CS 189: Introduction to Machine Learning - Discussion 8

1. Midterm Review

   a) We have trained an SVM with a Gaussian kernel:

   $$K(\mathbf{u}, \mathbf{v}) = e^{\frac{-(\mathbf{u}-\mathbf{v})^2}{2\sigma^2}}$$

   Now we have a set of n support vectors (the training points the SVM keeps) $\{\mathbf{x}^{(i)}\}$, the associated training labels $\{y^{(i)}\}$ and alpha weights $\{\alpha_i\}$.
   How do we classify a new test point $\mathbf{x}$?

   Solution:

   $$\begin{aligned}\hat{y} &= \text{sign}(\mathbf{w}^T\Phi(\mathbf{x})), \text{ and } \mathbf{w} = \sum_{i=1}^{n} \alpha_i \Phi(\mathbf{x}^{(i)}) \\ &= \text{sign}\Big(\sum_{i=1}^{n} \alpha_i \Phi(\mathbf{x}^{(i)})\Phi(\mathbf{x})\Big) \\ &= \text{sign}\Big(\sum_{i=1}^{n} \alpha_i K(\mathbf{x}^{(i)}, \mathbf{x})\Big) \\ &= \text{sign}\Big(\sum_{i=1}^{n} \alpha_i e^{\frac{-(\mathbf{x}^{(i)}-\mathbf{x})^2}{2\sigma^2}}\Big)\end{aligned}$$

   b) What's the difference between perceptron and Hebb's rule?

   Solution: The perceptron algorithm only considers misclassified training points.

   c) What's the difference between the classifier given by an SVM or perceptron algorithm, and logistic regression?

   Solution: In the end, both methods find a linear desicion boundary between two classes. Logistic regression models the probability that a given data point ($\mathbf{x}$) belongs to one of two classes (y_1), specifically $P(Y = 1|X = x)$. Because logistic regression models this probability, we have more than just a classification of a point ($Y = 1$ or $Y = -1$); we also have a measure of the probability, or confidence, that the point belongs to a specific class.

d) What's the difference between generative models and discriminative models?

Solution: Both models seek to find the probability of the output Y given the data X, $P(Y|X)$. Descriminitive techniques model $P(Y|X)$ directly, focusing on a direct mapping from the input data to the output data. Generative techniques model the joint probability of X and Y, $P(X,Y)$, typically modelling $P(X|Y)$ and $P(Y)$ and using Bayes' rule to arrive at $P(Y|X) \propto P(X|Y)P(X)$. Both approaches have models and both use data. The discriminative approach tries to rely heavily on the data, while the generative approach tries to take advantage of prior knowledge or assumptions of how the data was created.

For example, in classification, a discriminitive approach could model $P(Y|X)$ directly as a logistic function, $P(Y = 1|X = \mathbf{x}) = \frac{1}{1+e^{\mathbf{w}^T\mathbf{x}}}$. In this case, the discriminative model includes parameter $\mathbf{w}$ that is chosen to best fit the logistic function to the data pairs X and Y. A generative approach to the same classification problem could model the class-conditional probability as Gaussian and the class probability as uniform. In this case the generative model includes parameters for the mean and variance of the two class-conditional probabilities.

e) What is the difference between LDA and PCA?

Solution: LDA is a supervised learning method. LDA is a generative classifier that assumes that for a certain class ($Y = 1$ or $Y = -1$), the data is drawn from a Gaussian distribution. The Gaussian distribution for each class can have a different mean, but is assumed to have the same covariance across classes. This assumption is necessary for the decision boundary to be linear.

PCA is an unsupervised learning method, so it isn't classifying anything. PCA assumes all the data is drawn from a single (not necessarily Gaussian) distribution with an associated covariance. PCA is a method for finding the vectors along which the data's variance is greatest. This is useful for analyzing the data. One can use this knowledge to reduce the dimension of the data as a preprocessing step, for example, but this is not required as part of PCA (PCA is just an analysis technique).

2. *Optional: Extra for Experts!* Curse of Dimensionality

We have a training set: $(\mathbf{x}^{(1)}, y^{(1)}), \ldots, (\mathbf{x}^{(n)}, y^{(n)})$, $\mathbf{x}^{(1)} \in \mathbb{R}^d$. Our 1-nearest neighbor classifier is:

$$\text{class}(\mathbf{x}) = y^{(i^*)} \quad \text{where } \mathbf{x}^{(i^*)} \text{ is the nearest neighbor of } \mathbf{x}.$$

Assume any data point $\mathbf{x}$ is inside the Euclidean ball of radius 1, i.e. $\|\mathbf{x}\|_2 \leq 1$. To be confident in our prediction, we want the distance between $\mathbf{x}$ and its nearest neighbor to be small, within some positive ϵ:

$$\|\mathbf{x} - \mathbf{x}^{(i^*)}\|_2 \leq \epsilon \quad \text{for all } \|\mathbf{x}\|_2 \leq 1. \tag{1}$$

For this condition hold, at least how many data points should be in the training set? How does this lower bound depend on the dimension d?

Hint: Think about the volumes of the hyperspheres, and use the union bound:

$$\text{vol}(\cup_{j=1}^{k} S_j) \leq \sum_{j=1}^{k} \text{vol}(S_j), \text{ where } S_j \text{ is a hypersphere.}$$

Solution: Let B_0 be the ball centered at the origin, having radius 1 (inside which we assume our data lies). Let $B_i(\epsilon)$ be the ball centered at $\mathbf{x}^{(i)}$, having radius ϵ. For inequality (1) to hold, for any point $\mathbf{x} \in B_0$, there must be at least one index i such that $\mathbf{x} \in B_i(\epsilon)$. This is equivalent to saying that the union of $B_1(\epsilon), \ldots, B_n(\epsilon)$ covers the ball B_0. Let $\text{vol}(B)$ indicate the volume of object B, then we have

$$\sum_{i=1}^{n} \text{vol}(B_i(\epsilon)) = n\text{vol}(B_1(\epsilon)) \geq \text{vol}(\cup_{i=1}^{n} B_i(\epsilon)) \geq \text{vol}(B_0).$$

This implies

$$n \geq \frac{\text{vol}(B_0)}{\text{vol}(B_1(\epsilon))} = \frac{c(1^d)}{c\epsilon^d} = \frac{1}{\epsilon^d}$$

Where the constant c is dependent on the formula for the volume of a hypersphere in d dimensions. This lower bound suggests that to make an accurate prediction on high-dimensional input, we need exponentially many samples in the training set. This exponential dependence is sometimes called the *curse of dimensionality*. It highlights the difficulty of using non-parametric methods for solving high-dimensional problems.