CSE517a Midterm

P

February 2015

NAME:	
Student ID:	
Email:	

GML	
kNN	
CART	
Bias Variance	
Boosting Bagging	
ERM	
TOTAL	

1 [??] General Machine Learning

Either circle **T** or **F**. Questions declared as **True** require no explanation (worth 1 point). Questions declared as **False** require a **one sentence** explanation (worth 2 points).

T/F Decreasing the depth of your decision tree (through pruning) will reduce test error.
False, only if your tree is overfitting.

T/F In k-fold cross validation you leave k inputs out, train your classifier on the remaining $n-k$ inputs and evaluate it on the leave-out inputs. You do this repeatedly and average to obtain a good estimate of your classifier's performance.

False, you make k splits, each of size n/k.

T/F The Adaboost algorithm stops when the training error is zero. This happens after a $O(log(n))$ iterations.
False, it stops if the maximum number of iterations is reached or if $\epsilon_t \geq 0.5$. Often you can keep boosting for a long time even after the training error is zero.

T/F Gradient Boosting is performing (stage-wise) gradient descent in function space.
True

T/F When you split your data into train and test you have to make sure you *always* do the splitting uniformly at random.
False, for example if there is a temporal component to your data you usually must split by time.

T/F If a classifier obtains 0% training error it cannot have 100% testing error.
False, a simple counter example is a classifier that memorizes the training data set.

T/F Increasing regularization tends to reduce the bias of your classifier.
False, it reduces the variance of your classifier. Often it can **increase** bias when it is set too high.

T/F If run without depth limit, the ID3 algorithm returns the maximally compact decision tree that is consistent with a data set (if it exists).
False, this is NP hard - the ID3 algorithm is only an approximation thereof.

T/F The best classifiers make no assumptions about your data at all.
False, all (working) classifiers must make assumptions on the data.

T/F Random Forests learn many high variance CART trees and reduce this variance by averaging the results. That's basically Bagging applied to (slightly modified) CART trees.
True (the slight modification isthe random feature sub-selection per split.)

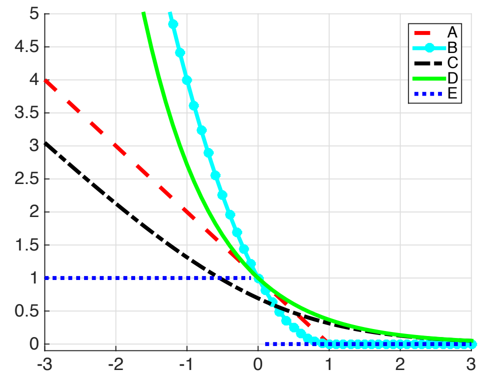
T/F As your training data set size, n, approaches infinity, the $k-$nearest neighbor classifier is guaranteed to have an error no worse than twice the Bayes optimal error.
True

T/F Squared loss regression trees require a time complexity $O(n^2)$ per split.
False, it can be done with $O(n\log(n))$ time complexity (HW2).

T/F The Bayes optimal error is the best classification error you could get if there was no noise.

False, it is the best classification error you could get if you knew the data distribution. In fact, this error is due to label uncertainty, *i.e.* noise.

2 [15] ERM and SVM



1. (3) Write down the loss and regularizers of *SVM*, *LASSO* and *Ridge Regression*. Provide names for all loss functions and regularizers.

$$\mathcal{L}_{svm} = \frac{1}{n}\sum_{i=1}^{n} \underbrace{\max\left[1 - y_i(\mathbf{w}^\top \mathbf{x}_i + b), 0\right]}_{\text{hinge-loss}} + \lambda \underbrace{\|\mathbf{w}\|_2^2}_{l_2-regularizer}$$

$$\mathcal{L}_{lasso} = \frac{1}{n}\sum_{i=1}^{n} \underbrace{((\mathbf{w}^\top \mathbf{x} + b) - y_i)^2}_{\text{squared loss}} + \lambda \underbrace{|\mathbf{w}|}_{l_1-regularizer}$$

$$\mathcal{L}_{ridge} = \frac{1}{n}\sum_{i=1}^{n} \underbrace{((\mathbf{w}^\top \mathbf{x} + b) - y_i)^2}_{\text{squared loss}} + \lambda \underbrace{\|\mathbf{w}\|_2^2}_{l_2-regularizer}$$

(For SVM and Lasso it is OK to drop the bias term b, as this was also done in the notes.)

