6.891 Machine learning and neural networks

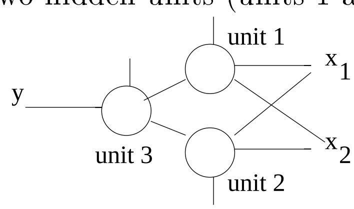
Mid-term exam

October 26, 2000

(2 points) Your name and MIT ID:

Problem 1

1. **(6 points)** Consider a two-layer neural network with two inputs x_1 and x_2, one output unit (unit 3) and two hidden units (units 1 and 2).



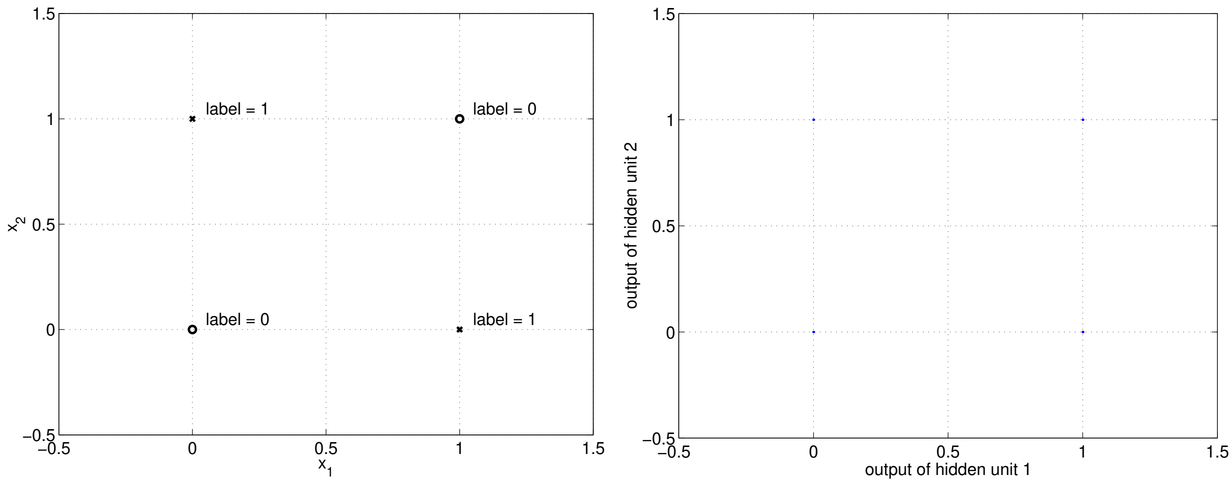
We assume that the transfer functions in the network are threshold functions. For example, unit 1 with two inputs x_1 and x_2 produces an output

$$y = \text{step}(\, w_{10} + w_{11}x_1 + w_{12}x_2 \,) \tag{1}$$

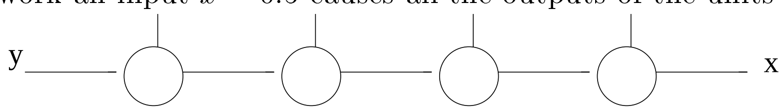
where $\text{step}(z) = 1$ if $z \geq 0$ and zero otherwise.

Show that such a network can solve the XOR problem in the left figure below.

In the left figure, mark the decision boundaries for the two hidden units corresponding to the solution. Also mark on each side of the boundary what the resulting output is for the corresponding unit. Map the outputs to the figure on the right and indicate which of the four input examples the outputs correspond to. Finally, draw the decision boundary corresponding to the output unit 3.



2. **(6 points)** Suppose we are trying to train the following chain like neural network with back-propagation. Assume that the transfer functions are logistic functions and that all the weights are initially set to 1 and all the biases are set to -0.5. Giving such a network an input $x = 0.5$ causes all the outputs of the units to become 0.5.



