CS4780 Final

December 2015

NAME:	
Net ID:	
Email:	

GML	
CART	
Bias Variance	
Kernel Machines	
Neural Networks	
TOTAL	

1 [??] General Machine Learning

Please identify if these statements are either True or False. Please justify your answer **if false**. Correct "True" questions yield 1 point. Correct "False" questions yield two points, one for the answer and one for the justification.

T/F In Adaboost, in each iteration weights of misclassified examples go up by the same multiplicative factor.
True.

T/F When a CART tree is built, you stop when a new split does not resolve in a reduction of impurity.
False, CART trees are myopic. E.g. in the XOR data set, the first splits do not improve purity.

T/F The impurity of a CART node is the sum of the two impurities of its children

$$G(S) = w_L G(S_L) + w_R G(S_R), \tag{1}$$

where S is the data set of the parent node, S_L and S_R are the two subsets that fall into the left and right child respectively. The weights w_L, w_R correspond to how many nodes the left and right sub-trees have respectively.
False, the left and right impurities must be weighted by the percentage of points that end up there (e.g. $\frac{|S|}{|S_L|}$).

T/F When built to full depth, CART trees tend to suffer from high bias.
False, they suffer from high variance.

T/F Boosting reduces bias.
True

T/F In Boosting you stop when the training error is zero.
False, you can continue boosting when the training error is zero and often it still improves the test error (as it grows the margin).

T/F Bagging introduces a new hyper-parameter (m), which specifies how many classifiers are averaged. This has to be set very carefully. If m is too high, the classifier is prone to overfitting.
False, Bagging converges as m increases. There is no increasing of overfitting.

T/F In Bagging, the samples for each individual classifier are drawn from the same distribution as the original data set and the samples are independent and identically distributed (i.i.d.).
False, the samples are not independent.

T/F A *weak learner* is a classifier that obtains worse than 50% error on binary classification tasks.
False, it obtains slightly better than 50% error.

T/F Kernel machines project the data into potentially very high dimensional spaces. This can make the algorithm very slow and it an be very time consuming to store these very high dimensional vectors.
False, the projection into high dimensions is *implicit* through the *kernel-trick*, and the high dimensional vectors are never stored.

T/F A kernel function $k(\mathbf{x}, \mathbf{z})$ is well defined if and only if it corresponds to some function ϕ such that $k(\mathbf{x}, \mathbf{z}) = \phi(\mathbf{x})^\top \phi(\mathbf{z})$. (Although this function may be intractable to compute.)
True

T/F Kernel ridge regression is a universal approximator.
True

T/F Neural Networks are universal approximators.
True

T/F The Perceptron is a universal approximators.
False, far from it. In fact, it only converges if the data set is linearly separable.

T/F Kernel matrices have to be positive semi-definite.
True

T/F The SVM optimization problem has a dual formulation. This dual has the same solution as the original (primal) formulation.
True

T/F For SVMs, *Support Vectors* are those training inputs that are on the correct side of the decision boundary.
False, support vectors are inputs that fall between the hyper-plane and the margin.

T/F The polynomial kernel with degree 2 is equivalent to computing an inner-product after a mapping into an infinitely dimensional feature space.
False, it is equivalent to an inner-product after a finite dimensional transformation. $\mathbf{x} \to \left[1, x_1, \ldots, x_d, x_1^2, x_1 x_2, \ldots, x_d^2\right]$.

T/F In order to *kernelize* an algorithm it has to be stated such that inputs are only accessed in terms pair-wise squared distances.
False, inner-products.

T/F Gaussian Processes and kernel-SVMs have in common that they both access the training inputs only through a kernel function.
True.

T/F One advantage of Gaussian Processes is that they return distributions, which gives a natural measure of "uncertainty".
True.

