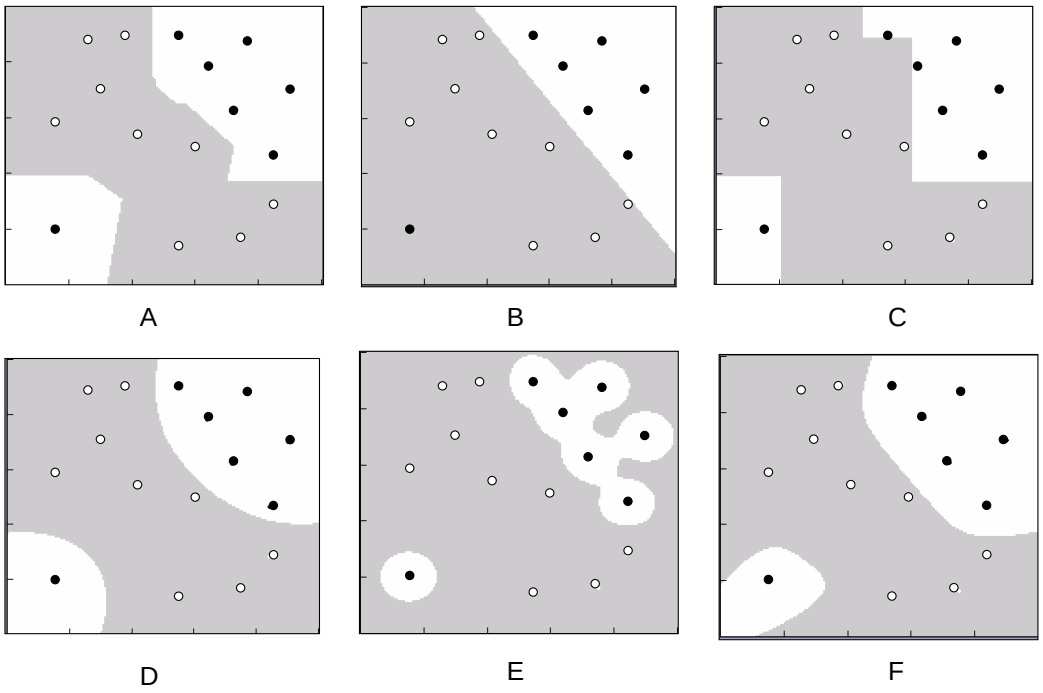
5 Learning hypothesis classes (16 points)

Consider a classification problem with two real-valued inputs. For each of the following algorithms, specify all of the separators below that it could have generated and explain why. If it could not have generated any of the separators, explain why not.



1. 1-nearest neighbor

   A — this is a Voronoi partition of the space, which describes the decision boundary produced by the 1-nearest neighbor algorithm

2. decision trees on real-valued inputs

   C — decision trees on real-valued inputs create a decision boundary that is made up of rectangles in the input space

3. standard perceptron algorithm

   none — the inputs are not linearly separable, and the standard perceptron algorithm does not terminate until it finds a linear separator that correctly classifies all of the training data

4. SVM with linear kernel

   B — the SVM algorithm will find some separator in the space that maximizes the margin, even if the data are not linearly separable

5. SVM with Gaussian kernel ($\sigma = 0.25$)

   E — a small sigma results in a classifier that more tightly fits the training data because the Gaussian bumps at each point are narrower

6. SVM with Gaussian kernel ($\sigma = 1$)

   D (or F) — a larger sigma results in a classifier that generalizes better because the Gaussian bumps at each point are wider. D is the separator actually generated by an SVM with Gaussian kernel, $\sigma = 1$, but we accepted F because it is difficult to tell which of these two would be generated without actually running the algorithm.

7. neural network with no hidden units and one sigmoidal output unit, run until convergence of training error

   B — a neural network with a single sigmoidal output will generate a linear classifier. The difference between a neural net with a single sigmoidal output and a perceptron unit is that the neural net training algorithm terminates when the error on a validation set reaches a minimum.

8. neural network with 4 hidden units and one sigmoidal output unit, run until convergence of training error

   F (and/or D) — a neural net with hidden units, run until convergence, can correctly classify all of the data (ruling out B). Also, the decision boundary will be smooth (why?) (ruling out A and C). Why not E?

6 Perceptron (8 points)

The following table shows a data set and the number of times each point is misclassified during a run of the perceptron algorithm, starting with zero weights. What is the equation of the separating line found by the algorithm, as a function of x_1, x_2, and x_3? Assume that the learning rate is 1 and the initial weights are all zero.

x_1	x_2	x_3	y	times misclassified
2	3	1	+1	12
2	4	0	+1	0
3	1	1	-1	3
1	1	0	-1	6
1	2	1	-1	11

$$\begin{aligned}\bar{w} &= \eta \sum_{i=1}^{m} \alpha_i y^i \bar{x}^i \\ &= (12)(1)(1,2,3,1) + (3)(-1)(1,3,1,1) + (6)(-1)(1,1,1,0) + (11)(-1)(1,1,2,1) \\ &= (-8,-2,5,-2)\end{aligned}$$

So the equation of the separating line is

$$-2x_1 + 5x_2 - 2x_3 - 8 = 0$$

7 SVMs (12 points)

Assume that we are using an SVM with a **polynomial kernel of degree 2**. You are given the following support vectors:

x_1	x_2	y
-1	2	+1
1	2	-1

The α values for each of these support vectors are equal to 0.05.

1. What is the value of b? Explain your approach to getting the answer.
   Answer: 0
2. What value does this SVM compute for the input point $(1, 3)$
   Answer: 0.05(1+(1,3).(-1,2))$^2 - 0.05(1 + (1, 3).(1, 2))^2] = 0.05[36 - 64] = -1.4$

8 Neural networks (18 points)

A physician wants to use a neural network to predict whether patients have a disease, based on the results of a battery of tests. He has assigned a cost of c_{01} to false positives (generating an output of 1 when it ought to have been 0), and a cost of c_{10} to generating an output of 0 when it ought to have been 1. The cost of a correct answer is 0.

The neural network is just a single sigmoid unit, which computes the following function:

$$g(\bar{x}) = s(\bar{w} \cdot \bar{x})$$

with $s(z)$ being the usual sigmoid function.

1. Give an error function for the whole training set, $E(\bar{w})$ that implements this error metric, for example, for a training set of 20 cases, if the network predicts 1 for 5 cases that should have been 0, predicts 0 for 3 cases that should have been 1 and predicts another 12 correctly, the value of the error function should be: $5c_{01} + 3c_{10}$.

   Answer:
   $$E(\bar{w}) = c_{10} \sum_{\{i|y^i=1\}} I(y^i \neq g(\bar{x}^i)) + c_{01} \sum_{\{i|y^i=0\}} I(y^i \neq g(\bar{x}^i))$$

2. Would this be an appropriate error criterion to use for a neural network? Why or why not?

   Answer: No good. Not differentiable.

3. Consider the following error function for the whole training set:

   $$E(\bar{w}) = c_{10} \sum_{\{i|y^i=1\}} (g(\bar{x}^i) - y^i)^2 + c_{01} \sum_{\{i|y^i=0\}} (g(\bar{x}^i) - y^i)^2$$

   Describe, in English that is not simply a direct paraphrase of the mathematics, what it measures.