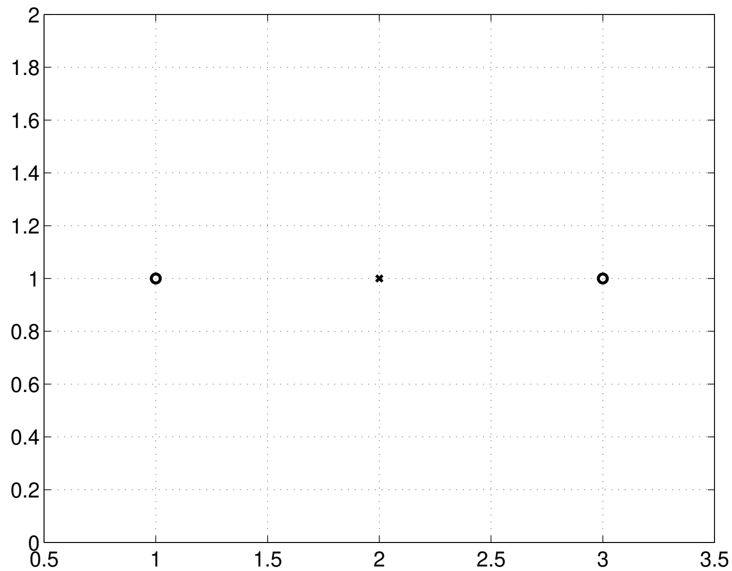
Now, given a single input $x = 0.5$ and corresponding target output $y = 1$, what can you say about the order of magnitude of the gradient updates for weights in this network? We are looking for a qualitative (not numerical) answer comparing the magnitude of the updates across the different units.

3. **(T/F – 2 points)** Applying back-propagation to train a neural network is guaranteed to find the globally optimal solution

4. **(T/F – 2 points)** Regardless of the choice of the transfer function, setting all the weights close to zero in a neural network makes the network function like a linear mapping from inputs to outputs

Problem 2

1. **(4 points)** We want to solve the following classification problem with boosting where the component classifiers are decision stumps. In the figure below, a) mark the decision boundary for the first decision stump and b) circle the points whose weight will *increase* as a result. Indicate the positive/negative side of the decision boundary. 'o's in the figure correspond to -1 labels and 'x's denote $+1$ labels.



2. **(4 points)** Give us the new weights on the three training examples after the first boosting iteration (you should not need a calculator for this)

3. **(6 points)** How many boosting iterations would we need in the previous example so that the combined classifier separates the examples perfectly? Please answer either a) 1 iteration b) 2 iterations, c) at least 3 iterations or d) boosting cannot separate the examples. Briefly explain why.

4. **(4 points)** Explain briefly why we might ever want to use boosting as opposed to some other way of combining simple classifiers into strong classifiers (such as adapting the forward-fitting algorithm to classification problems)

5. **(T/F – 2 points)** The weighted error of each of the component classifiers (error relative to the current weights on the training examples) always goes down as a function of the number of boosting iterations

6. **(T/F – 2 points)** The training error of the combined classifier decreases monotonically as a function of boosting iterations

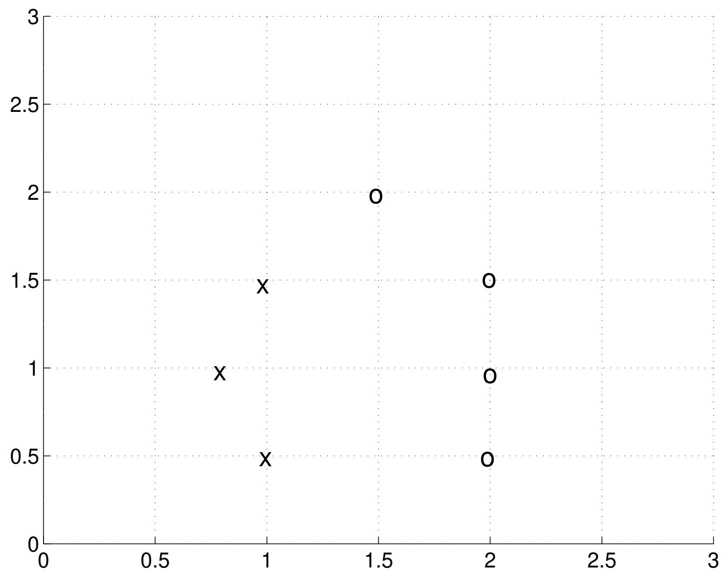
7. **(T/F – 2 points)** After some k boosting iterations, the next component classifier may achieve only weighted training error close to 0.5 (relative to the current weights on the examples) even if the combined classifier perfectly separates the training examples

Problem 3

1. **(6 points)** Give us the simplest kernel function that permits support vector machines to represent all the decision boundaries that a mixture of two Gaussian model can

(without any constraints). Also provide a brief justification. We assume that the two components in the mixture model correspond to the binary classification labels.

2. **(4 points)** In the figure below, draw the maximum margin linear decision boundary and indicate the resulting support vectors by circling them.



