Name and Student ID: Signature:

Machine Learning BLG527E, Jan 18, 2012, Final Exam. ANSWERS

1	2	3	4	Total

Duration: 120 minutes.
Write your answers neatly in the space provided for them. Write your name on each sheet. Books and notes are closed. Good Luck!

QUESTIONS

QUESTION1) [40 points, 10 points each] (use at most 10 sentences per question, use of formulas, drawings etc. to better express yourself is encouraged.)

a) How do you control an SVM's model complexity?

*-SVM minimizes the error function: E = C * Etrain + ||w||2. We can change the value of C, smaller C means less emphasis on training error and more emphasis on small w, hence simpler models. Small w also means large margin*

-You can use different kernel functions. For example linear kernel is less complicated than the quadratic.

-You can force the SVM to use smaller number of support vectors.

b) How do you train (i.e. decide on the best weights) a multilayer perceptron?
How can you decide on the best complexity?
How can you speed up the learning process?

-We minimize the training error by means of gradient descent. We subtract the learning rate times the gradient of the error with respect to weights from the current weight values. We can use backpropagation algorithm to compute the gradient.

-We can use k-fold cross validation. We divide up the whole dataset into K parts. Use the kth part for validation and all the other parts for training. We compute the average validation error for each different number of hidden units. The number of hidden units with the smallest validation error should be chosen.

-We can speed up the learning by means of adaptive learning rate and momentum. We can also use the Hessian matrix to better model the error surface.

c) How do you use Parzen windows for density estimation, classification and regression?

Density estimation: *Parzen windows compute the probability of x based on the distance between x and all the training instances using the kernel estimator as:*

$$\hat{p}(x) = \frac{1}{Nh}\sum_{t=1}^{N} K\left(\frac{x - x^t}{h}\right)$$ *here* $$K(u) = \frac{1}{\sqrt{2\pi}}\exp\left[-\frac{u^2}{2}\right]$$ *is the kernel function.*

All training instances are used for the estimation. The training instances closer to x have larger kernel values and hence affect the density estimator more.

$$\hat{p}(\mathbf{x} \mid C_i) = \frac{1}{N_i h^d}\sum_{t=1}^{N} K\left(\frac{\mathbf{x} - \mathbf{x}^t}{h}\right) r_i^t \quad \hat{P}(C_i) = \frac{N_i}{N}$$

$$g_i(\mathbf{x}) = \hat{p}(\mathbf{x} \mid C_i)\hat{P}(C_i) = \frac{1}{Nh^d}\sum_{t=1}^{N} K\left(\frac{\mathbf{x} - \mathbf{x}^t}{h}\right) r_i^t$$

Classification: *where r_i^t are the class labels for training data.*

$$\hat{g}(x) = \frac{\sum_{t=1}^{N} K\left(\frac{x - x^t}{h}\right) r^t}{\sum_{t=1}^{N} K\left(\frac{x - x^t}{h}\right)}$$

Regression: *where r^t s are the given outputs for training data.*

d) What are the differences and similarities between logistic regression and multilayer perceptron classifier?

Differences:

Logistic regression classifier has only one layer, MLP has at least two layers.
Logistic regression classifier can not classify data which is not linearly separable, MLP can.
For MLP we need more complicated routines (backpropagation) to compute the error gradient.

Similarities:

Both use a weightes sum of inputs passed through a sigmoidal nonlinearity.
Both are trained using gradient descent.
Both can be used for classification or regression problems.

QUESTION2) [20points] Consider the 2-means algorithm on a set S consisting of the following 4 points in the plane: b=(1,3), c=(2,3), e=(3,2), g=(4,2). The algorithm uses the Euclidean distance metric to assign each point to its nearest centroid; ties are broken in favor of the centroid to the left/down. A starting configuration is a subset of 2 starting points from S that form the initial centroids. A 2-partition is a partition of S into 2 subsets; thus {b,c}, {e,g} is a 2-partition; clearly any 2-partition induces a set of two centroids in the natural manner. A 2-partition is **stable** if repetition of the 2-means iteration with the induced centroids leaves it unchanged.
a. How many starting configurations are there?
b. What are the stable 2-partitions?
c. What is the number of starting configurations leading to each of the stable 2-partitions in (b) above?

