Last name (CAPTIALS): ___________________________________

First name (CAPITALS): __________________________________

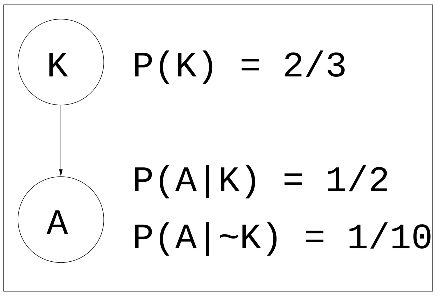
Andrew User ID (CAPITALS): (without the @andrew.cmu.edu bit): ___________________________

15-781 Final Exam, Fall 2001

- You must answer any nine questions out of the following twelve. Each question is worth 11 points.
- You must fill out your name and your andrew userid clearly and in block capital letters on the front page. You will be awarded 1 point for doing this correctly.
- If you answer more than 9 questions, your best 9 scores will be used to derive your total.
- Unless the question asks for explanation, no explanation is required for any answer. But you are welcome to provide explanation if you wish.

1 Bayes Nets Inference

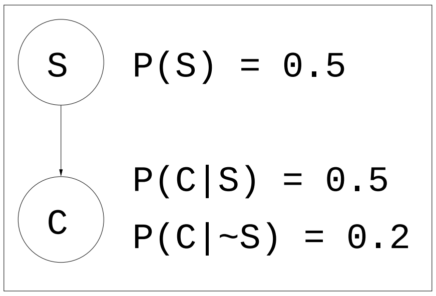
(a) Kangaroos.



Half of all kangaroos in the zoo are angry, and 2/3 of the zoo is comprised of kangaroos. Only 1 in 10 of the other animals are angry. What's the probability that a randomly-chosen animal is an angry kangaroo?

```
P(A ^ K) = P(K) P(A|K) = 2/3 * 1/2 = 1/3
```

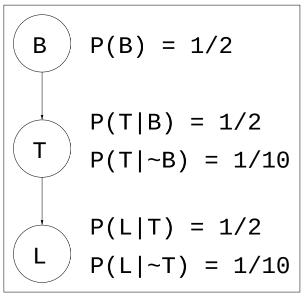
(b) Stupidity.



Half of all people are stupid. If you're stupid then you're more likely to be confused. A randomly-chosen person is confused. What's the chance they're stupid?

```
P(S|C) = P(S^C) / [P(S^C) + P(~S ^ C)] = 1/4 / (1/4 + 1/10) = 5/7
```

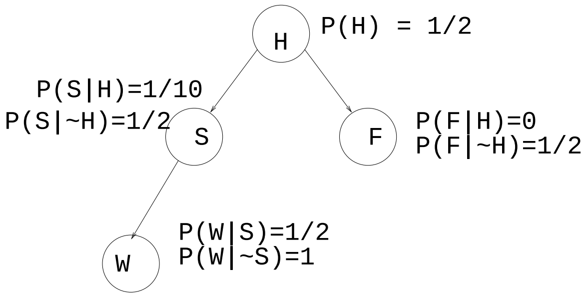
(c) Potatoes.



Half of all potatoes are big. A big potato is more likely to be tall. A tall potato is more likely to be lovable. What's the probability that a big lovable potato is tall?

```
P(T|B^L) = P(T^B^L) / [P(T^B^L + P(~T^B^L)] = 1/8 / (1/8 + 1/40) = 5/6
```

(d) **Final part.**



What's $P(W \wedge F)$?

```
P(W^F) = P(W^F^H) + P(W^F^~H)    but  P(W^F^H) = 0

P(W^F^~H) = P(F|~H)P(W^~H) =
               1/2 * (P(W ^ S ^ ~H) + P(W ^ ~S ^ ~H)) = 1/2 * ( 1/8 + 1/4) = 3/16
```

2 Bayes Nets and HMMs

(a) Let nbs(m) = the number of possible Bayes Network graph structures using m attributes. (Note that two networks with the same structure but different probabilities in their tables do not count as different structures). Which of the following statements is true?

