10-701/15-781 Machine Learning, Fall 2003

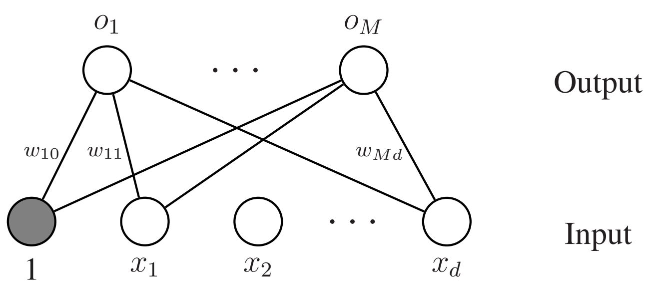
Homework 2 Solution

If you have questions, please contact Jiayong Zhang <zhangjy@cs.cmu.edu>.

---

1. (Error Function) The sum-of-squares error is the most common training criterion for neural nets primarily because of its analytical simplicity. Nevertheless, there are many other possible choices of error function which can be considered depending on the particular application. In this excise you will explore one of such alternatives, which is designed to better approximate the sample error.

   Consider the following simple two-layer neural network for M-category classification.



   The network has M output units. Each has a sigmoid activation function. There are no hidden units. Given a set of training samples $\{(\mathbf{x}_n, y_n)\}_{n=1}^N, y_n \in \{1, \ldots, M\}$, the sum-of-squares criterion E_{MSE} can be written as

$$E_{MSE} = \sum_{n=1}^{N} E_n = \sum_{n=1}^{N} \sum_{m=1}^{M} \left(t_m^{(n)} - o_m^{(n)}\right)^2$$

   where $\mathbf{t}^{(n)} = (t_1^{(n)}, \ldots, t_M^{(n)})$ is the binary valued target vector associated with the n-th sample

$$t_m^{(n)} = \begin{cases} 1, & m = y_n \\ 0, & \text{otherwise.} \end{cases}$$

   The sample error is defined as $E_{MCE} = \sum_n \ell_n$, where

$$\ell_n = \begin{cases} 0, & o_{y_n}^{(n)} = \max_j o_j^{(n)} \\ 1, & \text{otherwise.} \end{cases}$$

   (a) (**10 pts**) Briefly state possible strengths and weaknesses of E_{MCE} compared to E_{MSE}.

   E_{MSE} is analytically simple (e.g. it allows the minimization with respect to the output weights to be expressed as a linear optimization problem which can be solved in closed form). The output of a network trained by minimizing E_{MSE}

approximate the posterior probabilities $P(\mathcal{C}_m|\mathbf{x})$, which can be related to the optimal classification through Bayes rule. However, such a probabilistic interpretation dependends on a sufficient large number of training samples and hidden units. E_{MSE} can be justified by maximum likelihood on the assumption of Gaussian distributed target data. However, the target values for the classification coding scheme are binary.

E_{MCE} is directly related to the optimal classification. It avoids the more general problem of density estimation as an intermediate step. However, E_{MCE} is a noncontinuous function, thus cannot be differentiated. In addition, E_{MCE} may not guarantee good generalization performance as it simply measures the training set error.

(b) (**10 pts**) Define

$$L_n = \frac{1}{1 + e^{-\xi(d_n+\alpha)}},$$
$$d_n = -o_{y_n}^{(n)} + \left[\frac{1}{M-1}\sum_{m,m\neq y_n} {o_m^{(n)}}^\eta\right]^{1/\eta}.$$

Briefly explain why L_n can be used to approximate ℓ_n by choosing appropriate parameters ξ, α and η.

i. When $\alpha = 0$ and ξ is sufficiently large, L_n approximates the step function with respect to d_n.

ii. When η is sufficiently large,

$$d_n \begin{cases} < 0 & \text{if } y_n = \arg\max_j o_j^{(n)}, \\ > 0 & \text{otherwise.} \end{cases}$$

As an extreme, $\lim_{\eta\to\infty} d_n = -o_{y_n}^{(n)} + \max_{m,m\neq y_n} o_m^{(n)}$.