2. (2.5) Match the loss to the figure (just write A, ..., E next to each loss function's name.)
   - Zero-One Loss=E
   - Hinge Loss=A
   - Squared Hinge Loss=B
   - Logistic Loss=C
   - Exponential Loss=D

3. (3.5) Which loss functions in Q2 could you minimize with Gradient Descent without modifications? Which could you minimize with Newton's Method? Justify your claims if you cannot use an approach for a given loss functions (no justification needed if a method *does* apply).
   - **Gradient Descent**: Squared Hinge Loss, Logistic Loss, Exponential Loss (only these are **once** differentiable)
   - **Newton's Method**: Logistic Loss, Exponential Loss (only these are **twice** differentiable)

4. (1) You want to train an SVM to classify your data, but you also want to obtain a *sparse* weight vector. How would you modify the objective to achieve this goal? Substitute l_2 regularization with l_1 regularization.

5. You train a logistic regression classifier with gradient descent. Let $g(\vec{w})$ denote the gradient of the loss ℓ with respect to $\vec{w}$ (you don't need to compute it.)

a) (2) State the update you are making from w_t to w_{t+1}.

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w} - \alpha g(\mathbf{w})$$

b) (4) Under what assumptions does this update reduce the loss? Prove your claim. If $\alpha > 0$ is very small and the gradient is non-zero, then we can use Taylo's approximation to write

$$\ell(\mathbf{w}_{t+1}) = \ell(\mathbf{w} - \alpha g(\mathbf{w})) \approx \ell(\mathbf{w}) - \alpha g(\mathbf{w})^\top g(\mathbf{w}) < \ell(\mathbf{w}) \tag{1}$$

This holds because $\alpha > 0$ and $g(\mathbf{w})^\top g(\mathbf{w}) > 0$.

3 [15] Bias Variance

1. (3) Scrooge McDuck trains a classifier, trying to predict stock market prices. He trains decision trees of limited depth d (as he wants to reduce CPU time and electricity cost). However, soon he realizes that his training and his testing errors are both much too high for his system to be useful. Name **two** possible explanations for what could be the root of the problem. This could either be due to his classifier having **high bias** or his data having **high noise**.

2. (3) Scrooge now decides to no longer limit the tree depth. Instead he trains trees with unlimited depth. To his big disappointment he observes very similar behavior (train and test error are too high). What explanation can you give him? The only way **full decision trees** can have high training error is if there are many training inputs that are identical but differ in label. He therefore has a setting with **high noise.**

3. (6) What does Bagging simulate and why would it reduce variance? The variance of a classifier is
$$\mathbb{E}\left[(\bar{h} - h_D(\vec{x}))^2\right],$$
where the random variable h_D denotes the classifier resulting from a random training set D and $\bar{h} = \mathbb{E}\left[h_D\right]_D$. Bagging simulates averaging M classifiers that are trained on M different training sets $h_{D_1}, \ldots, h_{D_M}$ where $h_D = \frac{1}{M}\sum_{i=1}^{M} h_{D_i}$. By the **weak law of large numbers** if these data sets were i.i.d. we would have that $h_D \to \bar{h}$ as $M \to \infty$ and therefore the variance would go to zero.

4. (3) You boost decision trees with very limited depth ($depth = 2$). How are bias and variance affected as the boosting iterations increase. Bias decreases as the number of iterations increases. Variance will increase at some point, as the number of iterations becomes high.