- (i) $\text{nbs}(m) < m$
- (ii) $m \leq \text{nbs}(m) < \frac{m(m-1)}{2}$
- (iii) $\frac{m(m-1)}{2} \leq \text{nbs}(m) < 2^m$
- (iv) $2^m \leq \text{nbs}(m) < 2^{\frac{m(m-1)}{2}}$
- (v) $2^{\frac{m(m-1)}{2}} \leq \text{nbs}(m)$

```
Answer is (v) because the number of undirected graphs with n vertices
is 2^[m choose 2], and there are even more acyclic directed graphs
```

(b) Remember that $I < X, Y, Z >$ means

X is conditionally independent of Z given Y

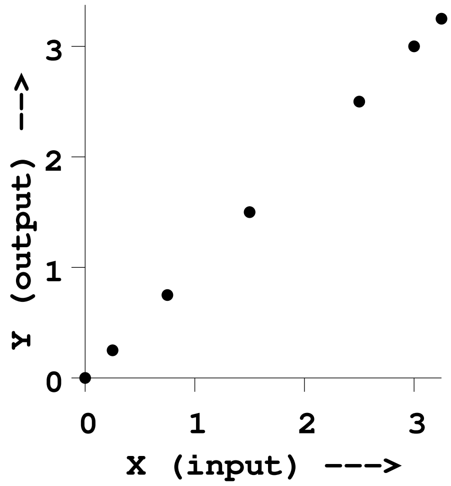
Assuming the conventional assumptions and notation of Hidden Markov Models, in which q_t denotes the hidden state at time t and O_t denotes the observation at time t, which of the following are true of all HMMs? Write "True" or "False" next to each statement.

- (i) $I < q_{t+1}, q_t, q_{t-1} >$
- (ii) $I < q_{t+2}, q_t, q_{t-1} >$
- (iii) $I < q_{t+1}, q_t, q_{t-2} >$
- (iv) $I < O_{t+1}, O_t, O_{t-1} >$
- (v) $I < O_{t+2}, O_t, O_{t-1} >$
- (vi) $I < O_{t+1}, O_t, O_{t-2} >$

```
(i) (ii) (iii) all TRUE
(iv) (v) (vi) all FALSE
```

3 Regression

(a) Consider the following data with one input and one output.



- (i) What is the mean squared training set error of running linear regression on this data (using the model $y = w_0 + w_1 x$)?

  0

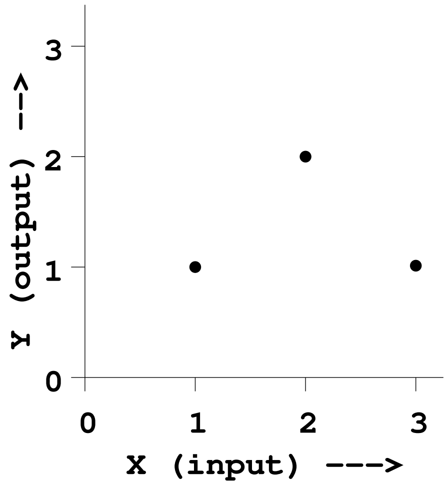
- (ii) What is the mean squared test set error of running linear regression on this data, assuming the rightmost three points are in the test set, and the others are in the training set.

  0

- (iii) What is the mean squared leave-one-out cross-validation (LOOCV) error of running linear regression on this data?

0

(b) Consider the following data with one input and one output.