2 [20] CART

Assume you are provided with the following data:

$$\begin{aligned}
\mathbf{x}_1 &= [0,1,1]^\top & y_1 &= -1 \\
\mathbf{x}_2 &= [0,0,1]^\top & y_2 &= -1 \\
\mathbf{x}_3 &= [0,1,0]^\top & y_3 &= -1 \\
\mathbf{x}_4 &= [1,1,1]^\top & y_4 &= -1 \\
\mathbf{x}_5 &= [1,0,1]^\top & y_5 &= -1 \\
\mathbf{x}_6 &= [1,1,1]^\top & y_6 &= -1 \\
\mathbf{x}_7 &= [1,0,1]^\top & y_7 &= +1 \\
\mathbf{x}_8 &= [1,0,1]^\top & y_8 &= +1 \\
\mathbf{x}_9 &= [1,1,0]^\top & y_9 &= +1
\end{aligned}$$

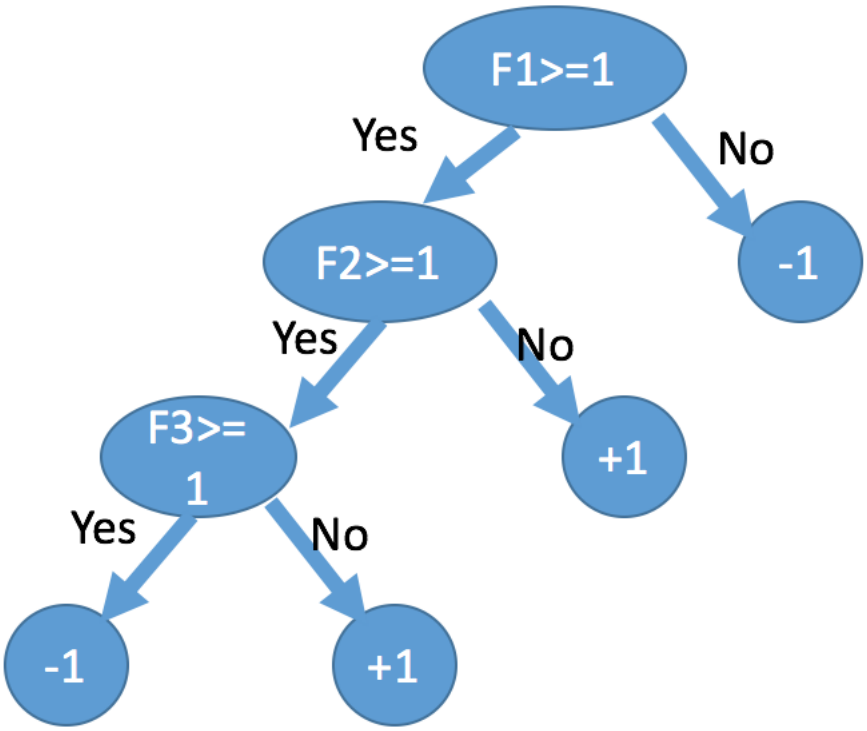
1. (1) State the base cases (terminating criteria) of the ID3 algorithm

   (a) all the data points in a subset of have the same label

   (b) there are no more attributes could be used to split the subset

   (c) Hit threshold of max depth

2. (2) Build a tree that splits on feature one on the root node, then on feature 2 on the next level (all nodes in that level), and feature 3 on the next level etc. until the termination criteria is reached. Draw the tree below.

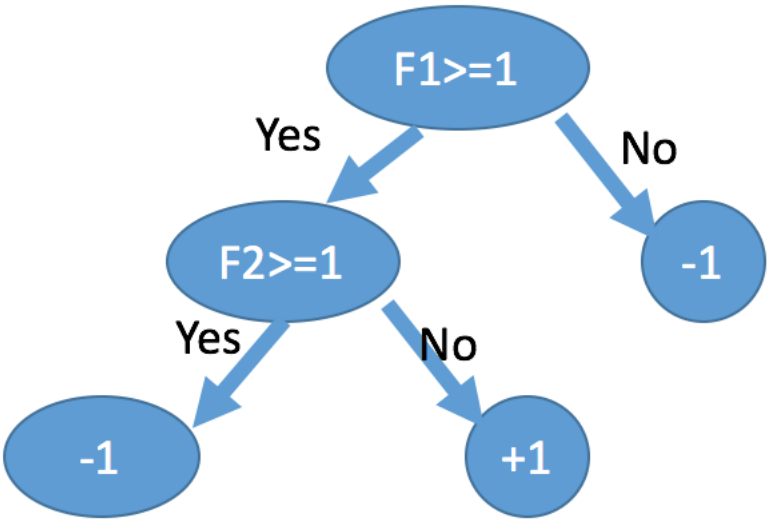


3. (2) What is the Gini impurity at the root of your tree?

