CS 189: Introduction to Machine Learning - Discussion 5

1. General Review

   a) Given a dataset, X, in $\mathbb{R}^n$, and the feature mapping ϕ corresponding to a quadratic kernel, what is the dimensionality of $\phi(X)$? What if the feature mapping was Gaussian/RBF? How do we deal with this high-dimensional data?

   b) Which types of learning problems have we seen so far? What types of models?

   c) What is Hebb's rule? How do we modify Hebb's rule to obtain logistic regression?

2. Feature Selection

Feature selection is the process of selecting a subset of a dataset's original features for usage in learning tasks. Any algorithm for this task has two parts: a search technique for proposing new feature subsets, as well as an evaluation measure which scores the different feature subsets. Wrapper methods involve retraining a model with these different subsets. Filter methods don't interact with the model, but rather just the features–modelling interactions between the features, and eliminating features that are highly correlated. Embedded methods embed the process of feature selection directly into the loss function.

a) Why might we want to perform feature selection?

b) Describe the simplest wrapper method you can think of for performing feature selection. What is its runtime? Can we do better?

c) Now, let's try the embedded method approach to feature selection.

   i) What kind of penalty term corresponds directly to feature selection?

   ii) What issues arise when optimizing a function with this penalty term?

   iii) Describe two vectors such that their ℓ-1 norms are equal, but their ℓ-2 norms are not.

   iv) Assume a method exists for optimizing non-differentiable functions, as long as they are convex.[1] How can we approximate the penalty term from part i, while still maintaining an ”optimizable” loss function?

[1] These are known as proximal gradient methods:
`http://www.eecs.berkeley.edu/~elghaoui/Teaching/EE227A/lecture18.pdf`

3. Virtual Leave One Out Cross-Validation[2]

K-Fold Cross Validation works pretty well, but imagine a case where our training data is really scarce. We could get even more usage out of our data by training on all data points except for one, then attempt to classify the remaining point. This corresponds to setting $K = n$ in K-Fold Cross Validation, and is called Leave-One-Out-Cross-Validation (LOOCV). Although this would seem to require n model fits, we'll derive a method which saves us this expense in the linear regression setting.

Let's write the formal definition of LOOCV:
$LOOCV = \sum_{i=1}^{n} (y_i - \hat{y_i}^{-i})^2$, where $\hat{y_i}^{-i}$ is the estimator of y_i with the i-th data point held-out when fitting the model.

a) After fitting our regression model, we have $\hat{y} = X\hat{w}^T$. With a tiny bit of algebra, we can rewrite our estimate of $\hat{y}$ in terms of the actual labels, $\hat{y} = Hy$, where H is an n-by-n matrix [3]. Write the equation for $\hat{y_i}$ in terms of H and y. Feel free to use the solution for $\hat{w}$ that we have found previously.

b) By definition, $\hat{y}^{-i}$ minimizes $\sum_{j \neq i}(y_j - \hat{y_j}^{-i})^2$. Prove $\hat{y}^{-i}$ minimizes the squared error for Z where

$$Z_j = \begin{cases} y_j, & j \neq i \\ \hat{y_i}^{-i}, & i = j \end{cases}$$

c) Prove $\hat{y_i}^{-i} = \hat{y_i} - H_{ii}y_i + H_{ii}\hat{y_i}^{-i}$

d) Using this result, show that $LOOCV = \sum_{i=1}^{n} (\frac{y_i - \hat{y_i}}{1 - H_{ii}})^2$. With this result, we can implement LOOCV while only fitting one regression model.

e) Give an example of a dataset where LOOCV will give a poor estimate of the error.

[2] Problem Set-Up: Credit to Tom Mitchell and Andrew W. Moore, CMU

[3] This matrix is known as the hat matrix, and as a number of useful statistical properties: `https://en.wikipedia.org/wiki/Hat_matrix`