   Answer: Weighted square magnitude of prediction error.

4. What is the gradient of this E with respect to $\bar{w}$?

   Answer:

   $$2c_{10} \sum_{\{i|y^i=1\}} (g(\bar{x}^i) - y^i)(ds/dz_i)\bar{x} + 2c_{01} \sum_{\{i|y^i=0\}} (g(\bar{x}^i) - y^i)(ds/dz_i)\bar{x}$$

   where

   $$ds/dz_i = g(\bar{x}^i)(1 - g(\bar{x}^i))$$

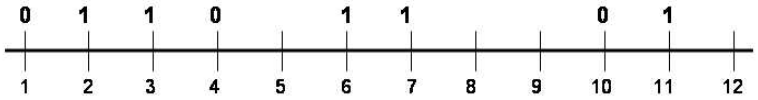
5. Give a complete algorithm for training a single sigmoid unit using this error function.
   Answer:
   (a) *Pick a random set of small weights* $\bar{w}$
   (b) $\bar{w} = \bar{w} - \eta * Gradient$ *for some small* r
   (c) *until* $\bar{w}$ *stops changing*

4 Machine Learning — Continuous Features (20 points)

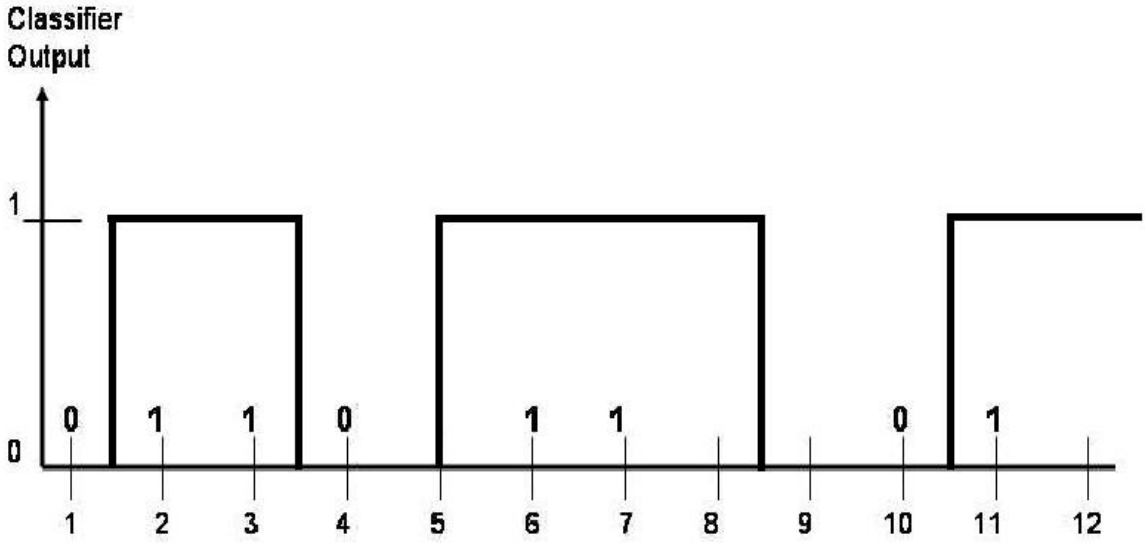
In all the parts of this problem we will be dealing with one-dimensional data, that is, a set of points (x^i) with only one feature (called simply x). The points are in two classes given by the value of y^i. We will show you the points on the x axis, labeled by their class values; we also give you a table of values.

4.1 Nearest Neighbors

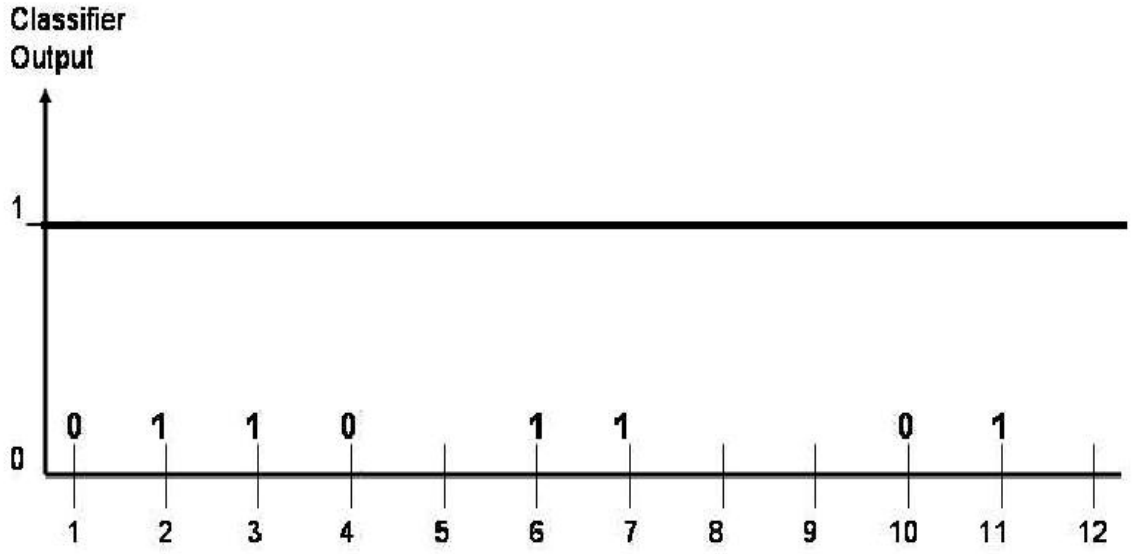
i	x^i	y^i
1	1	0
2	2	1
3	3	1
4	4	0
5	6	1
6	7	1
7	10	0
8	11	1



1. In the figure below, draw the output of a 1-Nearest-Neighbor classifier over the range indicated in the figure.

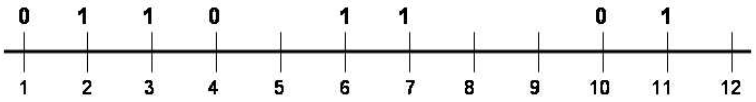


2. In the figure below, draw the output of a 5-Nearest-Neighbor classifier over the range indicated in the figure.



4.2 Decision Trees

Answer this problem using the same data as in the Nearest Neighbor problem above.



Which of the following three tests would be chosen as the top node in a decision tree?

$$x \leq 1.5 \qquad x \leq 5 \qquad x \leq 10.5$$

Justify your answer.