4 [17] kNN / Curse of Dimensionality

1. (3) Kim K. uses kNN classification with the Euclidean distance, *i.e.* $dist(\vec{x}, \vec{z}) = \sqrt{\sum_{i=1}^{d}(\vec{x}_i - \vec{z}_i)^2}$. She has $n = 100000$ data points in her training set, each with $d = 50$ dimensions. She is frustrated, because during test-time her classifier is too slow. What would you recommend her to do in order to speed up her classifier during test time? She could use ball-trees, which would still work weelleven in 50 dimensions.

2. She completely ignores your advice and instead analyses the code herself. She realizes that most of her computation time is spent computing the $\sqrt{}$ operator. Annoyed by this wasted CPU power, she decides to use the *squared distance* instead, $[dist(\vec{x}, \vec{z})]^2$ (and drops all square-root computations). Although faster to compute, this squared Euclidean distance is no longer a metric, as it does not satisfy the triangular inequality.

   a) (3) Will this missing property affect her kNN accuracy? Explain why/ why not. No, it will not affect the accuracy as the neighborhood ordering is unchanged under the squared distances.

   b) (3) If she is determined to use the squared distance, is your answer to question 1 still valid? Explain why/why not. Yes, you can no longer use ball-trees, as their pruning depends on the triangular inequality.

3. (3) Kanye W. wants to use the kNN classifier on images of dresses to classify them as either *white with gold stripes* (+1) or *blue with black stripes* (-1). The data set is rather high dimensional $d = 1000000$ (each feature is a pixel) and he only has about $n = 5000$ images. Should he be worried about the curse of dimensionality? Explain why/why not. No, he shouldn't, as the data set is intrinsicly low dimensional.

4. (2) Explain how the neighborhood size k affects the classifier's **bias** and **variance**. If k is small, the classifier has high variance. If k is large, it increases in bias and reduces in variance.

5. (3) Provide one scenario in which you would want to use a kNN classifier instead of a linear SVM classifier and vice versa. You might want to use kNN if you need to model your data with non-linear decision boundaries. You want to use SVM on very high dimensional text data, where a linear classifier is sufficient and much faster.

5 [9] Decision Trees

1. (3) What is the prediction value at a leaf of a regression tree (with squared-loss impurity)? Prove that this is optimal.
   Predict the average label $\bar{y}$
   Given the squared loss over a set S:

$$\begin{aligned}
L(S, h(S)) &= \frac{1}{|S|} \sum_{(x,y)\in S} (y - h(x))^2 \\
&= \frac{1}{|S|} \sum_{(x,y)\in S} y^2 - 2yh(x) + h(x)^2 \\
\frac{dL(S,h(S))}{dh(x)} &= \frac{1}{|S|} \sum_{(x,y)\in S} -2y + 2h(x) \\
0 = \frac{dL(S,h(S))}{dh(x)} &= \frac{1}{|S|} \sum_{(x,y)\in S} -2y + 2h(x) \\
0 &= \frac{1}{|S|} \sum_{(x,y)\in S} -y + h(x) \\
&= \frac{1}{|S|} \sum_{(x,y)\in S} (-y) + \frac{1}{|S|}|S|h(x) \\
0 &= -\bar{y} + h(x) \\
\bar{y} &= h(x) \\
\frac{dL(S,h(S))}{dh(x)^2} &= 2
\end{aligned}$$

   Therefore $\bar{y} = h(x)$ minimizes the squared loss

2. (2) Given the definition of the Gini Index of a set:

$$G(S) = \sum_{k=1}^{c} p_k(1 - p_k) \tag{2}$$

   where c is the total number of classes, and p_k is the probability that an item of set S is of class k. In the situation where there are 3 classes, when is G(S) maximized, when minimized? (no derivation necessary)?
   G(S) is maximized when there is an equal probability of any of the 3 classes being in the set (i.e. they all appear in equal proportion), G(S) is minimized when the probability of a single class equals 1, and the probibility of any other class is 0 (i.e. there is only items of a single class in S)

3. (2) Explain in what sense decision trees are "myopic".
   Decision trees are myopic because it greedily tries to reduce the impurity of the set, and once it has done so, totally ignores examples on a different side of the decision boundary.

