Homework 6

Sara Beery
CS 156A - Learning Systems

November 12, 2018

1 Overfitting and Deterministic Noise

Deterministic noise depends on H, as some models approximate f better than others. Assume that $H' \subset H$ and that f is fixed. In general (but not necessarily in all cases), if we use H' instead of H, deterministic noise will increase, as the model will have less flexibility with which to fit the true distribution.

1. **[b]** In general, deterministic noise will increase.

2 Regularization with Weight Decay

In the following problems we use the data provided in the files

http://work.caltech.edu/data/in.dta
http://work.caltech.edu/data/out.dta

as a training and test set respectively. Each line of the files corresponds to a two-dimensional input $x = (x_1, x_2)$, so that $X = \mathbb{R}^2$, followed by the corresponding label from $Y = \{1, 1\}$. We are going to apply Linear Regression with a non-linear transformation for classification. The nonlinear transformation is given by

$$\Phi(x_1, x_2) = (1, x_1, x_2, x_1^2, x_2^2, x_1x_2, |x_1x_2|, |x_1 + x_2|).$$

Recall that the classification error is defined as the fraction of misclassified points.

Recall that the Linear Regression solution is:

$$w = (Z^T Z)^{-1} Z^T y,$$

where $Z = \Phi(X)$. After applying Linear Regression, we find that the in-sample classification error is 0.0286 and the out-of-sample classification error is 0.084. This is closest to:

2. **[a]** $(0.02, 0.08)$

Now we add weight decay to Linear Regression, that is, add the term $\frac{\lambda}{N}\sum_{i=0}^{7} w_i^2$ to the squared in-sample error, using $\lambda = 10^k$. Recall that the Linear Regression solution with weight decay is:

$$w = (Z^T Z - \lambda I)^{-1} Z^T y,$$

After applying Linear Regression with weight decay for $k = -3$ we find that the in-sample classification error is 0.2857 and the out-of-sample classification error is 0.08. This is closest to:

3. **[d]** $(0.03, 0.08)$

Using $k = 3$, we find that the in-sample classification error is 0.3714 and the out-of-sample classification error is 0.436. This is closest to:

4. **[e]** $(0.4, 0.4)$

If we vary the value of k over the set $\{2, 1, 0, -1, -2\}$, we find that $k = -1$ gives us an out-of-sample error of 0.056, which is the smallest value, and thus:

5. **[d]** -1

6. **[b]** 0.06

3 Regularization for Polynomials

Polynomial models can be viewed as linear models in a space Z, under a nonlinear transform $\Phi : X \to Z$. Here, Φ transforms the scalar x into a vector z of Legendre polynomials, $z = (1, L_1(x), L_2(x), ..., L_Q(x))$. Our hypothesis set will be expressed as a linear combination of these polynomials:

$$H_Q(x) = \left\{ h \middle| h(x) = w^T z = \sum_{q=0}^{Q} w_q L_q(x) \right\}.$$

We define

$$H(Q, C, Q_0) = \left\{ h \middle| h(x) = w^T z \in H_Q, w_q = C \ for \ q \geq Q_0 \right\}.$$

Note that

$$H(10, 0, 3) = \left\{ h \middle| h(x) = \sum_{q=0}^{10} w_q L_q(x), w_q = 0 \ for \ q \geq 3 \right\} = \left\{ h \middle| h(x) = \sum_{q=0}^{2} w_q L_q(x) \right\} = H_2,$$

and similarly,

$$H(10, 0, 4) = H_3,$$

Also note that

$$H_a \subset H_b \ \forall \ a < b, (a, b) \in \mathbb{N}^2.$$

Therefore

$$H_a \cap H_b = H_a \ \forall \ a < b, (a, b) \in \mathbb{N}^2,$$

and combining the above, we see that

$$H(10, 0, 3) \cap H(10, 0, 4) = H_2 \cap H_3 = H_2.$$

Thus:

7. **[c]** $H(10, 0, 3) \cap H(10, 0, 4) = H_2$

4 Neural Networks

A fully connected Neural Network has $L = 2; d^{(0)} = 5, d^{(1)} = 3, d^{(2)} = 1$. Only products of the form $w_{ij}^{(l)} x_i^{(l-1)}$, $w_{ij}^{(l)} \delta_j^{(l)}$, and $x_i^{(l-1)} \delta_j^{(l)}$ count as operations (even for $x_0^{(l-1)} = 1$), without counting anything else. The total number of operations in a single iteration of backpropagation (using SGD on one data point) is calculated by counting the number of operations in the forward pass, the number of operations in the backward pass, and the number of operations needed to update the weights.

Assuming that $d(l)$ includes the bias term for each layer, we have 4 active units and one bias unit in layer 0, 2 active units and one bias unit in layer 1, and one active unit (the output) in layer 2. We have weights flowing into all units except the bias units, and weights flowing out of all units. Therefore we have $5 * 2 = 10$ weights between layers 0 and 1, and $3 * 1 = 3$ weights between layers 1 and 2.

The forward pass involves updating the x values for our selected training data point and calculating the loss. This includes an $w_{ij}^{(l)} x_i^{(l-1)}$ operation for each weight, a total of 13 operations.

The backward pass will involve one $w_{ij}^{(l)} \delta_j^{(l)}$ operation per weight, a total of 13 operations.

Updating the weights involves one $x_i^{(l-1)} \delta_j^{(l)}$ operation per weight, an additional 13 operations.

This gives us a total of 39 operations, which is closest to:

8. **[c]** 40

Let us call every node in a Neural Network a unit, whether that unit is an input variable or a neuron in one of the layers. Consider a Neural Network that has 10 input units (the constant $x_0^{(0)}$ is counted here as a unit), one output unit, and 36 hidden units (each $x_0^{(l)}$ is also counted as a unit). The hidden units can be arranged in any number of layers

$l = 1, , L - 1$, and each layer is fully connected to the layer above it.

The minimum possible number of weights will occur if each active hidden unit is in a separate layer, taking into account that each layer needs a bias term ($x_0^{(l)} = 1$) as well as at least one active unit. After the first 10 input units you have a long chain of 18 hidden layers, each with a single active unit and a bias unit, and then the output layer of a single unit. The bias units do not have weights flowing into them, but do have weights flowing out of them. This gives you $10 * 1 + 18 * (2 * 1) = 46$ weights.

9. [**b**] 46

The maximum possible number of weights will occur if you have two hidden layers, the first containing 21 active units and one bias unit, and the second containing 13 active units and one bias unit. This gives you $10*21+22*13+14 = 510$ weights. I found this architecture by looping over potential combinations of layers in python.

10. [**e**] 510