Recall that entropy for each side of a split is:

$$H = -p \log p - (1-p) \log(1-p)$$

So, for $x \leq 1.5$ *we have:*

$$\begin{aligned} H &= \frac{1(0) + 7\left(-\frac{5}{7}\log_2(\frac{5}{7}) - \frac{2}{7}\log_2(\frac{2}{7})\right)}{8} \\ H &= \frac{7(0.35 + 0.52)}{8} \\ H &= 0.761 \end{aligned}$$

while $x \leq 5$

$$\begin{aligned} H &= \frac{4\left(-\frac{2}{4}\log_2(\frac{2}{4}) - \frac{2}{4}\log_2(\frac{2}{4})\right) + 4\left(-\frac{3}{4}\log_2(\frac{3}{4}) - \frac{1}{4}\log_2(\frac{1}{4})\right)}{8} \\ H &= \frac{4(0.5 + 0.5) + 4(0.31 + 0.50)}{8} \\ H &= 0.905 \end{aligned}$$

and $x \leq 10.5$ *gives us:*

$$\begin{aligned} H &= \frac{1(0) + 7\left(-\frac{4}{7}\log_2(\frac{4}{7}) - \frac{3}{7}\log_2(\frac{3}{7})\right)}{8} \\ H &= \frac{7(0.46 + 0.52)}{8} \\ H &= 0.85 \end{aligned}$$

So, we choose the split with the least average entropy, which is $x \leq 1.5$.

You may find this table useful.

x	y	-(x/y)*lg(x/y)		x	y	-(x/y)*lg(x/y)
1	2	0.50		1	8	0.38
1	3	0.53		3	8	0.53
2	3	0.39		5	8	0.42
1	4	0.50		7	8	0.17
3	4	0.31		1	9	0.35
1	5	0.46		2	9	0.48
2	5	0.53		4	9	0.52
3	5	0.44		5	9	0.47
4	5	0.26		7	9	0.28
1	6	0.43		8	9	0.15
2	6	0.53		1	10	0.33
5	6	0.22		3	10	0.52
1	7	0.40		7	10	0.36
2	7	0.52		9	10	0.14
3	7	0.52				
4	7	0.46				
5	7	0.35				
6	7	0.19				

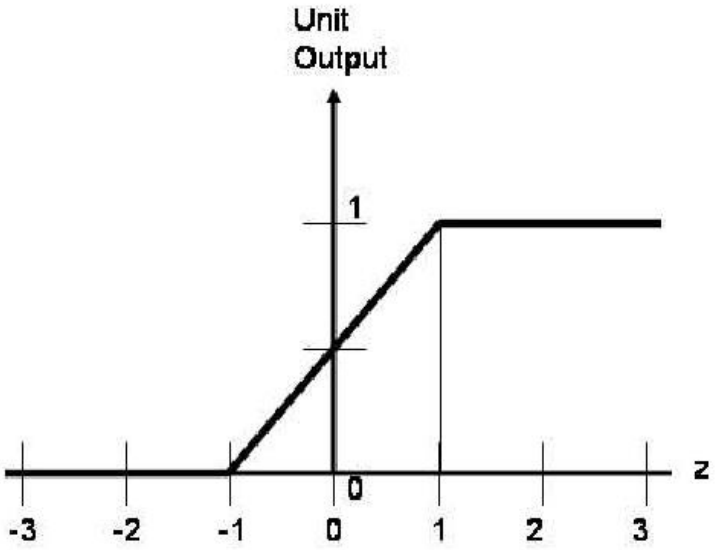
4.3 Neural Nets

Assume that each of the units of a neural net uses one of the the following output functions of the total activation (instead of the usual sigmoid $s(z)$)

- **Linear:** This just outputs the total activation:

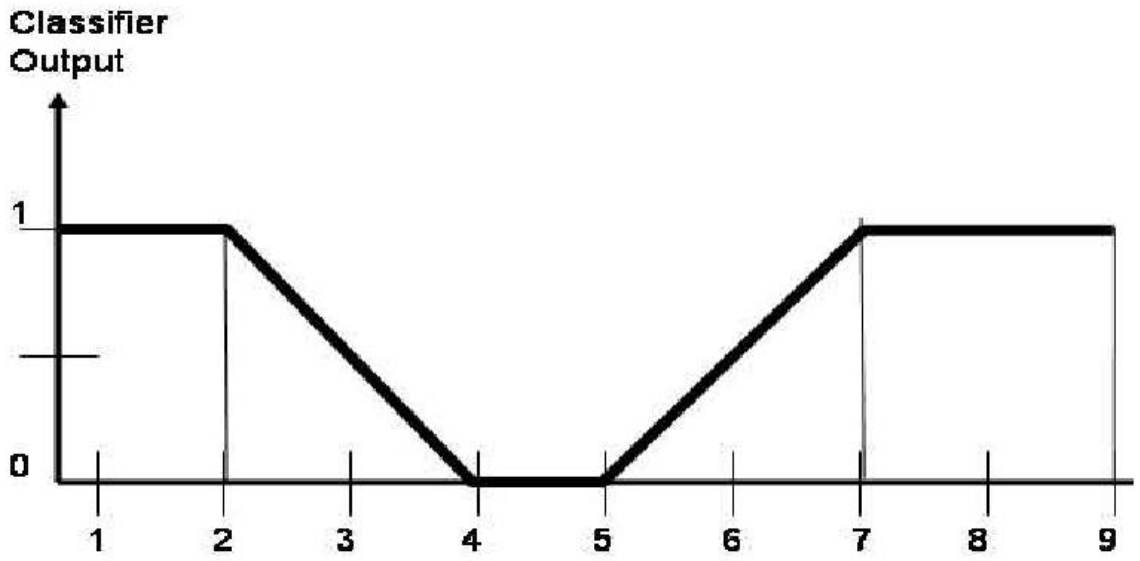
$$l(z) = z$$

- **Non-Linear:** This looks like a linearized form of the usual sigmoid funtion:



$$\begin{aligned} f(z) &= 0 \quad \text{if } z < -1 \\ f(z) &= 1 \quad \text{if } z > 1 \\ f(z) &= 0.5(z+1) \quad \text{otherwise} \end{aligned}$$

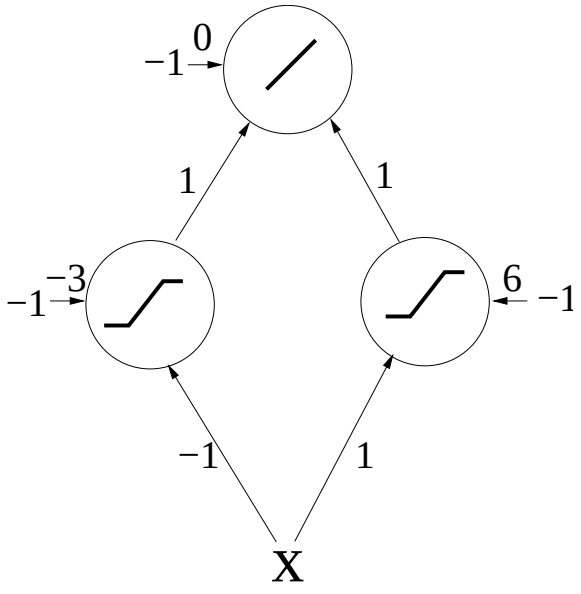
Consider the following output from a neural net made up of units of the types described above.



1. Can this output be produced using only linear units? Explain.

   No. Linear units can only generate a linear separator.

2. Construct the simplest neural net out of these two type of units that would have the output shown above. When possible, use weights that have magnitude of 1. Label each unit as either Linear or Non-Linear.