4. (2) What are two methods of preventing over-fitting in decision trees?
   Limiting the max depth, or pruning the tree on a validation set are two methods

6 [15] Boosting Bagging

1. (4) Ludwig van Beethoven claims that when he trains a Random Forests classifier he does not need any validation or test data and can train the classifier on the entire data set, even to select model parameters? Is this true/false? Justify your answer. Yes, he can use the out-of-bag inputs as validation set for each tree.

2. (5) What is the loss function that Adaboost minimizes? Prove that it is an upper bound on the training error.

   Adaboost minmizes the **exponential loss**.

$$\ell = \frac{1}{n}\sum_{i=1}^{n} e^{-y_i H(\mathbf{x}_i)}$$

   If $y_i H(\mathbf{x}_i > 0$ then $e^{-y_i H(\mathbf{x}_i)} > 0$, and if $y_i H(\mathbf{x}_i \leq 0$ then $e^{-y_i H(\mathbf{x}_i)} \geq 1$, and if It follows that

$$\mathrm{E}_{tr} = \frac{1}{n}\left[\sum_{i:y_i H(\mathbf{x}_i)\geq 0} 1 + \sum_{i:y_i H(\mathbf{x}_i)<0} 0\right] \leq \frac{1}{n}\left[\sum_{i:y_i H(\mathbf{x}_i)\geq 0} e^{-y_i H(\mathbf{x}_i)} + \sum_{i:y_i H(\mathbf{x}_i)<0} e^{-y_i H(\mathbf{x}_i)}\right] = \ell$$

   (It is also OK to assume that $H(\mathbf{x_i}) \in \{+1, -1\}$).

3. (6) Boosting chooses the next weak learner $h \in \mathcal{H}$ to maximize the inner-product with the gradient of ℓ w.r.t. the ensemble classifier $H(\vec{x}) = \sum_{t=1}^{T} \alpha_t h_t(\vec{x})$

$$h = \text{argmax} \sum_{i=1}^{n} \frac{\partial \ell(H)}{\partial H(x_i)} h(x_i)$$

   In AdaBoost (with $y_i \in \{-1, 1\}$) this can be viewed as re-weighting the examples in each iteration. In Gradient Boosting (with $y \in \mathcal{R}$) this can be viewed as fitting the residual error with a squared loss. Derive **one of these two** updates. *(There is **no extra credit** for deriving both.)*

Adaboost:

$$h = \text{argmin} \sum_{i=1}^{n} -y_i h(x_i) e^{-y_i H(x_i)}$$
$$= \text{argmax} \sum_{i=1}^{n} w_i y_i h(x_i)$$

with $w_i = \frac{e^{-y_i H(x_i)}}{Z}$, where Z is a normalization constant.

$$y_i h(x_i) = \begin{cases} 1 & if y_i = h(x_i) \\ -1 & if y_i \neq h(x_i) \end{cases}$$

h is the classifier that maximizes weighted training accuracy!

GBRT: Assume $\sum_{i=1}^{n} h(x_i)^2$ is constant.

$$h = \text{argmin} \sum_{i=1}^{n} -2(y_i - H(x_i))h(x_i)$$

Let $r_i = (y_i - H(x_i))$ be the residual error of $H(x_i)$. Note: r_i is independent of $h(x_i)$.

$$h = \text{argmin} \sum_{i=1}^{n} r_i^2 - 2r_i h(x_i) + h(x_i)^2$$
$$= \text{argmin} \sum_{i=1}^{n} (r_i - h(x_i))^2$$

h is the classifier that minimizes the squared loss on the residual errors!