6.034 Quiz 1, Spring 2007

1 Decision Trees (10 points)

1. (5 pts) Write an expression for the average entropy of the test $x_2 > -0.5$ for the following data:

i	x_1^i	x_2^i	y^i
1	-1	-1	-1
2	-1	1	-1
3	-1	0	1
4	1	1	1

You do not need to find the final numerical value, but do not leave any variables in your expression.

$$AE = \frac{1}{4}E_1 + \frac{3}{4}E_2$$
$$E_1 = 0$$
$$E_2 = -\frac{2}{3}\log_2\frac{2}{3} - \frac{1}{3}\log_2\frac{1}{3}$$

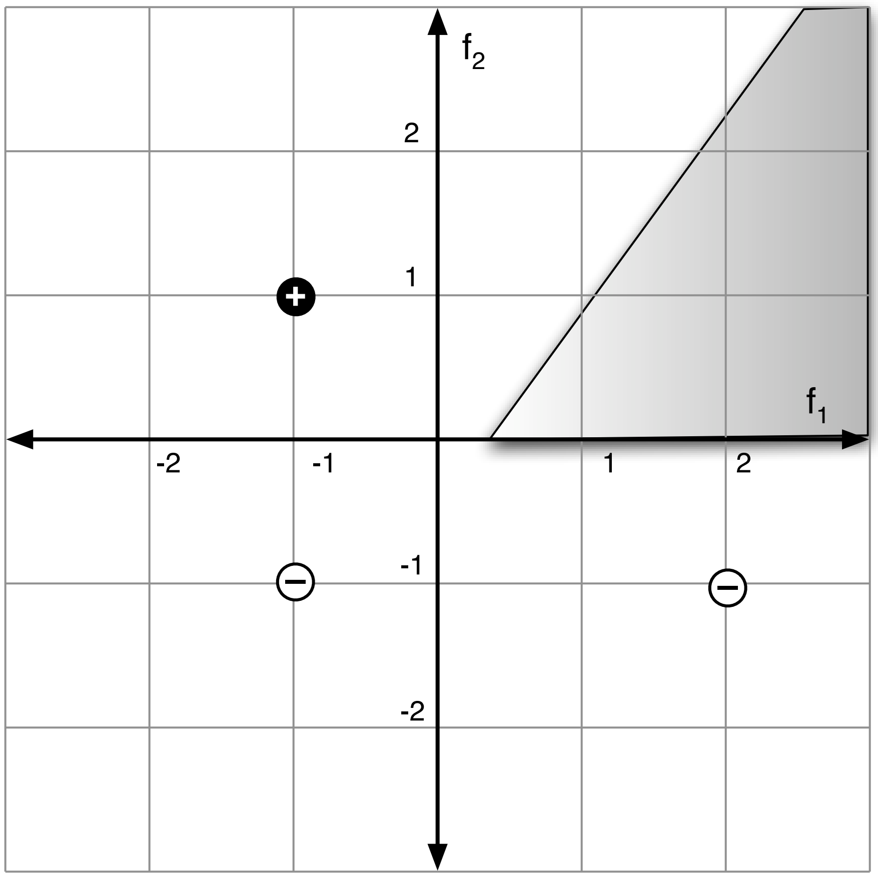
2. (5 pts) For the data above, write a test (of the form $x_i > value$) considered by the decision tree algorithm, that has average entropy of 1.

$$x_2 > 0.5$$

Note that there are other tests that don't split the data, which trivially have entropy of 1, but those would not be considered by the decision tree algorithm.

2 Nearest Neighbors (10 points)

In the following figure, shade in the region in the feature space below where the predictions of 1-nearest-neighbors would disagree with the predictions from a standard decision tree. The training data are only the three points shown.