4.4 SVM

What are the values for the α_i and the offset b that would give the maximal margin linear classifier for the two data points shown below? You should be able to find the answer without deriving it from the dual Lagrangian.

i	x^i	y^i
1	0	1
2	4	-1

We know that the $w = \sum_i \alpha_i x^i y^i$. *Thus:*

$$\begin{aligned} w &= \alpha_1 x^1 y^1 + \alpha_2 x^2 y^2 \\ w &= \alpha_1(0)(1) + \alpha_2(4)(-1) \\ w &= -4\alpha_2 \end{aligned}$$

We know further that $\sum_i y^i \alpha_i = 0$, *so the alphas must be equal. Lastly, we know that the margin for the support vectors is 1, so* $wx_1 + b = 1$, *which tells us that* $b = 1$, *and* $wx_2 + b = -1$, *which tells us that* $w = -0.5$. *Thus we know that* $\alpha_1 = \alpha_2 = \frac{1}{8}$.

5 Machine Learning (20 points)

Grady Ent decides to train a single sigmoid unit using the following error function:

$$E(\mathbf{w}) = \frac{1}{2}\sum_i (y(\mathbf{x}^i, \mathbf{w}) - y^{i*})^2 + \frac{1}{2}\beta \sum_j w_j^2$$

where $y(\mathbf{x}^i, \mathbf{w}) = s(\mathbf{x}^i \cdot \mathbf{w})$ with $s(z) = \frac{1}{1+e^{-z}}$ being our usual sigmoid function.

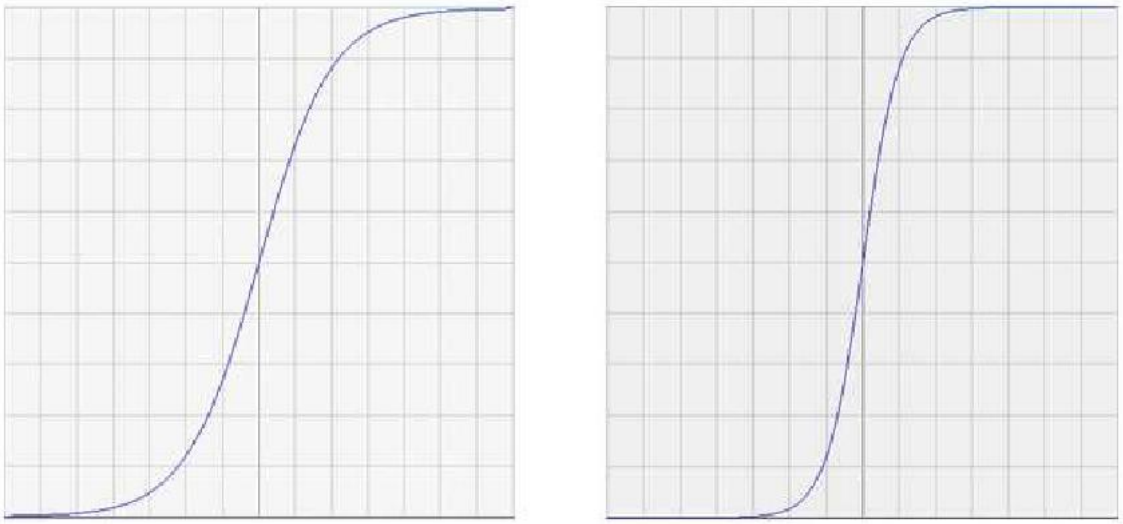
1. Write an expression for $\frac{\partial E}{\partial w_j}$. Your answer should not involve derivatives.

$$\begin{aligned}
\frac{\partial E}{\partial w_j} &= \frac{\partial}{\partial w_j}\left(\frac{1}{2}\sum_i (y(x^i, w) - y^{i*})^2\right) + \frac{\partial}{\partial w_j}\left(\frac{1}{2}\beta\sum_j w_j^2\right) \\
&= \frac{\partial E}{\partial y}\frac{\partial y}{\partial z}\frac{\partial z}{\partial w_j} + \beta w_j \\
&= \sum_{i=i} (y - y^{i*})(y)(1-y)(x_j^i) + \beta w_j
\end{aligned}$$

2. What update should be made to weight w_j given a single training example $< \mathbf{x}, y^* >$. Your answer should not involve derivatives.

   $w_j' = w_j - \eta((y - y^*)(y)(1-y)(x_j) + (\beta w_j))$

3. Here are two graphs of the output of the sigmoid unit as a function of a single feature x. The unit has a weight for x and an offset. The two graphs are made using different values of the magnitude of the weight vector ($\|\mathbf{w}\|^2 = \sum_j w_j^2$).



A B

Which of the graphs is produced by the larger $\|\mathbf{w}\|^2$? Explain.

Graph B because for a given input x, the larger the magnitude of the weight vector, the greater the change in the output of the sigmoid relative to that x.

4. Why might penalizing large $\|\mathbf{w}\|^2$, as we could do above by choosing a positive β, be desirable?

Penalizing large weight vectors would be desirable so that we avoid saturation. If the magnitude of the weight vector is too large, gradient descent will not work because the derivative of the sigmoid for large values is nearly 0.

Also, large weights can cause overfitting, since they let you draw boundaries closer to the points. As we saw with SVMs, being able to get closer to the points actually increases the size of the hypothesis class.

5. How might Grady select a good value for β for a particular classification problem?

Grady could use cross-validation for several values of β and use the value which yields the best cross-validation accuracy.

6 Pruning Trees (20 points)

Following are some different strategies for pruning decision trees. We assume that we grow the decision tree until there is one or a small number of elements in each leaf. Then, we prune by deleting individual leaves of the tree until the score of the tree starts to get worse. The question is how to score each possible pruning of the tree.

For each possible definition of the score below, explain whether or not it would be a good idea and give a reason why or why not.

1. The score is the percentage correct of the tree on the training set.

   Not a good idea. The original tree was constructed to maximize performance on the training set. Pruning any part of the tree will reduce performance on the training set.

2. The score is the percentage correct of the tree on a separate validation set.

   A good idea. The validation set will be an independent check on whether pruning a node is likely to increase or decrease performance on unseen data.

3. The score is the percentage correct of the tree, computed using cross validation.

   Not a good idea. Cross-validation allows you to evaluate algorithms, not individual hypotheses. Cross-validation will construct many new hypotheses and average their performance, this will not tell you whether pruning a node in a particular hypothesis is worthwhile or not.

4. The score is the percentage correct of the tree, computed on the training set, minus a constant C times the number of nodes in the tree.

   C is chosen in advance by running this algorithm (grow a large tree then prune in order to maximize percent correct minus C times number of nodes) for many different values of C, and choosing the value of C that minimizes training-set error.

   Not a good idea. Running trials to maximize performance on the training set will not give us an indication of whether this algorithm will produce answers that generalize to other data sets.

5. The score is the percentage correct of the tree, computed on the training set, minus a constant C times the number of nodes in the tree.

   C is chosen in advance by running cross-validation trials of this algorithm (grow a large tree then prune in order to maximize percent correct minus C times number of nodes) for many different values of C, and choosing the value of C that minimizes cross-validation error.