X	Y
1	1
2	2
3	1

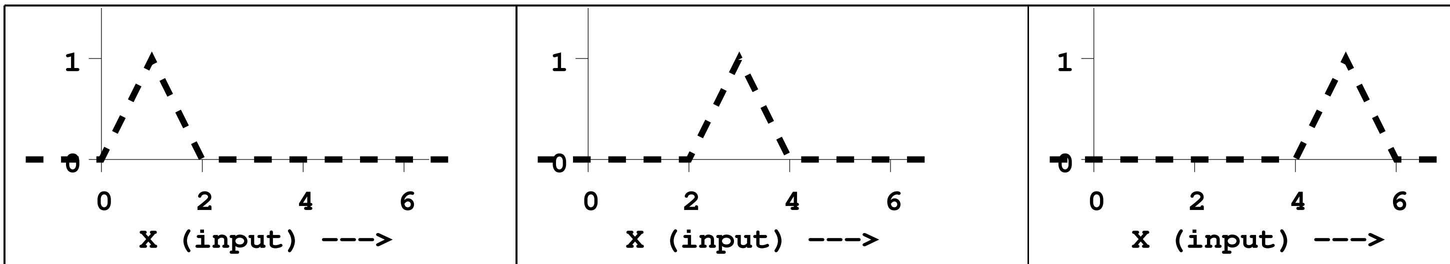
- (i) What is the mean squared training set error of running linear regression on this data (using the model $y = w_0 + w_1 x$)? (Hint: by symmetry it is clear that the best fit to the three datapoints is a horizontal line).

```
SSE = (1/3)^2 + (2/3)^2 + (1/3)^2 = 6/9
MSE = SSE/3 = 2/9
```

- (ii) What is the mean squared leave-one-out cross-validation (LOOCV) error of running linear regression on this data?

```
1/3 * (2^2 + 1^2 + 2^2) = 9/3 = 3
```

(c) Suppose we plan to do regression with the following basis functions:



$$\begin{array}{rcl} \phi_1(x) = & 0 & \text{if } x < 0 \\ \phi_1(x) = & x & \text{if } 0 \le x < 1 \\ \phi_1(x) = & 2 - x & \text{if } 1 \le x < 2 \\ \phi_1(x) = & 0 & \text{if } 2 \le x \end{array}$$

$$\begin{array}{rcl} \phi_2(x) = & 0 & \text{if } x < 2 \\ \phi_2(x) = & x - 2 & \text{if } 2 \le x < 3 \\ \phi_2(x) = & 4 - x & \text{if } 3 \le x < 4 \\ \phi_2(x) = & 0 & \text{if } 4 \le x \end{array}$$

$$\begin{array}{rcl} \phi_3(x) = & 0 & \text{if } x < 4 \\ \phi_3(x) = & x - 4 & \text{if } 4 \le x < 5 \\ \phi_3(x) = & 6 - x & \text{if } 5 \le x < 6 \\ \phi_3(x) = & 0 & \text{if } 6 \le x \end{array}$$

Our regression will be $y = \beta_1\phi_1(x) + \beta_2\phi_2(x) + \beta_3\phi_3(x)$.

Assume all our datapoints and future queries have $1 \le x \le 5$. Is this a generally useful set of basis functions to use? If "yes", then explain their prime advantage. If "no", explain their biggest drawback.

```
NO

They're forced to predict y=0 at x=2 and x=4 (and forced
to be close to zero nearby) no matter what the values of beta.
```

4 Regression Trees

Regression trees are a kind of decision tree used for learning from data with a real-valued output instead of a categorical output. They were discussed in the "Eight favorite regression algorithms" lecture.

On the next page you will see pseudocode for building a regression tree in the special case where all the input attributes are boolean (they can have values 0 or 1).

The MakeTree function takes two arguments:

- D, a set of datapoints
- and A, a set of input attributes.

