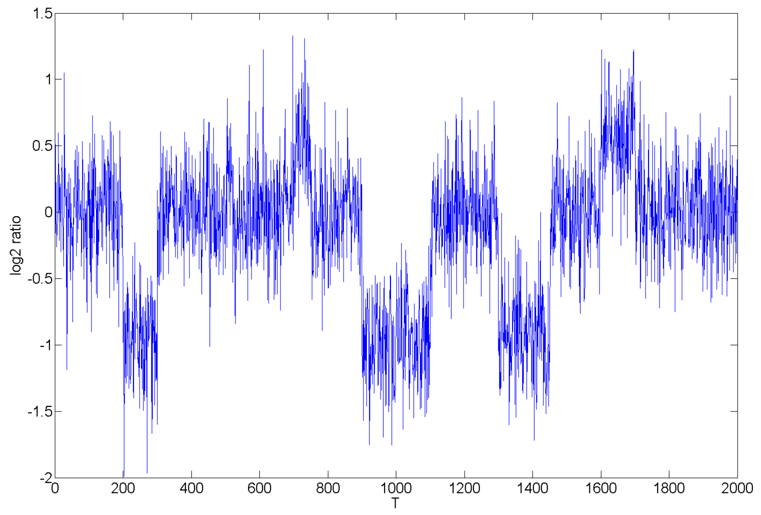

Figure 1: Copy number variation sequence. X-axis represents location over the genome. There are 11 segments

3. To estimate the parameters of an HMM we use the EM algorithm. We start the EM algorithm by randomly initializing the model parameters. In the E-step we run the forward-backward algorithm over the sequence(s) using the current estimate of the model parameters, and in the M-step we re-estimate the model parameters using the expected sufficient statistics from the E-step. We repeat this two steps until the log-likelihood of the observed sequence asymptotes (i.e. converges).

   i. [**2 points**] Using the α and β probabilities, write down the MLE estimate of π_i and a_{ij}.

   ii. [**5 points**]Using the α and β probabilities, in addition to the observed sequences, write down the MLE estimate of μ_k and σ_k^2. (Hint: this is somehow similar in spirit to the Gaussian Naive Bayes, i.e, it can be represented in terms of $P(Y_t = i|x_1, \cdots, x_T)$)

   iii. Implement the EM algorithm. Stop iterating when the relative change in log-likelihood (LL) falls below 1e-3, where the relative change in LL $= |\frac{\text{LL at iteration r} - \text{LL at iteration (r-1)}}{\text{LL at iteration (r-1)}}|$

   iv. [**25 points**] Run your EM algorithm over the given sequence. Initialize π and the transition matrix randomly (make sure they are correct probability distributions), and initialize μ_i by drawing it randomly from $\mathcal{N}(0,1)$ and σ_i^2 by drawing it uniformly from [.01,1]. Repeat this experiment 5 times each of which with different random initialization. **Record** the final LL of each run and **write down** the EM-parameters for the best and worst LL. **Also draw** the trace of the LL over iterations for the runs with worst and best final LL. **What** do you observe and why? (**Note**: if all your runs give the same LL, try to initialize μ by drawing it from $\mathcal{N}(0,2)$, in my implementation, I was able to get different LL either way).

4. Decoding.

   i. Implement the Viterbi decoding algorithm. (perform all operations in log-space to avoid under flow).

   ii. [**15 points**] Using the estimated parameters by the EM-algorithm, run the Viterbi algorithm to get the most-probably hidden state sequence. Choose two runs from your

5-runs in (3.iv): one that gets the correct segmentation and one that doesn't. Overlay the segmentation of each run over the data in Fig. 1. To do that, for each point, draw the mean of its corresponding hidden state as recovered by Viterbi-decoding. (I am looking here for two figures, one for each run, each of which has the data and the segmentation, please draw the segmentation with a thicker marker, i.e 'linewidth'=5 in matlab).

5. [**1 points**] Lets assume you don't know the correct segmentation, did the lowest and highest LL in 3.iv correspond to a correct and incorrect segmentation?

6. [**3 points**] Would you expect that a GMM with three mixtures would recover the correct segmentation, and why? Either argue that GMM will result in the exact segmentation or point to areas in Fig. 1 in which GMM will fail.

7. [**2 points**] If we don't know the correct number of states K, suggest a scheme to select K. You shouldn't assume that you have a reference to the correct segmentation. Moreover, assume we only have one sequence.