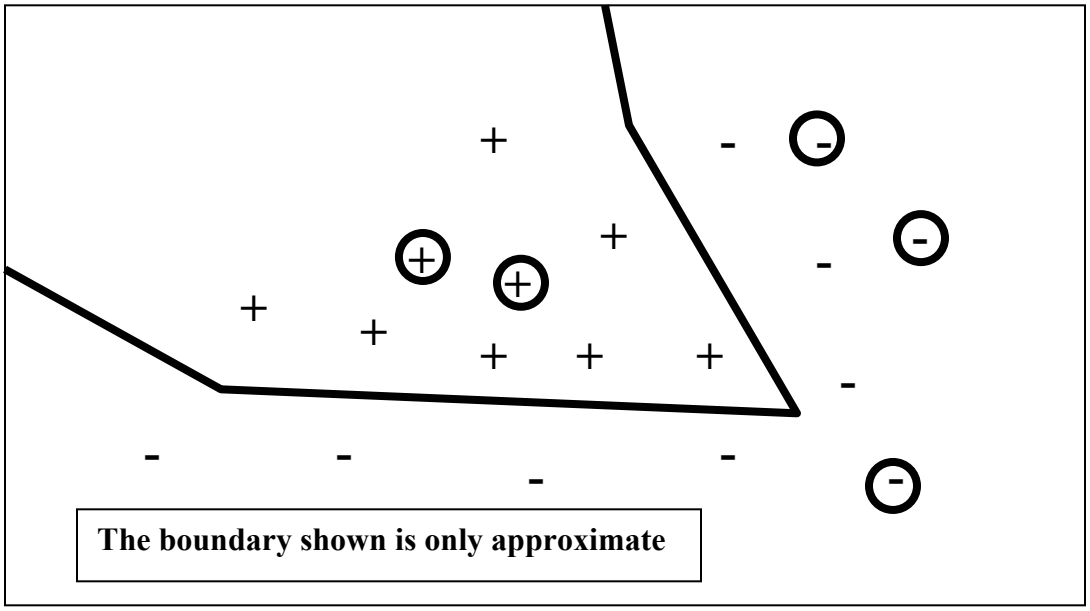
   A good idea when we don't have enough data to hold out a validation set. Choosing C by cross-validation will hopefully give us an effective general way of penalizing for complexity of the tree (for this type of data).

Problem 4: Learning (25 points)

Part A: (5 Points)

Since the cost of using a nearest neighbor classifier grows with the size of the training set, sometimes one tries to eliminate redundant points from the training set. These are points whose removal does not affect the behavior of the classifier for any possible new point.

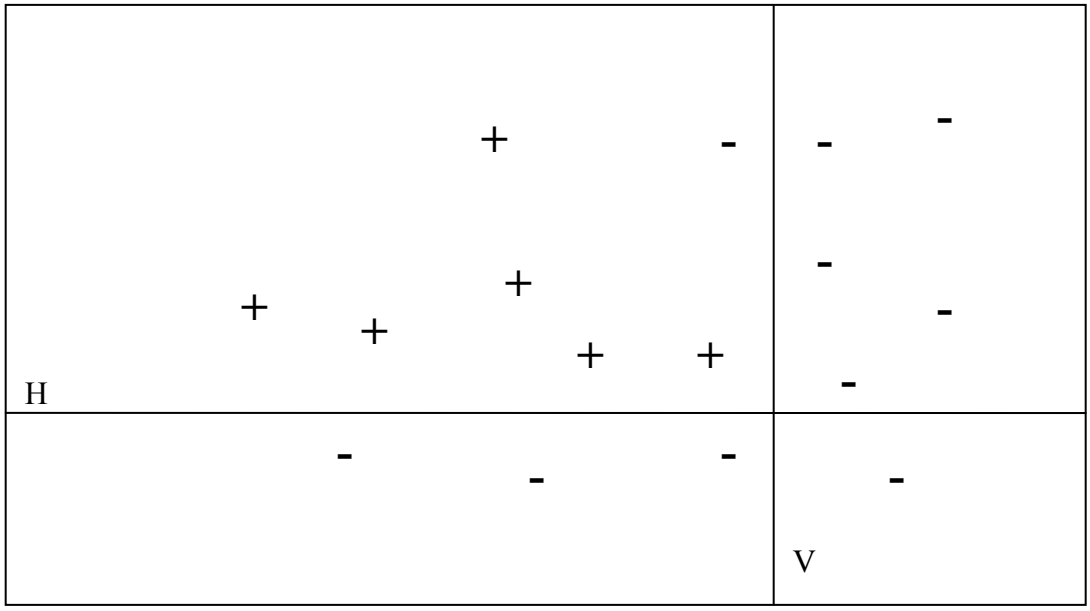
1. In the figure below, sketch the decision boundary for a 1-nearest-neighbor rule and circle the redundant points.



2. What is the general condition(s) required for a point to be declared redundant for a 1-nearest-neighor rule? Assume we have only two classes (+, -). Restating the definition of redundant ("removing it does not change anything") is not an acceptable answer. Hint – think about the neighborhood of redundant points.

Let the Voronoi cell for a training point be the set of points that are closest to that point (as opposed to some other training point). The Voronoi cell of a redundant point touches only on other Voronoi cells of points of the same class.

Part B: (5 Points)



Which of H or V would be preferred as an initial split for a decision (identification) tree? Justify your answer numerically.

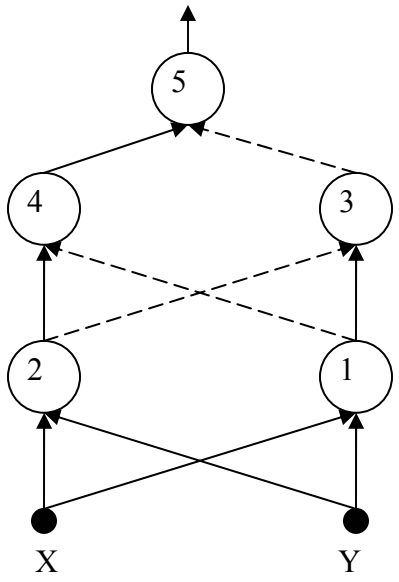
V = 0.60625
H = 0.75

So, V is chosen

x	y	-(x/y)*lg(x/y)
1	2	0.50
1	3	0.53
2	3	0.39
1	4	0.50
3	4	0.31
1	5	0.46
2	5	0.53
3	5	0.44
4	5	0.26
1	6	0.43
2	6	0.53
5	6	0.22
1	7	0.40
2	7	0.52
3	7	0.52
4	7	0.46
5	7	0.35
6	7	0.19

x	y	-(x/y)*lg(x/y)
1	8	0.38
3	8	0.53
5	8	0.42
7	8	0.17
1	9	0.35
2	9	0.48
4	9	0.52
5	9	0.47
7	9	0.28
8	9	0.15
1	10	0.33
3	10	0.52
7	10	0.36
9	10	0.14

Part C: (10 Points)



In this network, **all the units are sigmoid except unit 5 which is linear** (its output is simply the weighted sum of its inputs). All the bias weights are zero. The dashed connections have weights of -1, all the other connections (solid lines) have weights of 1.