It then makes the best regression tree it can using only the datapoints and attributes passed to it. It is a recursive procedure. The full algorithm is run by calling MakeTree with D containing every record and A containing every attribute. *Note that this code does no pruning, and that it assumes that all input attributes are binary-valued.*

Now read the code on the next page, after which question (a) will ask you about bugs in the code.

```
MakeTree(D,A)
Returns a Regression Tree

1.     For each attribute a in the set A do...
1.1        Let D0 = { (xk,yk) in D such that xk[a] = 0 }
           // Comment: xk[a] denotes the value of attribute a in record xk
1.2        Let D1 = { (xk,yk) in D such that xk[a] = 1 }
           // Comment: Note that D0 union D1 == D
           // Comment: Note too that D0 intersection D1 == empty
1.3        mu0 = mean value of yk among records in D0
1.4        mu1 = mean value of yk among records in D1
1.5        SSE0 = sum over all records in D0 of (yk - mu0) squared
1.6        SSE1 = sum over all records in D1 of (yk - mu0) squared
1.7        Let Score[a] = SSE0 + SSE1

2.    // Once a score has been computed for each attribute, let...
      a* = argmax  Score[a]
              a

3.    Let D0 = { (xk,yk) in D such that xk[a*] = 0 }
4.    Let D1 = { (xk,yk) in D such that xk[a*] = 1 }
3.    Let LeftChild = MakeTree(D0,A - {a*})
      // Comment: A - {a*} means the set containing all elements of A except for a*
4.    Let RightChild = MakeTree(D1,A - {a*})
5.    Return a tree whose root tests the value of a*, and whose ``a* = 0''
      branch is LeftChild and whose ``a* = 1'' branch is RightChild.
```

(a) Beyond the obvious problem that there is no pruning, there are three bugs in the above code. They are all very distinct. One of them is at the level of a typographical error. The other two are more serious errors in logic. Identify the three bugs (remembering that the lack of pruning is not one of the three bugs), explaining why each one is a bug. It is not necessary to explain how to fix any bug, though you are welcome to do so if that's the easiest way to explain the bug.

`Line 1.6 should use (yk - mu1)^2`

`Line 2 should use argmin`

`The algorithm is missing the base case of the recursion`

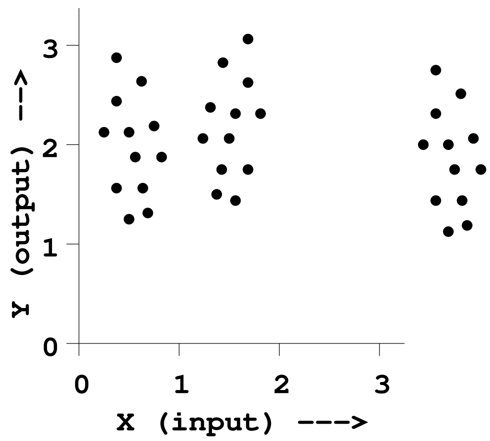
(b) Why, in the recursive calls to MakeTree, is the second argument "$A - \{a*\}$" instead of simply "A"?

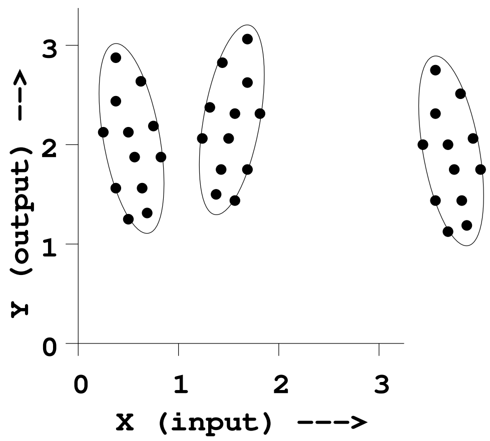
```
Because a* can't possibly be chosen in any recursive calls
```

5 Clustering

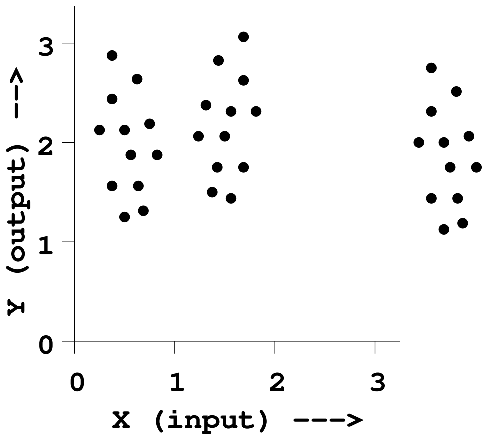
In the left of the following two pictures I show a dataset. In the right figure I sketch the globally maximally likely mixture of three Gaussians for the given data.

