CS 475 Machine Learning: Homework 6
Structured Prediction
Due: Wednesday December 7, 2016, 11:59pm

Jin Yong Shin
JHED : Jshin44

1 Analytical (25 points)

1. (12 points) One of the uses of PCA is to create better data representations for classification. Consider two different classes in a binary classification task. The first class has points generated using a Gaussian with $\mu_1 = \{1, -1\}$ and covariance

$$\mathbf{\Sigma}_1 = \begin{pmatrix} 1 & 0 \\ 0 & .001 \end{pmatrix}$$

The second class has points generated using a Gaussian with $\mu_2 = \{1, 1\}$ and covariance

$$\mathbf{\Sigma}_2 = \begin{pmatrix} 2 & 0 \\ 0 & .002 \end{pmatrix}$$

a. How would a linear classifier perform when trained to classify between these two classes in the given representation? Why?

b. Suppose PCA is run on this two dimensional data to produce a one dimensional representation. Describe the principal component that PCA would select.

c. How would a linear classifier perform when trained to classify between these two classes in the one dimensional PCA representation? Why?

d. Suppose instead you train a neural network on this dataset. The neural network has a single hidden layer with a single hidden node, and a single output node corresponding to the label. Compare the performance of this neural network to a linear classifier trained on a representation of this data learned by PCA.

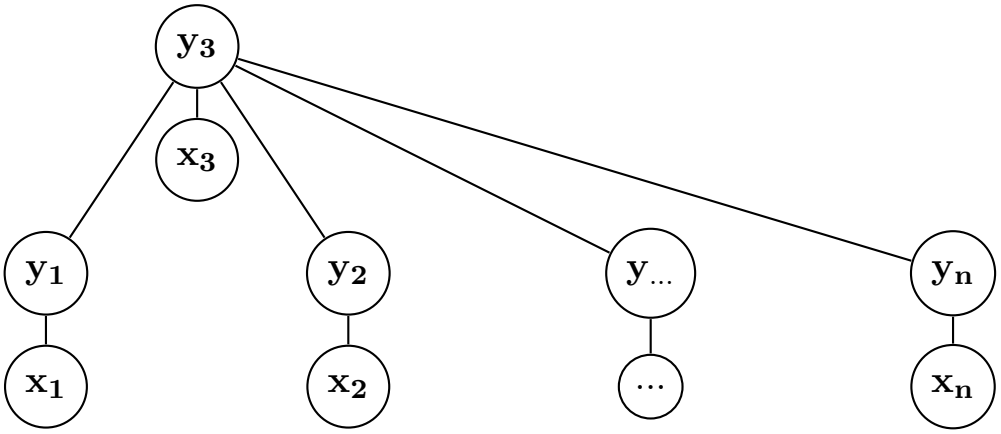
a. For the given data, we have multinomial distribution for each classes. In first class, we have $\mu_{11} = 1$, $\sigma_{11} = 1$, $\mu_{12} = -1$, $\sigma_{12} = 0.001$. In second class, $\mu_{21} = 1$, $\sigma_{21} = 2$, $\mu_{22} = 1$, $\sigma_{22} = 0.002$. By looking at the data above, the classifier will learn the hyperplane that separates the two classes. Hyperplane can be differed by the type of linear classifier. Suppose that we are projecting the data into 2-d grid, where X_1 represents x-axis and X_2 represents y-axis. Then if we run correct classifier algorithm such as SVM (max-margin), then we will most likely get hyperplane that sits on $X_1 = 0$ horizontally.
Slightly more formally, if we run SVM algorithm, then we will update W based on correct or incorrect labels in the training set. After train, the prediction process will find classifier that will sit on $X_1 = 0$ since two classes have small variance respect to X_2 dimension. However, total variance from the class are large as we can see that -1 vs. 1, it is separable by classifier.

b. As we have learned from the lecture, we will first compute eigenvectors and eigenvalues with the given covariance Σ. Then selected component will be come from eigenvector with the highest eigenvalue. When we examine this data, PCA will select data with the largest variance (meaning linear combination of variables with the highest variance). Therefore, PCA will choose dimension 2 data, here is called X_2, because X_2 has larger variance between two classes. Variance of x_{12} and x_{22} are greater than x_{11} and x_{21}.
Even if we drop X_1 and still get a good classifier. Also even if we have small variance within each class for X_2 is small, variance in the whole data (when we look at each class), variance is large.

c. with the dimensionality reduction using one dimension PCA from two classes, this would not make the data more separable during the training because we are reducing dimension which is same as giving away information. However, if we reduce the information, it might be useful for the generalization of the learned model therefore the performance of unseen data will increase. Even though dimension reduction has computational complexity, the algorithm in training and testing can be fasten with the reduction. Overall, performance with single PCA can be generalization with little faster speed but still have some disadvantages of losing information. However, if we correctly analyze the correlation of the features, this also can be prevented.
When we examine at this data case, then classifier using X_2 dimension (one dimension) will perform same as 2-dimension classifier in terms of accuracy. Because we extract classifier from PCA which is X_2 based on the highest variance, X_2 will still catch high variances that are same as original high dimension. This is beauty of PCA method because we still capture similar variance even if we transform to lower dimension classifier. The reason of using PCA and lower dimension is stated above. This data also has same positive aspects in efficiency and cost.

d. The performance of a single layer auto encoder (single layer neural network) is nearly equivalent to PCA because even if W obtained from PCA and neural network are not essentially the same but the subspace spanned by PCA and single layer neural network is same respect to W. This is because single hidden layer can only get linear dimensionality reduction which is also same as PCA. However, PCA is more efficient realization in the form of SVD.
When we look at this data specifically, weight of X_2 goes into our hidden layer Z_1 increases as the training goes further while X_1 diminishes. In terms of correctness, W that are obtained from PCA classifier and neural network are not necessarily same.