1. Given X=0 and Y=0, what are the output values of each of the units?

Unit 1 = 0.5
Unit 2 = 0.5
Unit 3 = 0.5
Unit 4 = 0.5
Unit 5 = 0

2. What are the δ values for each unit (as computed by backpropagation defined for squared error) assume that the desired output for the network is 4.

Unit 1 = 0.5
Unit 2 = -0.5
Unit 3 = 1
Unit 4 = -1
Unit 5 = -4

3. What would be the new value of the weight connecting units 2 and 3 assuming that the learning rate for backpropagation is set to 1?

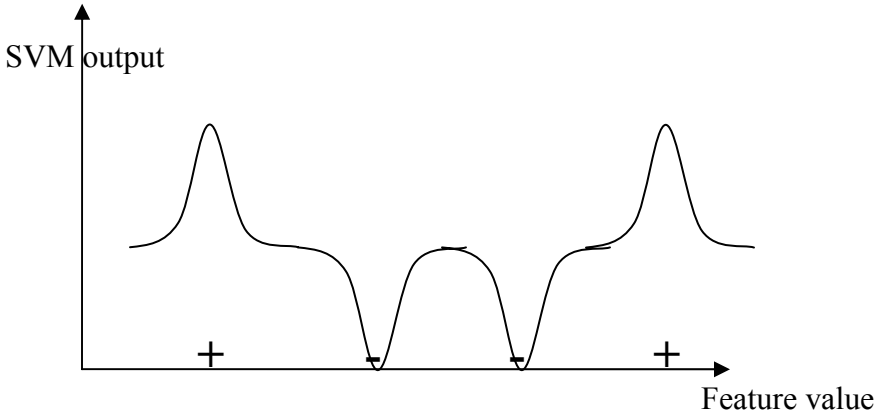
$$w_{2,3} = (-1) - (1)(1)(0.5) = -1.5$$

Part D: (10 Points)

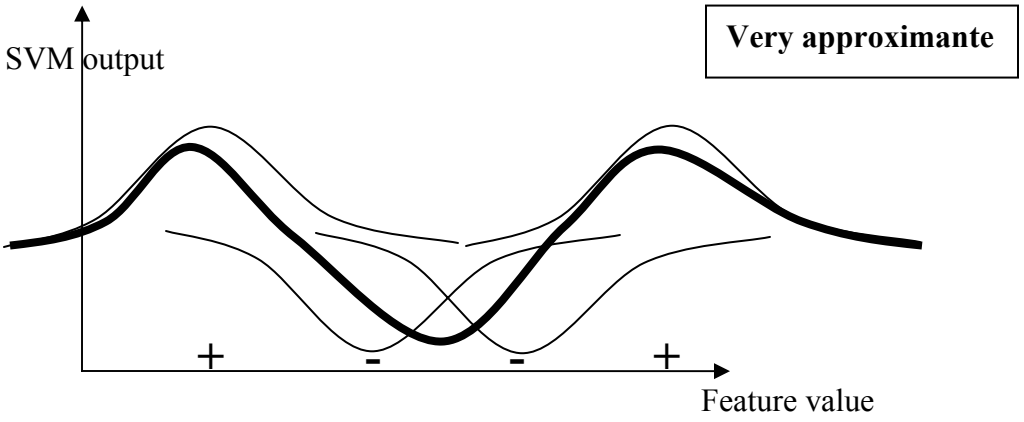
1. Consider the simple one-dimensional classification problem shown below. Imagine attacking this problem with an SVM using a radial-basis function kernel. Assume that we want the classifier to return a positive output for the + points and a negative output for the – points.

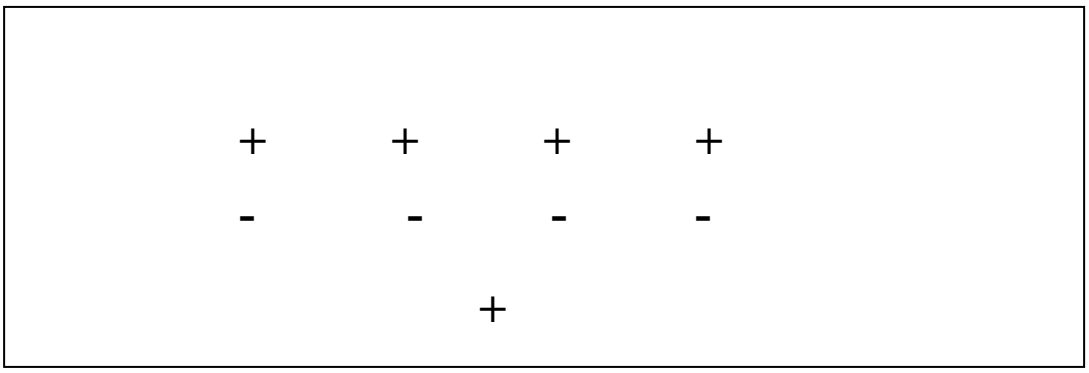
   Draw a plausible classifier output curve for a trained SVM, indicating the classifier output for every feature value in the range shown. Do this twice, once assuming that the standard deviation (σ) is very small relative to the distance between adjacent training points and again assuming that the standard deviation (σ) is about double the distance between adjacent training points.

   Small standard deviation (σ):



   Large standard deviation (σ):





2. Would you expect that a polynomial kernel with d=1 would be successful in carrying out the classification shown above? Explain.

   No, a first degree polynomial is a line and this is not linearly separable.

3. Assume we use an SVM with a radial-basis function kernel to classify these same data points. We repeat the classification for different values of the the standard deviation (σ) used in the kernel. What would you expect to be the relationship between the standard deviation used in the kernel and the value of the largest Lagrange multiplier (a_i) needed to carry out the classification? That is, would you expect that the max a_i would increase or decrease as the standard deviation decreases? Explain your answer.

   As the standard deviation decreases, we expect the max Lagrange multiplier to decrease. When the Gaussians overlap a lot, the multiplier for the isolated + has to be quite big to "raise" the classifier output to positive. When the Gaussians don't overlap, the multiplier will not need to be large.

Part E: (5 Points)

Given a validation set (a set of samples which is separate from the training set), explain how it should be used in connection with training different learning functions (be specific about the problems that are being addressed):

1. For a neural net

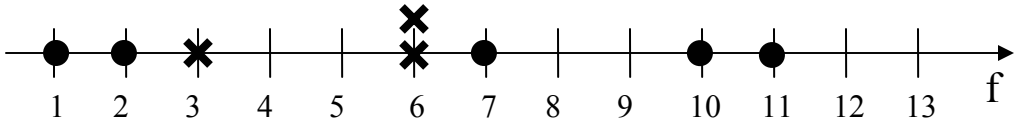
Use the validation set to:

   a. *choose the number of units in the network*
   b. *decide when to stop backpropagation*

2. For a decision (identification) tree

Use the validation set for pruning the tree – that is, drop tests that do not improve the performance on validation set.