- Assume we have protective code in place that prevents any degenerate solutions in which some Gaussian grows infinitesimally small.
- And assume a GMM model in which all parameters (class probabilities, class centroids and class covariances) can be varied.

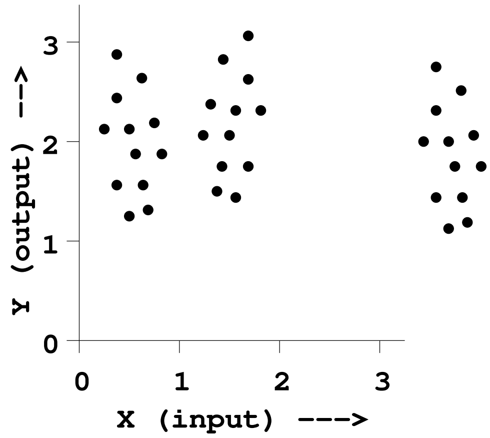




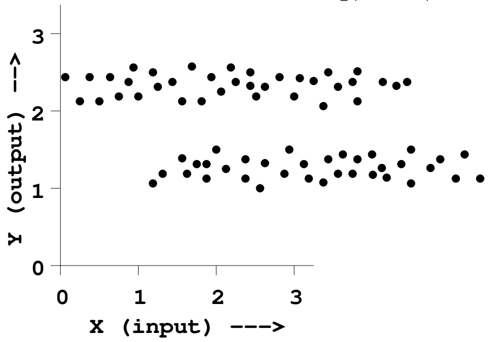
(a) Using the same notation and the same assumptions, sketch the globally maximally likely mixture of **two** Gaussians.



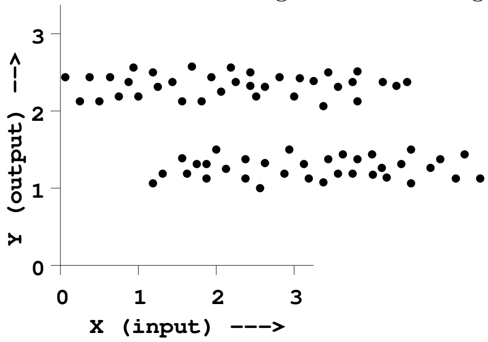
(b) Using the same notation and the same assumptions, sketch a mixture of three distinct Gaussians that is stuck in a suboptimal configuration (i.e. in which infinitely many more iterations of the EM algorithm would remain in essentially the same suboptimal configuration). *(You must not give an answer in which two or more Gaussians all have the same mean vectors—we are looking for an answer in which all the Gaussians have distinct mean vectors).*



(c) Using the same notation and the same assumptions, sketch the globally maximally likely mixture of two Gaussians in the following, new, dataset.



(d) Now, suppose we ran k-means with $k = 2$ on this dataset. Show the rough locations of the centers of the two clusters in the configuration with globally minimal distortion.



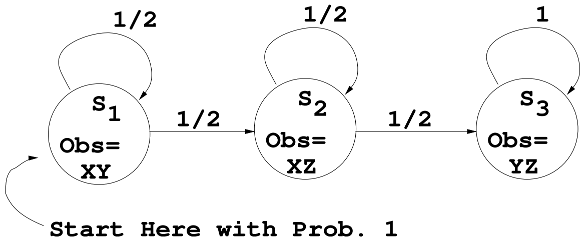
6 Regression algorithms

For each empty box in the following table, write in "Y" if the statement at the top of the column applies to the regression algorithm. Write "N" if the statement does not apply.