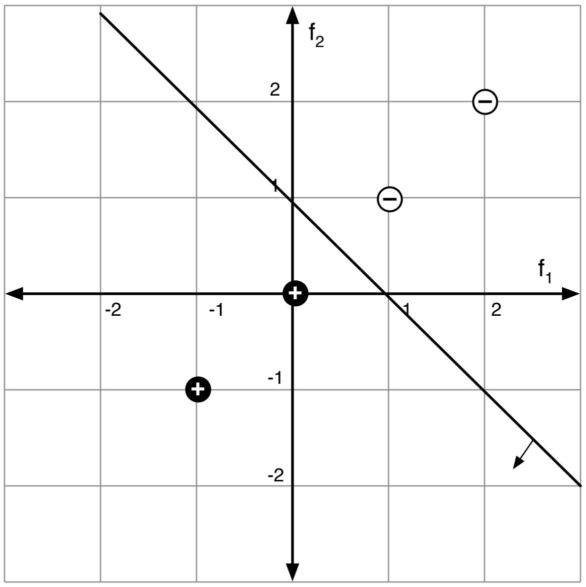


In the shaded region, nearest neighbors predicts the negative class while a decision tree would predict the positive class.

3 SVM (15 points)

Given the following training data:

i	x_1^i	x_2^i	y^i
1	0	0	1
2	-1	-1	1
3	1	1	-1
4	2	2	-1



the Lagrange parameters α_i that would be found for a linear kernel SVM are:

1. $\alpha_1 = -1, \alpha_4 = -1$
2. $\alpha_1 = -1, \alpha_3 = -1$
3. $\alpha_1 = 1, \alpha_4 = 1$
4. $\alpha_1 = 1, \alpha_3 = 1$ is correct
5. $\alpha_1 = 2, \alpha_4 = 2$
6. $\alpha_1 = 2, \alpha_3 = 2$

and the value of the offset is:

1. $b = 1$ is correct
2. $b = 0$
3. $b = -1$

Circle all the true choices.

Note that all the α_i are required to be positive, so that rules out the first two choices. We also know that $\sum_i \alpha_i y^i = 0$ but all the remaining choices satisfy this condition. However, we can see by inspection that points 1 and 3 are the support vectors, so $\alpha_2 = \alpha_4 = 0$, thus ruling out choices 3 and 5. So, only choices 4 and 6 are plausible.

Recall that the weight vector will be a linear combination of the support vectors:

$$[w_1, w_2] = (\alpha_1)(1)[0,0] + (\alpha_3)(-1)[1,1] = [-\alpha_3, -\alpha_3]$$

We also know that the margin for the support points must be equal to 1. Simply substitute point 3 into the definition of margin ($y^i(w \cdot x^i + b)$):

$$-1([-\alpha_3, -\alpha_3] \cdot [1,1] + b) = -1(-2\alpha_3 + b) = 1$$

Similarly, for point 1

$$1([-\alpha_3, -\alpha_3] \cdot [0,0] + b) = b = 1$$

From these we can see that $\alpha_3 = 1$ and $b = 1$.

4 Perceptron (20 points)

We saw that it was possible to generalize SVMs to kernel SVMs so as to handle non-linearly separable data. We can generalize the perceptron to a kernel perceptron in a similar fashion. Indicate precisely how to generalize the learning algorithm **and** the use of the perceptron to classify data, given a kernel $K(.,.)$.

A simple modification of the dual perceptron algorithm gives us a kernel perceptron, this is essentially identical to what we did for SVM's:

- *Evaluating the margin is normally done by* $\sum_{j=1}^{m} \alpha_j y^j x^j \cdot x^i$*, we replace that with* $\sum_{j=1}^{m} \alpha_j y^j K(x^j, x^i)$
- *In the kernel case, we don't return a weight vector, we just return the non-zero* α_j *and the corresponding* x_j *(the support vectors), this is what we will need to do the classification.*
- *Classification of an unknown point* u *is done by* $sign(\sum_{j=1}^{m} \alpha_j y^j K(x^j, u))$

Many people tried to modify the non-dual form of the perceptron algorithm by using $K(w, x^i)$ *to compute the margin. However, then they had to attempt to deal with the update rule that mentions* w *and* x^i *explicitly (not in dot product) by using the actual feature mapping function. However, the whole point of the kernel is that we* **never** *want to use that feature mapping (which may actually not be finitely representable), we want to* **only** *use dot products (the kernel values). There are a lot of additional conceptual problems with this approach, e.g. it needs to add transformed feature vectors to get a weight vector, which now would have the dimensionality of the high dimensional space, which would mean that it is the wrong thing to use in connection with the kernel in the first place.*

5 Regression (20 points)

In the notes, we saw regression trees, which split the data so as to minimize average variance and then return the average y values of the data at the leaf nodes of the tree.