3. **(4 points)** Suppose we are given a sequence of training examples $\mathbf{x}_1, \mathbf{x}_2, \ldots$ as well as the corresponding binary ± 1 labels $y_1, y_2, \ldots$. We predict the label for the n^{th} example by training our binary classifier, say $h(\mathbf{x}; \mathbf{w})$, with the first $n-1$ examples and labels and, subsequently, predicting the n^{th} label as the output of $h(\mathbf{x}; \hat{\mathbf{w}}_{n-1})$ ($\hat{\mathbf{w}}_{n-1}$ here denote the parameters that we find on the basis of the first $n-1$ training examples). Such a situtionn might arise, for example, in predicting changes in stock prices, where $\mathbf{x}$ captures our knowledge of the current situation.

   We wish to use support vector machines for this classification task. Give two reasons for why our ability to predict the n^{th} label might not improve or might even get worse as the number of training examples or n increases (up to say $n = 1000$). Your reasons may pertain to assumptions or properties of support vector machines.

4. **(4 points)** Give one reason for and against maximum margin separation of the training examples

5. **(4 points)** Briefly explain how we should set the constant C (level of regularization) in a regularized logistic regression:

$$J(\mathbf{w}) = \sum_{i=1}^{n} \log P(y_i|\mathbf{x}_i, \mathbf{w}) - \frac{C}{2}\|\mathbf{w}\|^2 \tag{2}$$

6. **(4 points)** Suppose we use the simple mutual information criterion $I(y; x_j)$ to select components x_j of the input vectors $\mathbf{x}$. We assume that we have a large number of training examples and the probabilities in $I(y; x_j)$ can be estimated accurately. Based on the resulting values for $I(y; x_j)$ can we ever definitely include/exclude a feature? Briefly explain why or why not.

7. **(T/F – 2 points)** The computational cost of estimating support vector machines increases linearly with the number of training examples

8. **(T/F – 2 points)** A polynomial kernel function of degree one can solve the XOR-problem discussed in problem 1

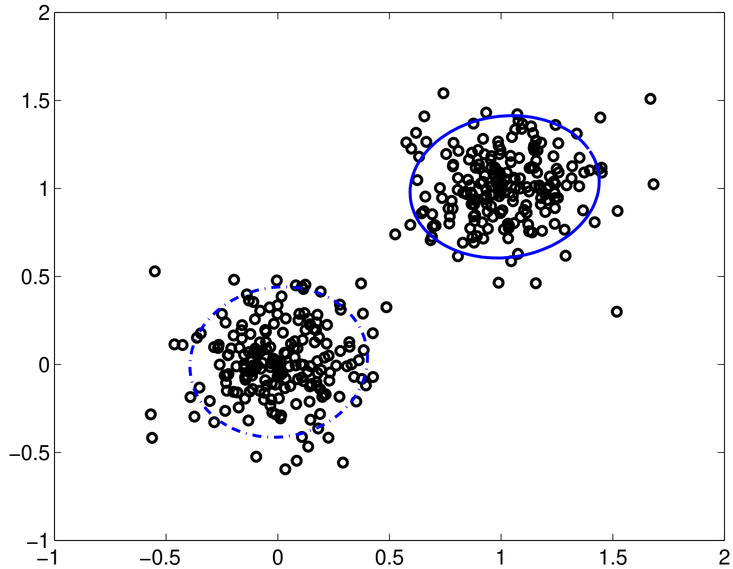
9. **(T/F – 2 points)** As far as generalization error is concerned, the choice of the kernel function matters only in terms of what the resulting VC-dimension is

Problem 4

1. **(4 points)** Give one reason for why we would want to use the EM- algorithm for training mixture models rather than trying to maximize the log-likelihood of the training data via gradient ascent.

2. **(3 points)** Why doesn't the EM-algorithm converge after one iteration?

3. **(3 points)** In the figure below, what is *not explicit* in the representation of the mixture of two Gaussians? In other words, what cannot I get even approximately from the figure? These are the type of figures that you have seen in lectures in the context of the EM algorithm.



4. **(4 points)** Suppose the mixture model given in the previous figure serves as our initial guess for the EM-algorithm. Could the EM- algorithm under any initial choice of the remaining parameters, those that are not specified in the figure, move the "solid" Gaussian over both clusters after one iteration? Briefly explain why or why not.

5. **(T/F – 2 points)** When the EM-algorithm converges for a mixture of Gaussians model, the mixing proportions $p_1, \ldots, p_k$ become equal to the posterior component probabilities $P(y = j|\mathbf{x}_i)$ for all training examples

6. **(T/F – 2 points)** Since Parzen windows is a non-parametric density estimation method the properties of the kernel bumps we assign over each training example are irrelevant.