	No matter what the training data is, the predicted output is guaranteed to be a continuous function of the input. (i.e. there are no discontinuities in the prediction). If a predictor gives continuous but undifferentiable predictions then you should answer "Y".	The cost of training on a dataset with R records is at least $O(R^2)$: quadratic (or worse) in R. For iterative algorithms marked with (*) simply consider the cost of one iteration of the algorithm through the data.
Linear Regression	Y	N
Quadratic Regression	Y	N
Perceptrons with sigmoid activation functions (*)	Y	N
1-hidden-layer Neural Nets with sigmoid activation functions (*)	Y	N
1-nearest neighbor	N	N
10-nearest neighbor	N	N
Kernel Regression	Y	N
Locally Weighted Regression	Y	N
Radial Basis Function Regression with 100 Gaussian basis functions	Y	N
Regression Trees	N	N
Cascade correlation (with sigmoid activation functions)	Y	N
Multilinear interpolation	Y	N
MARS	Y	

7 Hidden Markov Models

Warning: this is a question that will take a few minutes if you really understand HMMs, but could take hours if you don't. Assume we are working with this HMM



$a_{11} = 1/2$	$a_{12} = 1/2$	$a_{13} = 0$	$b_1(X) = 1/2$	$b_1(Y) = 1/2$	$b_1(Z) = 0$	$\pi_1 = 1$
$a_{21} = 0$	$a_{22} = 1/2$	$a_{23} = 1/2$	$b_2(X) = 1/2$	$b_2(Y) = 0$	$b_2(Z) = 1/2$	$\pi_2 = 0$
$a_{31} = 0$	$a_{32} = 0$	$a_{33} = 1$	$b_3(X) = 0$	$b_3(Y) = 1/2$	$b_3(Z) = 1/2$	$\pi_3 = 0$

Where

$$
\begin{aligned}
a_{ij} &= P(q_{t+1} = S_j | q_t = S_i) \\
b_i(k) &= P(O_t = k | q_t = S_i)
\end{aligned}
$$

Suppose we have observed this sequence

XZXYYZYZZ

(in long-hand: $O_1 = X, O_2 = Z, O_3 = X, O_4 = Y, O_5 = Y, O_6 = Z, O_7 = Y, O_8 = Z, O_9 = Z$). Fill in this table with $\alpha_t(i)$ values, remembering the definition:

$$\alpha_i(t) = P(O_1 \wedge O_2 \wedge ...O_t \wedge q_t = s_i)$$

So for example,

$$\alpha_3(2) = P(O_1 = X \wedge O_2 = Z \wedge O_3 = X \wedge q_3 = S_2)$$

t	$\alpha_t(1)$	$\alpha_t(2)$	$\alpha_t(3)$
1	1/2		
2		1/8	
3		1/32	
4			1/128
5			1/256
6			1/512
7			1/1024
8			1/2048
9			1/4096

8 Locally Weighted Regression

Here's an argument made by a misguided practitioner of Locally Weighted Regression.

> Suppose you have a dataset with R_1 training points and another dataset with R_2 test points. You must predict the output for each of the test points. If you use a kernel function that decays to zero beyond a certain Kernel width then Locally Weighted Regression is computationally cheaper than regular linear regression. This is because with locally weighted regression you must do the following for each query point in the test set,
>
> - Find all the points that have non-zero weight for this particular query.
> - Do a linear regression with them (after having weighted their contribution to the regression appropriately).
> - Predict the value of the query.
>
> whereas with regular linear regression you must do the following for each query point:
>
> - take all the training set datapoints.
> - Do an unweighted linear regression with them.
> - Predict the value of the query.
>
> The locally weighted regression frequently finds itself doing regression on only a tiny fraction of the datapoints because most have zero weight. So most of the local method's queries are cheap to answer. In contrast, regular regression must use every single training point in every single prediction and so does at least as much work, and usually more.