Imagine that we want to, instead, fit straight lines to the data at the leaf nodes. That is, we store a linear equation $y = \sum_n a_n x_n + b$ at each leaf. To make a prediction for a new point, we find the correct leaf and then substitute the features of the point into the equation to get a predicted y.

What measure would you use for deciding which feature to split the data on while building the tree? Explain why your measure is a good choice.

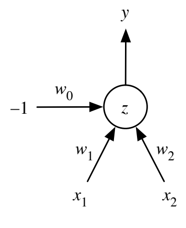
You want to measure how well the best lines fit the data sets produced by a split.

To evaluate a feature test, find the best line fit (by whatever method) for the data sets produced by the split, measure the error in the fit, for example, the sum of the square of the actual y's and the predicted y's from the equation. Then, find the weighted average of this error for the two splits and use that in the same way we have used entropy and variance in previous decision-tree algorithms.

Note that we want to stop splitting if the data sets become smaller than 3 points.

6 Neural Nets (25 points)

Consider a single artificial neural net unit in which we replace the sigmoid function with a quadratic function $y = z^2$, where z is the activation of the unit, the weighted sum of the inputs.



A

1. (10 pts) Give the (on-line) gradient descent learning rule for weight w_1 of this unit. Assume the training example is $x^i = [x_1^i, x_2^i]; y^i$. Your formula should involve this training example, the unit output y, the unit activation z and the step size η.

   The error (for a single training point) is $E = \frac{1}{2}(y - y^i)^2$.

   The gradient of the error is $\nabla_w E = (y - y^i)[\frac{\partial y}{\partial w_1}, \frac{\partial y}{\partial w_2}]$.

   The update rule is $w = w - \eta \nabla_w E$, *so*

$$w_1 = w_1 - \eta(y - y^i)\frac{\partial y}{\partial w_1}$$

   and, since $y = z^2 = (w_1 x_1^i + w_2 x_2^i - w_0)^2$, *then*

$$\frac{\partial y}{\partial w_1} = 2(w_1 x_1^i + w_2 x_2^i - w_0)x_1^i = 2zx_1^i$$

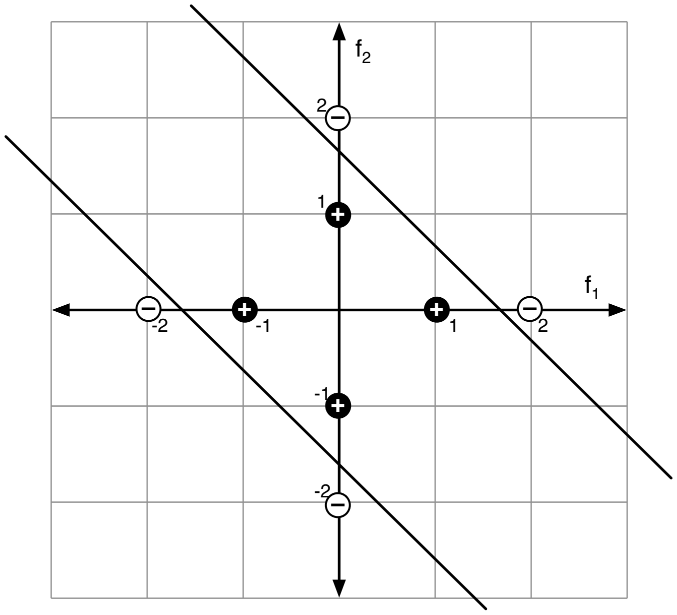
   finally

$$w_1 = w_1 - 2\eta(y - y^i)zx_1^i$$

2. (5 pts) Give a rule for mapping the output of unit (y) into two class labels: 0 and 1. There are many such rules possible, feel free to pick one that simplifies your answer to the next part of this question.

Predict 1 *if* $y \leq 3$ *and predict* 0 *otherwise.*

3. (10 pts) Pick weights for this unit so that it can accurately classify the data points below. Also draw (approximately) the decision boundaries on the figure.



$w_0 = 0$
$w_1 = 1$
$w_2 = 1$

The decision boundaries can be $y = z^2 = (x_1 + x_2)^2 = 3$. *Many other possibilities work as well, but this is probably simplest.*