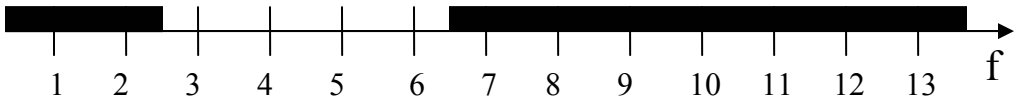
Problem 1: Classification (40 points)



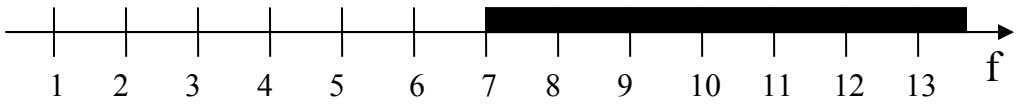
The picture above shows a data set with 8 data points, each with only one feature value, labeled f. Note that there are two data points with the same feature value of 6. These are shown as two X's one above the other, but they really should have been drawn as two X's on top of each other, since they have the same feature value.

Part A: (10 Points)

1. Consider using 1-Nearest Neighbors to classify unseen data points. On the line below, darken the segments of the line where the 1-NN rule would predict an O given the training data shown in the figure above.



2. Consider using 5-Nearest Neighbors to classify unseen data points. On the line below, darken the segments of the line where the 5-NN rule would predict an O given the training data shown in the figure above.

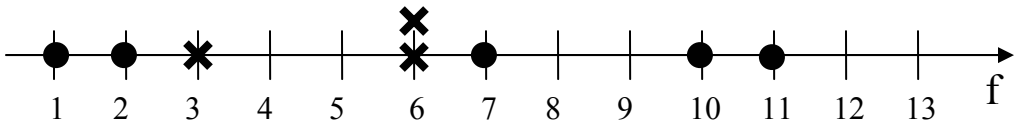


3. If we do 8-fold cross-validation using 1-NN on this data set, what would be the predicted performance? Settle ties by choosing the point on the left. Show how you arrived at your answer.

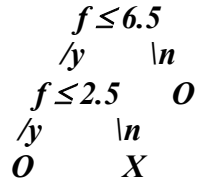
The point at 1 would be correct, nearest neighbor is at 2
The point at 2 would be correct, nearest neighbor is at 1 (tie)
The point at 3 would be incorrect, nearest neighbor is 2
Both points at 6 would be correct, nearest neighbor is the other point at 6.
The point at 7 would be incorrect, nearest neighbor is 6 (tie)
The point at 10 would be correct, nearest neighbor is 11
The point at 11 would be correct, nearest neighbor is 10

So,6 correct, 2 incorrect => 75% would be predicted performance

Part B: (8 Points)



Using this same data set, show the decision tree that would be built from this data. Assume that the tests in the tree are of the form f ≤ c. For each test show the approximate value of the average disorder for that test. To help you compute this, there's a small table of values of –(x/y)*log(x/y) for small integer x and y.

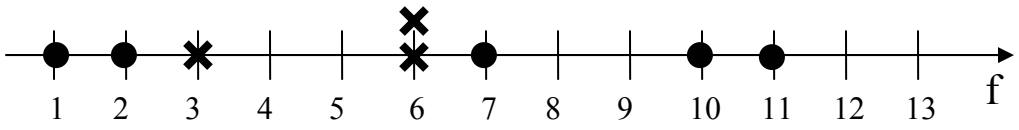


The top node as average disorder = 5/8[-2/5lg(2/5)-3/5lg(3/5)] + 3/8*0 = 0.61
The other decision node has 0 average disorder.

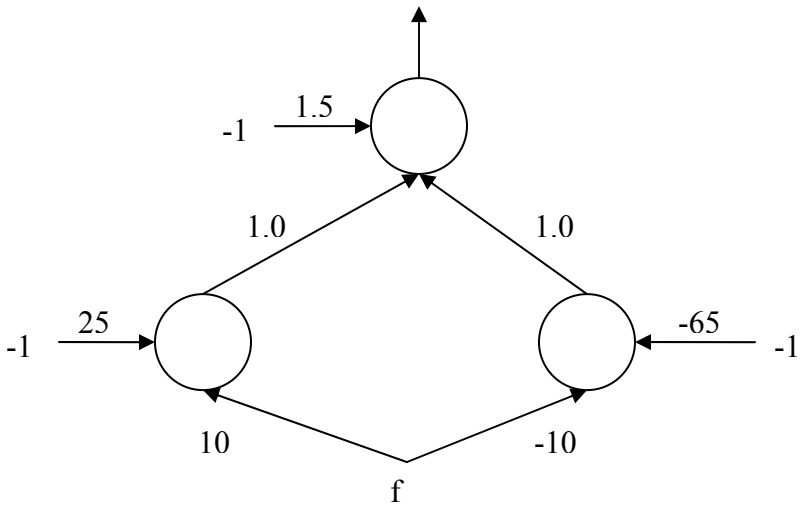
x	y	-(x/y)*lg(x/y)
1	2	0.50
1	3	0.53
2	3	0.39
1	4	0.50
3	4	0.31
1	5	0.46
2	5	0.53
3	5	0.44
4	5	0.26
1	6	0.43
2	6	0.53
5	6	0.22
1	7	0.40
2	7	0.52
3	7	0.52
4	7	0.46
5	7	0.35
6	7	0.19

x	y	-(x/y)*lg(x/y)
1	8	0.38
3	8	0.53
5	8	0.42
7	8	0.17
1	9	0.35
2	9	0.48
4	9	0.52
5	9	0.47
7	9	0.28
8	9	0.15
1	10	0.33
3	10	0.52
7	10	0.36
9	10	0.14

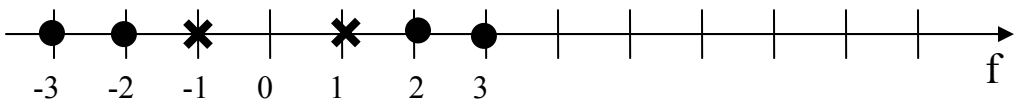
Part C: (12 Points)



Construct the **simplest** neural net (using sigmoid units) that can accurately classify this data. Pick a set of appropriate weights for the network. Assume that we will predict O when the network's output is less than 0.5 and X when the output is above 0.5.
Whenever possible use weights that are either 1, -1, 10 or -10.



Part D: (10 Points)