(c) (**10 pts**) Find the network update rule when using the criterion function $E = \sum_n L_n$.

Let β be the learning rate parameter, the batch weight update rule will be

$$\Delta w_{ji} = -\beta \sum_n \frac{\partial L_n}{\partial w_{ji}}$$

So the main task is to compute the term $\partial L_n/\partial w_{ji}$. For simplicity, we will ignore the superscripts of sample index (n).

By chain rule,

$$
\begin{aligned}
\frac{\partial L_n}{\partial w_{ji}} &= \frac{\partial L_n}{\partial d_n} \cdot \sum_{m=1}^{M} \frac{\partial d_n}{\partial o_m} \cdot \frac{\partial o_m}{\partial w_{ji}} \\
&= \frac{\partial L_n}{\partial d_n} \cdot \frac{\partial d_n}{\partial o_j} \cdot \frac{\partial o_j}{\partial w_{ji}} \qquad \text{// Note that } \frac{\partial o_m}{\partial w_{ji}} = 0 \text{ if } m \neq j.
\end{aligned}
$$

As o_j is the output from a sigmoid unit, we have

$$\frac{\partial o_j}{\partial w_{ji}} = o_j(1 - o_j) \cdot x_i$$

Similarly, following the fact that L_n is a sigmoid with respect to $\xi(d_n + \alpha)$,

$$\frac{\partial L_n}{\partial d_n} = L_n(1 - L_n) \cdot \xi$$

If $j = y_n$, then $\partial d_n / \partial o_j = -1$. Otherwise,

$$
\begin{aligned}
\frac{\partial d_n}{\partial o_j} =& \frac{1}{\cancel{\eta}} \left[\frac{1}{M-1} \sum_{m, m \neq y_n} {o_m}^{\eta} \right]^{\frac{1}{\eta} - 1} \cdot \frac{\cancel{\eta}}{M-1} o_j^{(\eta-1)} \\
=& \frac{1}{M-1} \left[\frac{1}{M-1} \sum_{m, m \neq y_n} {o_m}^{\eta} \right]^{-\frac{\eta-1}{\eta}} o_j^{(\eta-1)} \\
=& \frac{1}{M-1} \left[\frac{o_j}{d_n + o_{y_n}} \right]^{(\eta-1)}
\end{aligned}
$$

To summarize, the complete expression for $\partial L_n / \partial w_{ji}$ is

$$
\frac{\partial L_n}{\partial w_{ji}} = \begin{cases} -\xi L_n(1 - L_n) o_j(1 - o_j) x_i & \text{if } j = y_n, \\ \dfrac{\xi}{M-1} L_n(1 - L_n) \left[\dfrac{o_j}{d_n + o_{y_n}} \right]^{(\eta-1)} o_j(1 - o_j) x_i & \text{otherwise.} \end{cases}
$$

(d) (**5 pts**) What problem would you expect to arise in network training if we choose $\xi \gg 0$?

The larger ξ, the more close L_n is to the step function. As a result, there will be more local minima in the energy function. A practical coase-to-fine trick is to start from small ξ and gradually increase with time.

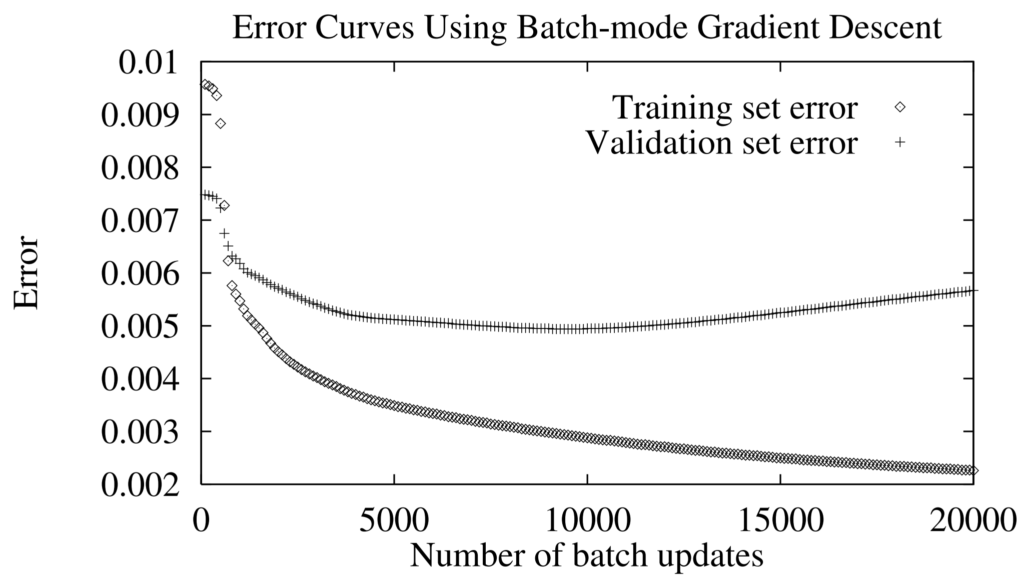
(e) (**5 pts**) Can you think of any continuous function other than L_n that is also able to approximate ℓ_n closely?