However, since they are in the same subspace as I have mentioned above, they will perform similarly as a classifier.



2. (13 points) Consider the graphical model shown above. In this model, $\mathbf{x}$ is a sequence of observations for which we want to output a prediction $\mathbf{y}$, which itself is a sequence, where the size of $\mathbf{y}$ is the same as $\mathbf{x}$. Unlike sequence models we discussed in class, this model has a tree structure over the hidden nodes. Assume that the potential functions have a log-linear form: $\psi(Z) = \exp\{\sum_i \theta_i f_i(Z)\}$, where Z is the set of nodes that are arguments to the potential function (i.e. some combination of nodes in $\mathbf{x}$ and $\mathbf{y}$,) θ are the parameters of the potential functions and f_i is a feature function.

a. Write the log likelihood for this model of a single instance $\mathbf{x}$: $p(\mathbf{y}, \mathbf{x})$.

b. Write the conditional log likelihood for this model of a single instance $\mathbf{x}$: $p(\mathbf{y}|\mathbf{x})$.

c. Assume that each variable y_i can take one of k possible states, and variable x_i can take one of k' possible states, where k' is very large. Describe the computational challenges of modeling $p(\mathbf{y}, \mathbf{x})$ vs $p(\mathbf{y}|\mathbf{x})$.

d. Propose an efficient algorithm for making a prediction for $\mathbf{y}$ given $\mathbf{x}$ and θ.

a. Here is not directed graph, then we will need to bring in cliques concept when we consider joint distribution. Suppose that x_c denotes all nodes in clique C.
$\psi_c(x_c)$ is a potential function over clique C.
But as we can see, only clique that are connected in this tree structure graphical model is x_i and y_i. Therefore, we can write Z_c as set of nodes that are in the clique Or define A as a transition probabilities (matrix) where A_{ij} denotes moving from state i to j. Then,

$$\begin{aligned} p(\mathbf{y}, \mathbf{x}) &= \frac{1}{Z}\Pi_A \psi_A(\mathbf{x}_A, \mathbf{y}_A) \\ &= \frac{1}{Z}\Pi_A \psi_A((\mathbf{x}_A, \mathbf{y}_A) \\ log(p(\mathbf{y}, \mathbf{x})) &= -\sum_{i=1}^{n} log(Z(\mathbf{x}_i)) + \sum_A log(\exp\{\sum_i \theta_i f_i(\mathbf{x}_A, \mathbf{y}_A)\}) \end{aligned}$$

where $Z = \sum_{\mathbf{x},\mathbf{y}} \Pi_A \psi(\mathbf{x}_A, \mathbf{y}_A)$

b. Same here, we will denote A as a transition probabilities (matrix) where A_{ij} denotes moving from state i to j. Then, $p(\mathbf{y}|\mathbf{x}) = \frac{1}{Z}\Pi_A \psi(\mathbf{x}_A, \mathbf{y}_A)$. Here we are given $\psi(Z) = \exp\{\sum_i \theta_i f_i(Z)\}$.

$$log(p(\mathbf{y}|\mathbf{x})) = log(\frac{1}{Z}\Pi_A \psi_A(\mathbf{x}_A, \mathbf{y}_A)) = -log(Z) + \sum_A log(\psi_A(\mathbf{x}_A, \mathbf{y}_A))$$

$$log(p(\mathbf{y}|\mathbf{x})) = -\sum_{i=1}^{n} log(Z(\mathbf{x}_i)) + \sum_A log(\exp\{\sum_i \theta_i f_i(\mathbf{x}_A, \mathbf{y}_A)\})$$

Where $Z = \sum_{\mathbf{y}} \Pi_A \psi(\mathbf{x}_m, \mathbf{y})$
Only difference in here is that conditional probability does not have to learn $p(\mathbf{x})$. Therefore, Z values differ.

c. We are given the information that y_i can take one of k possible states and x_i can take one of k' possible states. And also, $k' >> k$ meaning that k' has a lot more options than k.
Computational challenges of modeling $p(\mathbf{y}, \mathbf{x})$ and $p(\mathbf{y}|\mathbf{x})$ happen when we consider choosing x. Since we are given x when we compute conditional probability, therefore, we do not have to consider k' many states. However, when we compute joint probability, then we have to also consider k' many states of x. Therefore, joint probability is much heavier computation. Whenever y_i chooses state, it only needs to consider states of x_i in joint probability $p(\mathbf{y}, \mathbf{x})$. Therefore, computation challenge of modeling will be $O((k * k')^n)$. However, when we compute conditional probability, x state is given. Therefore, computational challenge of modeling $p(\mathbf{y}|\mathbf{x})$ is $O(k^n)$. Therefore, computationally, join probability is much more challenging than conditional probability.

d. For the tree structured graphical model, we can use $max - product$ algorithm. We substitute factored graph expansion into max probability thus enhancing maximization with products. Final maximization is performed over product of all messages arriving at the root node. In training, take derivative of conditional log likelihood over θ and solve for θ. When computing probabilities (when we predict), we will use max-product algorithm to find the highest probability sequence. This algorithm is also known as Viterbi decoding.