$\frac{4}{9}$

4. (3) Assume you have the following validation set, and prune the tree obtained in part 2. Draw the pruned tree. What is the validation error before pruning? What is the validation error after pruning?

$$\mathbf{v_1} = [0, 1, 0]^\top \qquad y_{v,1} = -1$$
$$\mathbf{v_2} = [1, 0, 1]^\top \qquad y_{v,2} = +1$$
$$\mathbf{v_3} = [1, 1, 0]^\top \qquad y_{v,3} = -1$$



5. (2) Add a training sample $\mathbf{x}_{10}$ with label y_{10} such that the data set cannot be classified correctly with a single decision tree.

6. (4) One could argue that the $k-$nearest neighbor classifier with $KD-$Trees is similar to an ID3 tree classifier. Name one advantage of each over the other.
kNN: The decision boundary is smoother. If a test input is close to the boundary of a tree split it can be classified differently than the majority of inputs inside that partition.
ID3 Tree: It is a lot faster.

7. (6) Suppose we define the loss function at a leaf of a regression tree as

$$L(S, h(S)) = \frac{1}{|S|} \prod_{(x,y)\in S} \exp((y - h(x))^2))$$

Given this loss function, what is the prediction value at a leaf?

take log, set derivative equal to 0. $y_S = sum_{(x,y)\in S} y$

3 [25] Bias Variance

1. (2) Your machine learning algorithm's validation error is too high. Your data is pretty clean (i.e. little noise). How can you identify if you suffer from high variance or high bias?

2. (6) Name three actions to counter *high Bias* and three actions to counter *high Variance*.
high Bias: Boosting, less regularization, feature engineering / more expressive features, kernelization; high Variance: Bagging, more regularization, feature selection, use simpler model, add more data

3. (5) You use 1-nearest neighbor (1-NN) classification on a data set, with test error ϵ. You know that 1-NN suffers from high variance, so you apply Bagging. However, as the number of averaged classifiers m becomes very large, you observe that the test error of the bagged classifier matches exactly ϵ. Can you explain this phenomenon?
Bagging selects inputs randomly with replacement. If a point is selected multiple times, it has no effect on the 1-nearest neighbor rule. In expectation, each input is selected in $> 50\%$ of the classifiers. This means for a test input, the nearest neighbor of the 1-NN classifier is the closest point in the majority of the cases, and therefore the label of the original nearest neighbor is the most commonly observed nearest neighbor label.

4. (2) In **AdaBoost** let the error of the first weak learner be $\epsilon > 0$. Remember from class that $\alpha = \frac{1}{2}\log(\frac{1-\epsilon}{\epsilon})$. State the weight update for correctly classified inputs and misclassified inputs (without normalization) in terms of e^{α}.
correctly classified inputs: $w_i \leftarrow w_i e^{-\alpha}$.
misclassified inputs: $w_i \leftarrow w_i e^{\alpha}$.

5. (5) Show that the normalization constant Z (the sum of all updated weights) is $Z = 2n\sqrt{\epsilon(1-\epsilon)}$, where n is the number of training inputs.
$Z = \epsilon n e^{\alpha} + (1-\epsilon)ne^{-\alpha} = 2n\sqrt{\epsilon(1-\epsilon)}$

6. (5) Finally, show that after the weight updates and normalizing all weights, the re-weighted error of the first weak learner is exactly 0.5.

$$n\epsilon\frac{e^{\alpha}}{Z} = n\frac{\sqrt{\epsilon(1-\epsilon)}}{2n\sqrt{\epsilon(1-\epsilon)}} = \frac{1}{2}$$

4 [12] Kernel Machines

1. (2) Show that a well-defined kernel function $k(\mathbf{x}, \mathbf{z})$ is symmetric. (i.e. $k(\mathbf{x}, \mathbf{z}) = k(\mathbf{z}, \mathbf{x})$)

2. (4) Show that $k(\mathbf{x}, \mathbf{z}) = \frac{\mathbf{x}^\top \mathbf{z}}{\|\mathbf{x}\|\|\mathbf{z}\|}$ is a well-defined kernel.