This argument has a serious error. Even if it is true that the kernel function causes almost all points to have zero weight for each LWR query the argument is wrong. What is the error?

```
Linear regression only needs to learn its weights (i.e. do the
appropriate matrix inversion) once in total. LWR must do
a separate matrix inversion for each test point.
```

9 Nearest neighbor and cross-validation

At some point during this question you may find it useful to use the fact that if U and V are two independent real-valued random variables then $Var[aU + bV] = a^2\,Var[U] + b^2\,Var[V]$.

Suppose you have 10,000 datapoints $\{(x_k, y_k) : k = 1, 2, ..., 10000\}$. Your dataset has one input and one output. The kth datapoint is generated by the following recipe:

$$\begin{aligned} x_k &= k/10000 \\ y_k &\sim N(0, 2^2) \end{aligned}$$

So that y_k is all noise: drawn from a Gaussian with mean 0 and variance $\sigma^2 = 4$ (and standard deviation $\sigma = 2$). Note that its value is independent of all the other y values. You are considering two learning algorithms:

- **Algorithm NN:** 1-nearest neighbor.
- **Algorithm Zero:** Always predict zero.

(a) What is the expected Mean Squared Training Error for **Algorithm NN**?

```
0
```

(b) What is the expected Mean Squared Training Error for **Algorithm Zero**?

```
4
```

(c) What is the expected Mean Squared Leave-one-out Cross-validation Error for **Algorithm NN**?

```
8 = E[(xk - x[x+1])^2]
```

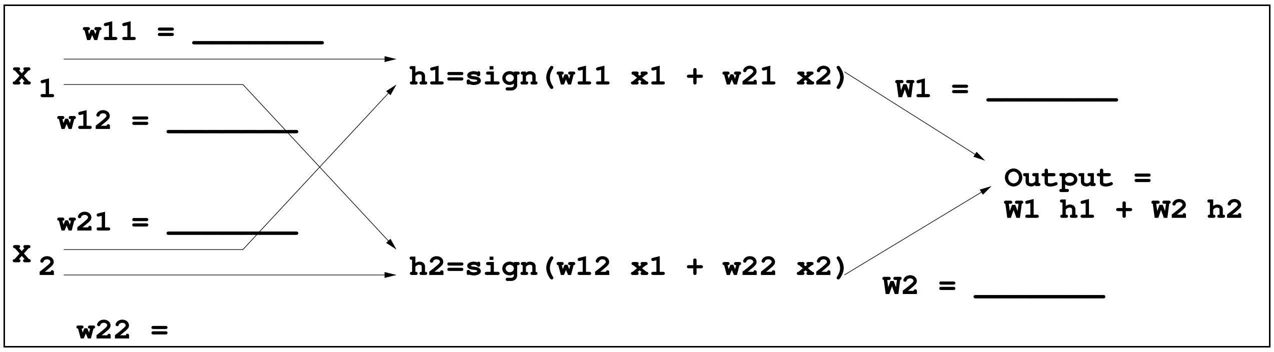
(d) What is the expected Mean Squared Leave-one-out Cross-validation Error for **Algorithm Zero**?

4

10 Neural Nets

(a) Suppose we are learning a 1-hidden-layer neural net with a sign-function activation

$$\begin{aligned} \text{Sign}(z) &= \quad 1 \quad \text{if } z \geq 0 \\ \text{Sign}(z) &= \ -1 \quad \text{if } z < 0 \end{aligned}$$



We give it this training set, which represents the exclusive-or function if you interpret -1 as false and +1 as true:

X_1	X_2	Y
1	1	-1
1	-1	1
-1	1	1
-1	-1	-1

On the diagram above you must write in six numbers: a set of weights that would give zero training error. (Note that constant terms are not being used anywhere, and note too that the output does not need to go through a sign function). Or..if it impossible to find a satisfactory set of weights, just write "impossible".

