DSE 220: Machine learning

Lab 1 — Nearest neighbor and logistic regression

1.1 Programming problems

1. *Cross-validation for nearest neighbor classification.*

   Download the `wine` data set from

   `https://archive.ics.uci.edu/ml/datasets/wine`

   This small data set has 178 observations. Each data point x consists of 13 features that capture visual and chemical properties of a bottle of wine. The label $y \in \{1, 2, 3\}$ indicates which of three wineries the bottle came from. The goal is to use the data to learn a classifier that can predict y from x.

   Suppose we use the entire data set of 178 points for 1-NN classification with Euclidean distance. We would like to estimate the quality of this classifier.

   (a) Use **leave-one-out cross-validation** (LOOCV) to estimate the **accuracy** of the classifier and also to estimate the 3×3 **confusion matrix**.

   (b) Estimate the accuracy of the 1-NN classifier using k-fold cross-validation using 20 different choices of k that are fairly well spread out across the range 2 to 100. Plot these estimates: put k on the horizontal axis and accuracy estimate on the vertical axis.

   (c) The various features in this data set have different ranges. Perhaps it would be better to normalize them so as to equalize their contributions to the distance function. There are many ways to do this; one option is to linearly rescale each coordinate so that the values lie in $[0, 1]$ (i.e. the minimum value on that coordinate maps to 0 and the maximum value maps to 1). Do this, and then re-estimate the accuracy and confusion matrix using LOOCV. Did the normalization help performance?

2. *Binary logistic regression.*

   The `heart disease` data set is described at:

   `https://archive.ics.uci.edu/ml/datasets/Heart+Disease`

   The course webpage has a file `heart.csv` that contains a more compact version of this data set with 303 data points, each of which has a 13-dimensional attribute vector x (first 13 columns) and a binary label y (final column). We'll work with this smaller data set.

   (a) Randomly partition the data into 200 training points and 103 test points. Fit a logistic regression model to the training data and display the coefficients of the model. If you had to choose the three features that were most influential in the model, what would they be?

   (b) What is the test error of your model?

   (c) Estimate the error by using 5-fold cross-validation on the training set. How does this compare to the test error?

3. *Stepwise forward selection.*

   Continuing from the previous problem, suppose we want a **sparse** solution: one that uses only a subset S of the 13 coordinates. One way to do this is with ℓ_1-regularized logistic regression. Another method, which we'll investigate here, is **stepwise forward selection**. This is a greedy procedure that chooses one feature at a time. If we want k features total, these features are selected as follows:

   - Let S be empty (this is the set of chosen features)
   - Repeat k times:
     - For every feature $f \notin S$:
       - Estimate the error of a classifier based on features $S \cup \{f\}$
     - Select the feature f with the smallest error estimate
     - Add this feature to S
   - Now learn a model based only on features S

   (a) Use this procedure to find a k-sparse logistic regression solution for the `heart disease` data, for $k = 1, 2, \ldots, 13$. Create a single plot showing the test error and cross-validation error for all these values of k.

   (b) What two features were chosen for $k = 2$? Plot the decision boundary in this case.

1.2 Mini-project: Coordinate descent

In this mini-project we consider a standard unconstrained optimization problem:

$$\min L(w)$$

where $L(\cdot)$ is some cost function and $w \in \mathbb{R}^d$. In class, we looked at several approaches to solving such problems—such as gradient descent and stochastic gradient descent—under differentiability conditions on $L(w)$. We will now look at a different, and in many ways simpler, approach:

- Initialize w somehow.
- Repeat: pick a coordinate $i \in \{1, 2, \ldots, d\}$, and update the value of w_i so as to reduce the loss.

Two questions need to be answered in order to fully specify the updates:

(i) Which coordinate to choose?

(ii) How to set the new value of w_i?

Give answers to these questions, and thereby flesh out a coordinate descent method. For (i), your solution should do something more adaptive than merely picking a coordinate at random.

Then implement and test your algorithm on a *logistic regression* problem, using the `heart disease` data set described above. Your answer should include the following elements.

(a) *A short, high-level description of your coordinate descent method.*

   In particular, you should give a concise description of how you solve problems (i) and (ii) above. Do you need the function $L(\cdot)$ to be differentiable, or does it work with any loss function?

(b) *Convergence.*

Under what conditions do you think your method converges to the optimal loss? There's no need to prove anything: just give a few sentences of brief explanation.

(c) *Experimental results.*

- Begin by running a standard logistic regression solver (e.g., from `scikit-learn`) on the training set. It should not be regularized: if the solver forces you to do this, just set the regularization constant suitably to make it irrelevant. Make note of the final loss L^*.
- Then, implement your coordinate descent method and run it on this data.
- Finally, compare to a method that chooses coordinates i uniformly at random and then *updates* w_i *using your method* (we'll call this "random-feature coordinate descent").
- Produce a clearly-labeled graph that shows how the loss of your algorithm's current iterate—that is, $L(w_t)$—decreases with t; it should asymptote to L^*. On the same graph, show the corresponding curve for random-feature coordinate descent.