*a. There are 4 choose 2 = 4!/(2!*2!) = 6 starting configurations: {b,c}, {b,e},{b,g},{c,e},{c,g},{e,g}.*
b. There is only one stable partition: {b,c} and {e,g}. You can show that all other partitions lead to this one after a number of kmeans interations and this partition remains stable.
c. All 6 configurations lead to the same partition.

QUESTION3) [20points] You need to write a machine learning algorithm to help people who are learning to write Turkish.

i) The algorithm needs to identify whether words in a sentence are in the order they are supposed to be in Turkish. For example, saying “Yaz Ali.” is less common than “Ali yaz.” in written language.
ii)If they are not, you need to suggest a better ordering.

What would you ask your customer to provide you with as data?
What machine learning method(s) would you use to solve tasks i) and ii).
How would you measure the success of your methods?

-I would ask for a large written corpus (collection of documents) and also whether each word is a noun, verb, adjective etc. Note that we do not really need a set of documents written in wrong Turkish (negative class instances).
-I would use an HMM. I would learn the A, B, PI parameters of the HMM using the Baum Welch algorithm on training data. For outputs, I would not use the words themselves but what the word is grammatically (e.g. noun, verb etc.). In order to figure out if a sentence is a valid sentence or not, I would produce all possible orderings of words in the sentence. For each of these sentences O , I would evaluate Pr(O|Lambda). If the given sentence has the highest probability, then I would tell my customer that it is, otherwise I would suggest the highest probabilty sentence to the customer.
-In order to evaluate the hmm, I would use a training and validation partition of the data. I would report the normalized number of misplacements between the best sentence predicted by hmm and the sentence in the validation partition as the performance measure.

Name and Student ID: Signature:

QUESTION4) [20points] You need to come up with a decision tree that separates four data points labeled as follows U = {(x,r(x))} = {((1,2),+), ((2,3),+), ((2,1),-), ((3,2),-)}.

Use entropy impurity: $$I_m = -\sum_{i=1}^{K} p_m^i \log_2 p_m^i$$
where p_m^1 is the ratio of the instances in node m which are in class i.

Let i=1 denote class + and i=2 denote class –

*At the root: p^1_m = 2/4=1/2, p^2_m = 2/4=1/2, I_m = -(1/2*log2(1/2) + 1/2*log2(1/2))=1*

We can partition along the first (x) or second (y) coordinate. For both coordinates we get the same best impurity. So we will show how we proceed for the x coordinate:
x values are given as: 1,2,3. We can partition as x<1, x<2, x<3, x>=3

For x<1 we do not partition at all, impurity remains the same.

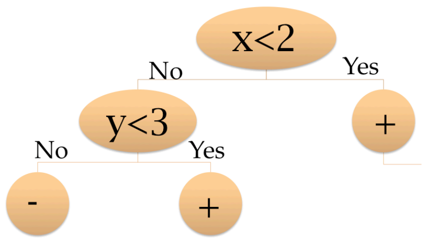
x<2 partition: partition1: {((1,2),+)} partition2: {((2,3),+), ((2,1),-), ((3,2),-)}.
*I_m^1=0, I_m^2=-(1/3*log(1/3)+2/3*log(2/3))*
*ImPartition = ¼ * I_m^1 + ¾* I_m^2 =3/4 * (-(1/3*log(1/3)+2/3*log(2/3)))*

x<3 partition: partition1: {((1,2),+), ((2,3),+), ((2,1),-)} partition2: {((3,2),-)}.
*I_m^2=0, I_m^1=-(1/3*log(1/3)+2/3*log(2/3))*
*ImPartition = ¼ * I_m^2 + ¾* I_m^1 =3/4 * (-(1/3*log(1/3)+2/3*log(2/3)))*

x>=3 partition: same results as x<3 partition.

Since they have the same partition impurity, we can choose any of these three partitions. So, we choose, for example to divide up the input space according to x<2.
Then we need to divide the (x<2==NO) nodes one more time using the impurity calculations this time in y dimension.

The resulting decision tree is:


x<2
No
Yes
y<3
No
Yes
+
-
+