`Impossible`

(b) You have a dataset with one real-valued input x and one real-valued output y in which you believe

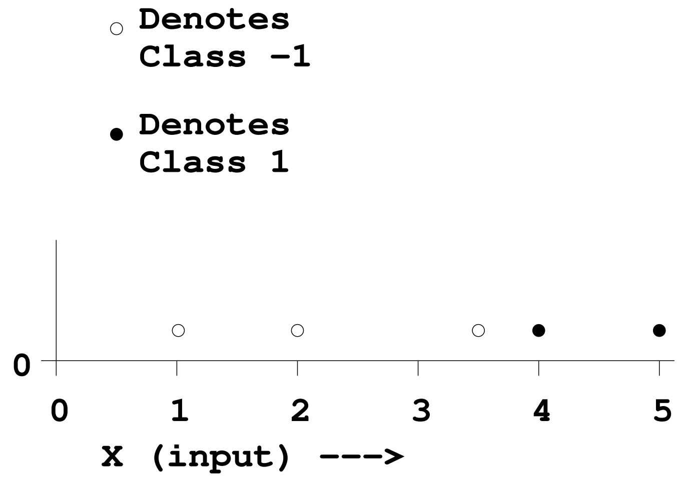
$$y_k = \exp(wx_k) + \epsilon_k$$

where (x_k, y_k) is the kth datapoint and ϵ_k is Gaussian noise. This is thus a neural net with just one weight: w.

Give the update equation for a gradient descent approach to finding the value of a that minimizes the mean squared error.

11 Support Vector Machines

Consider the following dataset. We are going to learn a linear SVM from it of the form $f(x) = \text{sign}(wx + b)$.



X	Y
1	-1
2	-1
3.5	-1
4	1
5	1

(a) What values for w and b will be learned by the linear SVM?

w = 4, b = -15

(b) What is the training set error of the above example? (expressed as the percentage of training points misclassified)

0

(c) What is the leave-one-out cross-validation error of the above example? (expressed as the percentage of left-out points misclassified)

2 wrong => 40%

(d) **True or False:** Even with the clever SVM Kernel trick it is impossibly computationally expensive, even on a supercomputer, to do the following:

Given a dataset with 200 datapoints and 50 attributes learn an SVM classifier with full 20th-degree-polynomial basis functions and then apply what you've learned to predict the classes of 1000 test datapoints.

FALSE

12 VC Dimension

(a) Suppose we have one input variable x and one output variable y. We are using the machine $f_1(x, \alpha) = \text{sign}(x + \alpha)$. What is the VC dimension of f_1?

`1`

(b) Suppose we have one input variable x and one output variable y. We are using the machine $f_2(x, \alpha) = \text{sign}(\alpha x + 1)$. What is the VC dimension of f_2?

`1`

(c) Now assume our inputs are m-dimensional and we use the following two-level, two-choice decision tree to make our classification:

```
                is x[A] < B?
                /          \
        if no  /            \  if yes
              /              \
          is x[C] < D?      is x[E] < F?
             / \                  / \
     if no /     \ if yes    if no/    \ if yes
          /       \              /       \
         /         \            /         \
     Predict     Predict     Predict    Predict
     Class G     Class H     Class I    Class J
```

Where the machine has 10 parameters

$$
\begin{aligned}
A &\in \{1, 2, ..., m\} \\
B &\in \Re \\
C &\in \{1, 2, ..., m\} \\
D &\in \Re \\
E &\in \{1, 2, ..., m\} \\
F &\in \Re \\
G &\in \{-1, 1\} \\
H &\in \{-1, 1\} \\
I &\in \{-1, 1\} \\
J &\in \{-1, 1\}
\end{aligned}
$$

What is the VC-dimension of this machine?

`4`