250B Problem Set 1

William Fedus

January 2015

1 Nearest Neighbor Classification

We train and test our algorithm on the MNIST dataset of handwritten digits which includes 60,000 training examples and 10,000 test examples. Each handwritten digit is encoded in a 28x28 gray scale image which can be flattened into a 784-dimensional vector, where the intensity of a particular pixel is the value along a certain dimension. In order to reduce the computational complexity of the classification, we seek a smaller set of prototypes within the full 60,000 training set.

1.1 Uniform Random Prototype Selection

For calibration, we first test the performance of the prototype selection against uniformly randomly selected prototypes within the training set.

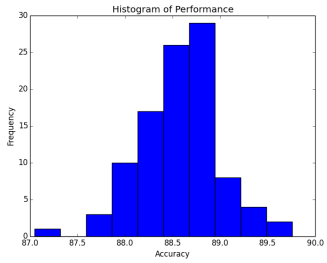

Figure 1: The histogram of the resulting classification accuracy for 1-NN on the test set. These were generated from a random subset of 1000 points and performed 100 times for sufficient statistics.

For randomized subsets of the data, we calculate the 95% error bar on our mean accuracy p via the formula.

$$\text{error} = \pm 2\sqrt{\frac{p(1-p)}{N}} \tag{1}$$

the results of performing 100 runs over the 1000-random subset choice, 50 runs over the 5000-random subset choice and 25 runs over the 10,000-random subset choice resulted in the following performance to within 95% confidence. With these results, we then tested against other

Number of Prototypes	Accuracy	Error
1000 Subset	0.887	$\pm$ 0.06
5000 Subset	0.936	$\pm$ 0.07
10000 Subset	0.948	$\pm$ 0.09

Table 1: Performance of the random subset choice for prototypes. Note that performance monotonically increases with the number of prototypes used for classification.

In order to help guide our search for useful prototypes, we can first consider the classification success within our random prototype selection. In particular, the classification rate of 0 and 1 is very high at 0.96 and 0.99 respectively. This indicates that these classes are generally easy to distinguish from other classes. However, the classification rates of 2, 5 and 8 are only 0.84, 0.83, 0.81; indicating that the vectors in this 784-dimensional space are not highly separated.

1.2 Alternate Prototype Selection

As an alternative prototype selection, we first consider the mean vectors of the 10 label categories. So for instance, for all training examples with label i, we average the vectors to arrive a single average for the entire set with label i. From here, if we are choosing M total prototypes, for each of the 10 labels, we find the closest $M/10$ vectors to the mean vector of the class. This now constitutes our new prototype set.

1.3 Pseudocode

1. For each i in set of labels
    (a) Determine average vector
2. For each n in training example
    (a) If n has label i, calculate distance from average vector
        i. Add to queue with distance as a key
3. For each i in set of labels
    (a) Take the first $M/10$ example from the queue
    (b) Append to subset list to be used for prototypes

1.4 Performance

Unfortunately, the performance of this approach was disappointing and was worse than the random subset for all M as seen in Table 2.

Number of Prototypes	Accuracy
1000 Subset	0.791
5000 Subset	0.840
10000 Subset	0.871

Table 2: Performance of the densest subset choice for prototypes. Note that performance monotonically increases with the number of prototypes used for classification.

The concept of choosing the densest vectors as prototypes around some mean prototype may seem well-motivated, but as this experiment indicates, it is a poor choice for use in 1-NN. The reason is that for harder to distinguish numbers that lie between the clusters in this higher dimensional space, the densest points will not be in the proximity of these and therefore, will provide little classification help.

1.5 Further Steps

Due to the high computational intensity of Nearest Neighbor, I was limited in the amount of experimentation I had time to perform. However, in order to improve beyond the random subset, it could be a very powerful technique to investigate which vectors are finding the most nearest neighbors within the training set. By subsetting our training data, we could potentially build prototypes that are most often helpful and therefore, this would likely improve performance.

2 Bayes Optimality

2.1 Bayes optimal classifier

For the provided probability distribution $\eta(x) = P(Y = 1|X = x)$, the Bayes optimal classifier, $h^*(x)$

$$h^*(x) = \begin{cases} 0, & \text{if } x \leq -0.5 \\ 1, & \text{if } -0.5 \leq x \leq 0.5 \\ 0, & \text{if } x \geq 0.5 \end{cases}$$

.
The optimal Risk is defined as $R^* = E_x \left[\min\left(\eta(x), 1 - \eta(x)\right)\right]$ where we take the expectation over the distribution of x given by $\mu(x)$.

$$R^*(x) = \begin{cases} E_x[0.2], & \text{if } x \leq -0.5 \\ E_x[1-0.8], & \text{if } -0.5 \leq x \leq 0.5 \\ E_x[0.4], & \text{if } x \geq 0.5 \end{cases}$$

if we do the integral to find E_x over each of the three cases, we arrive at the following result,

$$R^*(x) = 0.275 \tag{2}$$

2.2 Training Set with Four Labeled Points

If we use 1-NN to determine our decision boundary, we simply will choose the label of the training example that is closest to our test example choosen along x. Our boundary now becomes

$$\text{1-NN}(x) = \begin{cases} 0, & \text{if } x \leq -0.6 \\ 1, & \text{if } -0.6 \leq x \leq 0.5 \\ 0, & \text{if } x \geq 0.5 \end{cases}$$

The error rate of this classifier, on the underlying distributions μ and η may be calculated by integrating over misclassification risk. In this case, we find a (true) error rate of 0.308 which is necessarily greater or equal to the error rate of the Bayes optimal classifier.

2.3 Cost-Sensitive Risk

Now we adjust our risk function to have asymmetric costs as given by this equation

$$R(h) = c_{01}P(Y = 0, h(X) = 1) + c_{10}P(Y = 1, h(X) = 0) \tag{3}$$

then the classifier that minimizes this cost-sensitive risk is

$$h(x) = 0 \ \forall \ x \tag{4}$$

to see why this is the case, we may consider the more general case below.

2.4 Minimum Cost-Sensitive Risk

Again, our aim is to minimize the cost-sensitive risk,

$$R(h) = c_{01}P(Y = 0, h(X) = 1) + c_{10}P(Y = 1, h(X) = 0) \tag{5}$$

which we may rewrite via the product rule

$$= \int P(X = x)dx \left[c_{01}P(Y = 0, h(X) = 1|X = x) + c_{10}P(Y = 1, h(X) = 0|X = x)\right] \tag{6}$$

however, we see that $P(Y|x)$ and $P(h|x)$ are conditionally independent given x, so we can rewrite this as

$$= \int \left[c_{01}\mathbb{1}_{h(x)=1}P(Y = 0|x) + c_{10}\mathbb{1}_{h(x)=0}P(Y = 1|x)\right] P(x)dx \tag{7}$$

where $\mathbb{1}_{h(x)=0}$ is an indicator function that takes on value 1 if $h(x) = 0$ and value 0 if $h(x) = 1$; vice versa for $\mathbb{1}_{h(x)=1}$.

Now if we simply consider the terms in the square braces $[\dots]$ and minimze these,

$$= c_{01}\mathbb{1}_{h(x)=1}(1 - \eta(x)) + c_{10}(1 - \mathbb{1}_{h(x)=1})\eta(x) \tag{8}$$

we find the whole integral will be minimal when the following condition holds

$$\eta(x) > \frac{c_{01}}{c_{01} + c_{10}} \tag{9}$$

This condition indicates that our minimum cost-sensitive risk is found to be independent of μ, the distribution from which the x are being generated.