Softmax: $$L_n = 1 - \frac{\exp(o_{y_n}^{(n)}/\sigma^2)}{\sum_{m=1}^{M} \exp(o_m^{(n)}/\sigma^2)}$$

Arc Tangent: $$L_n = 0.5 + \frac{1}{\pi} \arctan\left[\xi(d_n + \alpha)\right]$$

Hyperbolic Tangent: $$L_n = 0.5 + 0.5 \tanh\left[\xi(d_n + \alpha)\right]$$

2. (Cross-validation) In a medical diagnosis problem, we want to train a neural network using the Back-propagation algorithm. In order to determine the amount of training, a simple validation technique is employed. Specifically, we randomly split the available samples into two equal sized parts, one for training and the other for validation. The error curves on these two sets are shown in the following plot. Note that the training error decreases monotonically against the number of batch updates, whereas the validation error does not.



Suppose now that we were to retrain the same neural network using exactly the same algorithm, but using ten times as much available data (50% for training and 50% for test).

(a) (**10 pts**) Would you expect the training curve to be different? If so, draw what you would expect. You only need to give a qualitative sketch. In either case, explain your reasoning.

(b) (**10 pts**) Would you expect the validation curve to be different? If so, draw what you would expect. You only need to give a qualitative sketch. In either case, explain your reasoning.

Notice that as the number of training and test examples approach infinity, these two curves must become identical. At that piont, they are both going to be measuring the true error of the learned hypothesis. Thus, as the volume of data grows, the two curves will move toward a line between these two. The validation set curve will move downward, because the true error will decrease. The training curve will move upward at the same time, because overfitting will decrease. In other words, the optimistically biased estimate of true error given by the training error will become less biased as the volume of training data approaches infinity.

Suppose now that we replace the simple validation with m-fold cross-validation.

(c) (**10 pts**) True or false, because each of the test sets are independent, the validation error curves from each fold are also independent. Explain your reasoning.

False. The training sets are correlated because they share examples. They become increasingly correlated for higher m.

(d) (**Optional, 5 pts**) True or false, there is an *a priori* good choice of m for us to decide the amount of training. Explain your reasoning.

m-fold cross-validation is an unbiased estimator for the expected accuracy of sample size $N - N/m$.

If you want to estimate the final accuracy of a classifier trained on N examples, then m-fold cross-validation is biased. The bias can be reduced by increasing the number of folds. But increasing it too much may increase the variance, since higher m increases the correlation in the training sets. The extent of such a trade-off depends on the underlying data distribution and the learning method you are using.

When cross-validation is used for model selection (e.g. selecting the early stopping for neural nets, where we are interested in the difference between two classifiers), the variance may be even more important assuming the bias affects all classifiers similarly. In such cases lower m values are often recommended. However, as m decreases, there is variance due to the instability of training set themselves, leading to an increase in variance.