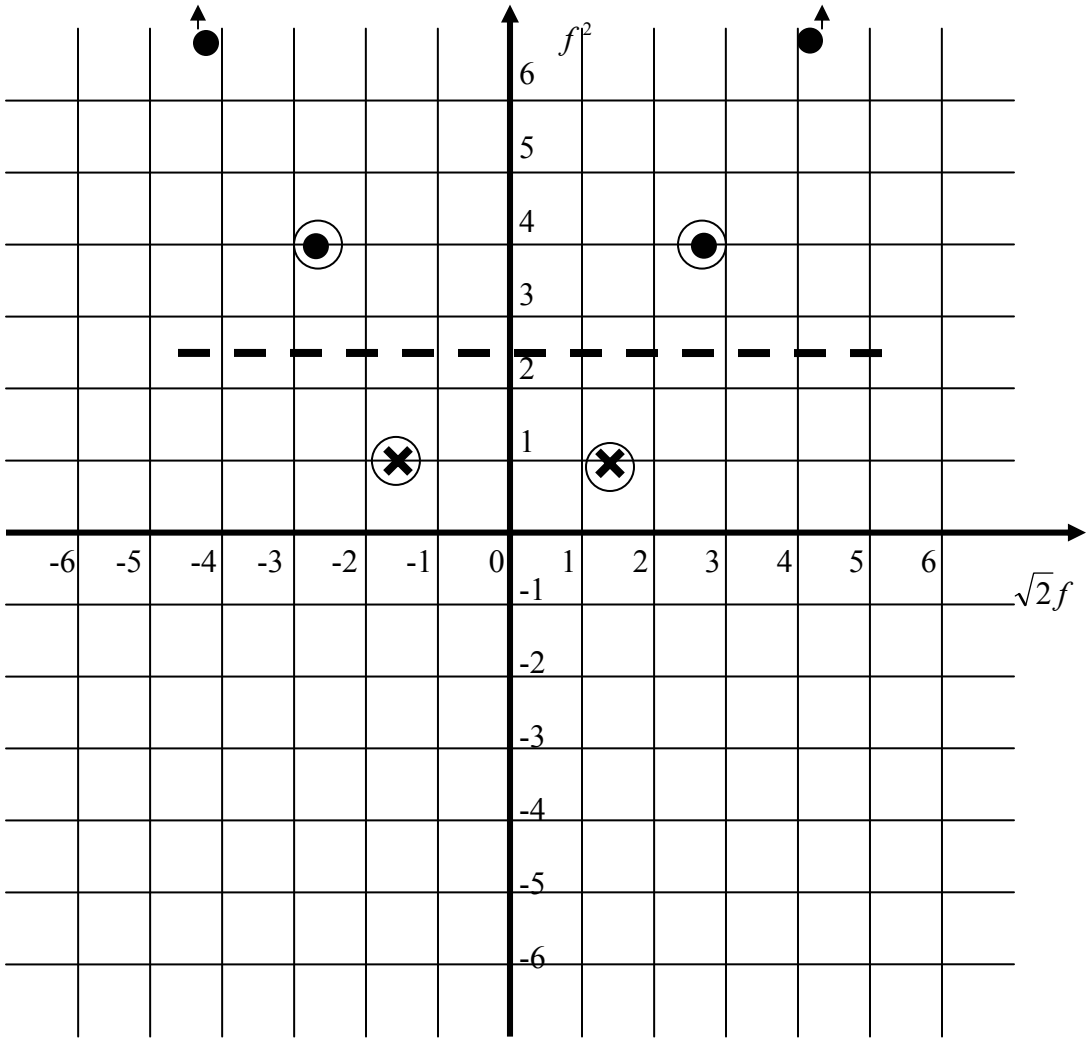
Consider the simplified data set above and consider using an SVM with a polynomial kernel with d=2. Let's say the data points are specified as:

X1 = [-2] Y1= -1
X2 = [-1] Y2= 1
X3 = [1] Y3= 1
X4 = [2] Y4= -1
X5=[-3] Y5= -1
X6=[3] Y6= -1

1. What are the kernel values?

K(x1,x1)	***25***
K(x1,x2)	***9***
K(x2,x3)	***0***
K(x3,x4)	***9***

2. Show a reasonably accurate picture of the **transformed** feature space.
   a. label the axes,
   b. label the data points,
   c. show the separating line that would be found by the SVM,
   d. circle the support vectors.



Problem 2: Overfitting (20 points)

For each of the supervised learning methods that we have studied, indicate how the method could overfit the training data (consider both your design choices as well as the training) and what you can do to minimize this possibility. There may be more than one mechanism for overfitting, make sure that you identify them all.

Part A: Nearest Neighbors (5 Points)

1. How does it overfit?
   Every point in dataset (including noise) defines its own decision boundary.
   The distance function can be chosen to do well on training set but less well on new data.

2. How can you reduce overfitting?
   Use k-NN for larger k
   Use cross-validation to choose k and the distance function

Part B: Decision Trees (5 Points)

1. How does it overfit?
   By adding new tests to the tree to correctly classify every data point in the training set.

2. How can you reduce overfitting?
   By pruning the resulting tree based on performance on a validation set.

Part C: Neural Nets (5 Points)

1. How does it overfit?

   By having too many units and therefore too many weights, thus enabling it to fit every nuance of the training set.

   By training too long so as to fit the training data better.

2. How can you reduce overfitting?

   Using cross-validation to choose a not too complex network

   By using a validation set to decide when to stop training.

Part D: SVM [Radial Basis and Polynomial kernels] (5 Points)

1. How does it overfit?

   In RBF, by choosing a value of sigma (the std dev of Gaussian) too small.

   In Polynomial, by choosing the degree of the polynomial too high

   By allowing the Lagrange multipliers to get too large

2. How can you reduce overfitting?

   Using cross-validation to choose the kernel parameters and the maximum value for the multipliers.

Problem 3: Spaminator (10 points)

Suppose that you want to build a program that detects whether an incoming e-mail message is spam or not. You decide to attack this using machine learning. So, you collect a large number of training messages and label them as spam or not-spam. You further decide that you will use the presence of individual words in the body of the message as features. That is, you collect every word found in the training set and assign to each one an index, from 1 to N. Then, given a message, you construct a feature vector with N entries and write in each entry a number that indicates how many times the word appears in that message.

Part A: (6 Points)

If you had to choose between a Nearest Neighbor implementation or an Decision Tree implementation, which would you choose? Justify your answer briefly both in terms of expected accuracy and efficiency of operation. Indicate the strength and weaknesses of each approach.

Nearest Neighbors does not work well in high dimensions.
The biggest problem in using Nearest Neighbor would be choosing which of the features are relevant (which is related to choosing the distance metric). This is particularly severe in this application because of the huge numbers of probably irrelevant features (words).
The ID tree approach would spend initial effort in choosing which words were relevant to making the decision.
Nearest neighbor is also very expensive during classification for high dimensional feature vectors. ID trees would be much more efficient.
So, ID trees would be clearly better choice on all criteria.
However, Naïve Bayes would probably be better than either of them.

Part B: (4 Points)

Assume that you wanted to reduce the size of the feature vectors (during training and classification), for each of the approaches below indicate why it might be a good or bad idea.

1. Use only the words that appear in spam messages.

   This would be a bad idea. There may be words that are common in spam and in non-spam. If you trained only with the spam words, you would end up deciding that many non-spam messages were spam.

2. Eliminate words that are very common in the whole data set.

This is generally a good idea. A count of "and", "is", "of", "the", etc. is not going to be very useful in differentiating spam from non-spam.

Problem 5: Backprop (10 points)

Suppose we want to do regression instead of classification and so we change the (final) output unit of a neural net to be a linear unit, which simply outputs the weighted sum of its inputs (no sigmoid):

$$y = \sum_i w_i x_i$$

All the other units in the network would still retain their sigmoids.

How would this change the backpropagation equations? Derive only the needed changes, do not repeat anything in the usual derivation that does not change.

The only thing that changes is the initial computation of the error term for the output layer. Instead of:

$$y_n(1 - y_n)(y_n - y_n^*)$$

we have just:

$$y_n - y_n^*$$

since the first two terms in the original expression were the derivative of the output sigmoid.