3. (2) Can you kernelize ID3 decision trees with the entropy splitting rule? If so, state the algorithm, if not, explain why it is impossible.

4. (2) One of the most popular kernels is the RBF kernel, $k(\mathbf{x}, \mathbf{z}) = \exp(-\gamma\|\mathbf{x} - \mathbf{z}\|^2)$. Assume we have two training points $\mathbf{x}_1, \mathbf{x}_2$ and a test-point $\mathbf{z}$ (for some $\gamma > 0$). In addition, you know that $\mathbf{z}$ is very close to $\mathbf{x}_1$, but very far away from $\mathbf{x}_2$. What statements are true about the kernel entries (multiple are possible):

   a) $k(\mathbf{x}_1, \mathbf{z}) \geq 0$ and $k(\mathbf{x}_2, \mathbf{z}) \geq 0$

   b) $k(\mathbf{x}_1, \mathbf{z})$ is close to 0, whereas $k(\mathbf{x}_2, \mathbf{z})$ is very large.

   c) $k(\mathbf{x}_1, \mathbf{z})$ is close to 1, whereas $k(\mathbf{x}_2, \mathbf{z})$ is very close to 0.

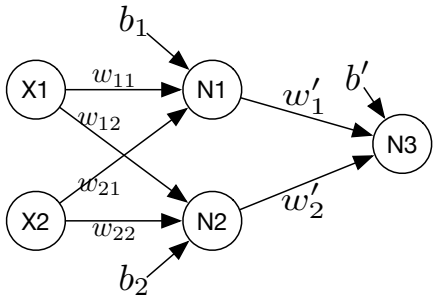
   d) $k(\mathbf{x}_1, \mathbf{z})k(\mathbf{x}_2, \mathbf{z})$ is close to 0.

   a),c),d) are true. 0.5 points for each correct bullet

5. (2) Explain the effects on the classifier if the constant γ is moved from a very large value to a very small value.

5 [11] Neural Network

Consider the following network with two inputs x_1, x_2, one output node N_3 and two hidden node N_1,N_2.

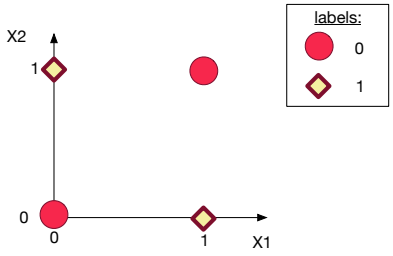


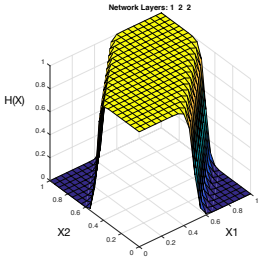
The weight w_{ij} connects X_i to N_j. Weight w'_j connects N_j to the output N_3.

1. Suppose N_1,N_2, N_3 all use **Sigmoid** transition functions, that is, their output is computed using the equation $\sigma(z) = \frac{1}{1+e^{-z}}$. Each node also has a bias term (b_1, b_2, b' respectively.)

   a) (2) A common approach to change the sigmoidal shape is to introduce a constant $a \geq 0$ and define the sigmoid function as $\sigma(z;a) = \frac{1}{1+e^{-az}}$. Sketch the sigmoid function $\sigma(z;a)$ as a function of z for $a = 1$ and for a very large ($a > 1000$).

   b) (5) Assume we use a sigmoid with a very large value of a. Assign weights such that this network **classifies** the XOR data set correctly. (Hint: The right figure shows the prediction of a neural net trained on the XOR data set.)





Please fill in your weights below:

w_{11}	w_{12}
w_{21}	w_{22}

w'_1
w'_2

b_1
b_2

b'

2. (2) Could you still classify the data correctly if node N_1 is removed from the network? If yes, assign the weights. If no, explain why not.
   No, as the classifier turns into a linear classifier.

3. (2) Name an advantage of Rectified Linear Units over Sigmoidal transition functions.
   They do not suffer from vanishing gradients.

This page was intentionally left blank.