3. (PAC Learning) Consider the hypothesis class H_{rd2} of "regular, depth-2 decision trees" over n Boolean variables. A "regular, depth-2 decision tree" is a depth-2 decision tree (a tree with four leaves, all distance 2 from the root) in which the left and right child of the root are *required to contain the same variable*. For instance, the following tree is in H_{rd2}.

```
        x3
      /    \
    x1      x1
   /  \    /  \
  +    -  -    +
```

(a) (**10 pts**) As a function of n, how many syntactically distinct trees are there in H_{rd2}? By "syntactically distinct", we mean trees that look different but may still represent the same function.

A H_{rd2} tree is constructed by filling blanks in the following structure. Two trees are syntactically different if they are different in any of the seven fields. In that case, the number of trees is $2^4 n(n-1)$.

```
             (   )
            /     \
       (   )       (   )
      /    \       /   \
  (   ) (   ) (   ) (   )
```

(b) (**10 pts**) Give an upper bound for the number of examples needed in the PAC model to learn any target concept in H_{rd2} with error ϵ and confidence δ.

$$m \geq \frac{1}{\epsilon} \left[\ln |H_{rd2}| + \ln(1/\delta)\right]$$

where $|H_{rd2}| = 2^4 n(n-1)$.

(c) (**10 pts**) Consider the following WEIGHTED-MAJORITY algorithm for the class H_{rd2}. You begin with all hypotheses in H_{rd2} assigned an initial weight equal to 1. Every time you see a new example, you predict based on a weighted majority vote over all hypotheses in H_{rd2}. Then, instead of eliminating the inconsistent trees, you cut down their weight by a factor of 2. How many mistakes will this procedure make at most, as a function of n and the number of mistakes of the best tree in H_{rd2}?

$$m \leq 2.4 \left[k + \log_2 |H_{rd2}|\right]$$

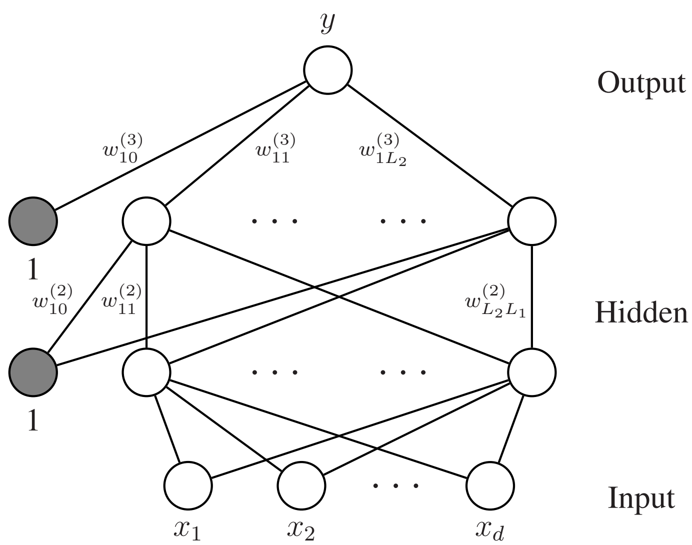
where $|H_{rd2}| = 2^4 n(n-1)$, and k is the number of mistakes made by the best tree in H_{rd2}.

(d) (**Optional, 5 pts**) Derive the number of semantically distinct trees in H_{rd2}, which leads to a tighter bound in (b).

The number of semantically distinct trees is the number of Boolean functions $f : \{0,1\}^n \to \{\pm 1\}$ that H_{rd2} can implement. These functions can be divided into three categories:

i. Functions of zero variable. There are only 2 possibilities: $f = 1$ or $f = -1$. They correspond to trees where all leaf nodes are labeled the same: $++++$ and $----$.

ii. Functions of one variable. There are $2n$ such functions, 2 for each of n variables. They correspond to trees with leaf node labellings $++--$, $--++$, $+-+-$ and $-+-+$.



iii. Functions of two variables. For each combination of two variables a and b, there are ten functions: $a \wedge b$, $\neg a \wedge b$, $a \wedge \neg b$, $\neg a \wedge \neg b$, $(a \wedge b) \vee (\neg a \wedge \neg b)$ and their negates. They correspond to trees with leaf node labellings $+---$, $-+--$, $--+-$, $---+$, $+--+$ and $-+++$, $+-++$, $++-+$, $+++-$, $-++-$.

So the final number is $2 + 2n + 10C_n^2 = 5n^2 - 3n + 2$.

4. (**Optional, 10 pts**, Radial Basis Function) Consider the following four-layer neural network.

This network consists of an input layer, two hidden layers, and an output layer. The first hidden layer has L_1 units, each of which computes the Gaussian radial basis function

$$o_j^{(1)} = \phi_j(\mathbf{x}) = \exp\left\{-\frac{\|\mathbf{x} - \boldsymbol{\mu}_j\|^2}{2\sigma_j^2}\right\} \qquad (j = 1, \ldots, L_1),$$

where $\mathbf{x}$ is the d-dimensional input vector with elements x_i, and $\boldsymbol{\mu}_j$ is the vector determining the center of basis function ϕ_j. The second hidden layer and the output layer consist of units with activation functions $f^{(2)}$ and $f^{(3)}$ respectively.

$$o_j^{(l)} = f^{(l)}(net_j) = f^{(l)}\left(\sum_{i=1}^{L_{l-1}} o_i^{(l-1)} w_{ji}^{(l)} + w_{j0}^{(l)}\right) \qquad (l = 2, 3).$$

Note that the forms of $f^{(l)}$ are left unspecified. You are given a set of training data $\{(\mathbf{x}_n, y_n)\}_{n=1}^N$.

(a) Suppose $y_n \in (0, +\infty)$, specify some network structure and parameter settings that allow the entire network to exactly implement the kernel regression estimate:

$$y(\mathbf{x}) = \frac{\sum_n y_n \exp\{-\|\mathbf{x} - \mathbf{x}_n\|^2/2h^2\}}{\sum_n \exp\{-\|\mathbf{x} - \mathbf{x}_n\|^2/2h^2\}}.$$

$$
\begin{array}{lll}
L_1 = N & L_2 = 2 & \\
f^{(2)} = \ln(\cdot) & f^{(3)} = \exp(\cdot) & \\
\boldsymbol{\mu}_j = \mathbf{x}_j & \sigma_j^2 = h^2 & \\
w_{10}^{(2)} = w_{20}^{(2)} = 0 & w_{1j}^{(2)} = y_j, w_{2j}^{(2)} = 1, & j = 1, \ldots, N \\
w_{10}^{(3)} = 0 & w_{11}^{(3)} = 1, w_{12}^{(3)} = -1 &
\end{array}
$$

(b) Suppose $y_n \in \{-1, +1\}$, and we further assume that both $p(\mathbf{x}|y = +1)$ and $p(\mathbf{x}|y = -1)$ follow the unimodal multivariate normal distribution with isotropic covariance

$$
p(\mathbf{x}|y = \pm 1) = \frac{1}{(2\pi)^{d/2}\lambda_{\pm 1}^d} \exp\left\{-\frac{\|\mathbf{x} - \boldsymbol{\gamma}_{\pm 1}\|^2}{2\lambda_{\pm 1}^2}\right\}.
$$

Specify some network structure and parameter settings that allow the entire network to output exactly the posterior probability $P(y = +1|\mathbf{x})$ when the number of training samples $N \to \infty$.

$$
\begin{array}{ll}
L_1 = 2 & L_2 = 2 \\
f^{(2)} = \ln(\cdot) & f^{(3)} = \exp(\cdot) \\
\boldsymbol{\mu}_1 = \hat{\boldsymbol{\gamma}}_{+1} & \sigma_1^2 = \hat{\lambda}_{+1}^2 \\
\boldsymbol{\mu}_2 = \hat{\boldsymbol{\gamma}}_{-1} & \sigma_2^2 = \hat{\lambda}_{-1}^2 \\
w_{10}^{(2)} = w_{12}^{(2)} = w_{20}^{(2)} = 0 & \\
w_{11}^{(2)} = P(y = +1)/\lambda_{+1}^d & w_{21}^{(2)} = P(y = +1)/\lambda_{+1}^d, w_{22}^{(2)} = P(y = -1)/\lambda_{-1}^d \\
w_{10}^{(3)} = 0 & w_{11}^{(3)} = 1, w_{12}^{(3)} = -1
\end{array}
$$