



CMSC726 Spring 2006: Bias-Variance Tradeoff and Ensemble Methods

readings: handed out in class
material from:
Tom Dietterich, <http://www.cs.orst.edu/~tgd>
Rich Maclin

Outline

- Bias-Variance Decomposition for Regression
- Bias-Variance Analysis of Learning Algorithms
- Ensemble Methods
- Effect of Bagging on Bias and Variance
- Effect of Boosting on Bias and Variance
- Summary and Conclusion

Intuition 1

- The goal in learning is not to learn an exact representation of the training data itself, but to build a statistical model of the process which generates the data. This is important if the algorithm is to have good generalization performance
- We saw that
 - models with too few parameters can perform poorly
 - models with too many parameters can perform poorly
- Need to optimize the complexity of the model to achieve the best performance
- One way to get insight into this tradeoff is the decomposition of generalization error into bias² + variance
 - a model which is too simple, or too inflexible, will have a large bias
 - a model which has too much flexibility will have high variance

Intuition

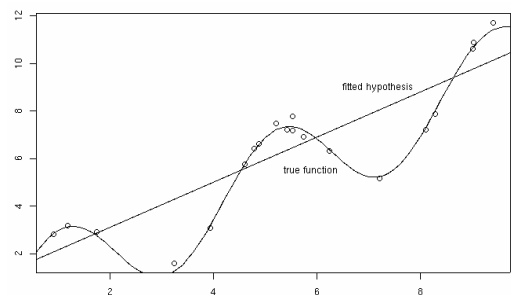
- bias:
 - measures the accuracy or quality of the algorithm
 - high bias means a poor match
- variance:
 - measures the precision or specificity of the match
 - a high variance means a weak match
- We would like to minimize each of these
- Unfortunately, we can't do this independently, there is a trade-off

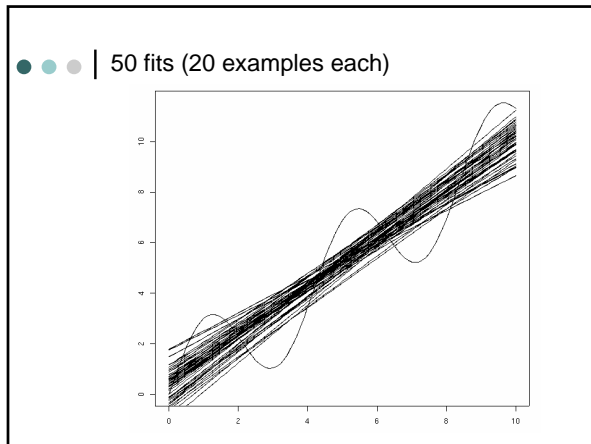
Bias-Variance Analysis in Regression

- True function is $y = f(x) + \epsilon$
 - where ϵ is normally distributed with zero mean and standard deviation σ .
- Given a set of training examples, $\{(x_i, y_i)\}$, we fit an hypothesis $h(x) = w \cdot x + b$ to the data to minimize the squared error

$$\sum_i [y_i - h(x_i)]^2$$

Example: 20 points
 $y = x + 2 \sin(1.5x) + N(0,0.2)$





● ● ● | Bias-Variance Analysis

- Now, given a new data point x^* (with observed value $y^* = f(x^*) + \epsilon$, we would like to understand the expected prediction error

$$E[(y^* - h(x^*))^2]$$

● ● ● | Classical Statistical Analysis

- Imagine that our particular training sample S is drawn from some population of possible training samples according to $P(S)$.
- Compute $E_p[(y^* - h(x^*))^2]$
- Decompose this into “bias”, “variance”, and “noise”

● ● ● | Lemma

- Let Z be a random variable with probability distribution $P(Z)$
- Let $\bar{Z} = E_p[Z]$ be the average value of Z .
- Lemma: $E[(Z - \bar{Z})^2] = E[Z^2] - \bar{Z}^2$

$$E[(Z - \bar{Z})^2] = E[Z^2 - 2Z\bar{Z} + \bar{Z}^2]$$

$$= E[Z^2] - 2E[Z]\bar{Z} + \bar{Z}^2$$

$$= E[Z^2] - 2\bar{Z}^2 + \bar{Z}^2$$

$$= E[Z^2] - \bar{Z}^2$$

- Corollary: $E[Z^2] = E[(Z - \bar{Z})^2] + \bar{Z}^2$

● ● ● | Bias-Variance-Noise Decomposition

$$E[(h(x^*) - y^*)^2] = E[h(x^*)^2 - 2h(x^*)y^* + y^{*2}]$$

$$= E[h(x^*)^2] - 2E[h(x^*)]E[y^*] + E[y^{*2}]$$

$$= E[(h(x^*) - \overline{h(x^*)})^2] + \overline{h(x^*)}^2 - 2\overline{h(x^*)}f(x^*) + E[(y^* - f(x^*))^2] + f(x^*)^2$$

$$= E[(h(x^*) - \overline{h(x^*)})^2] + \underbrace{(\overline{h(x^*)}^2 - f(x^*)^2)}_{\text{BIAS}} + \underbrace{E[(y^* - f(x^*))^2]}_{\text{NOISE}}$$

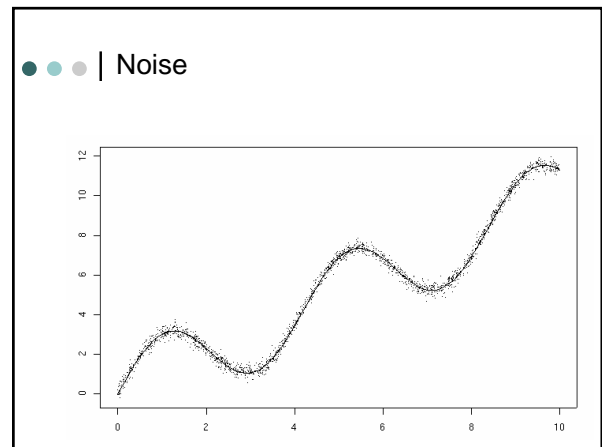
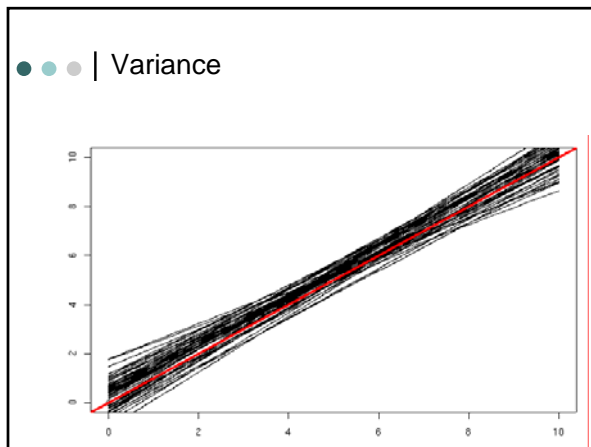
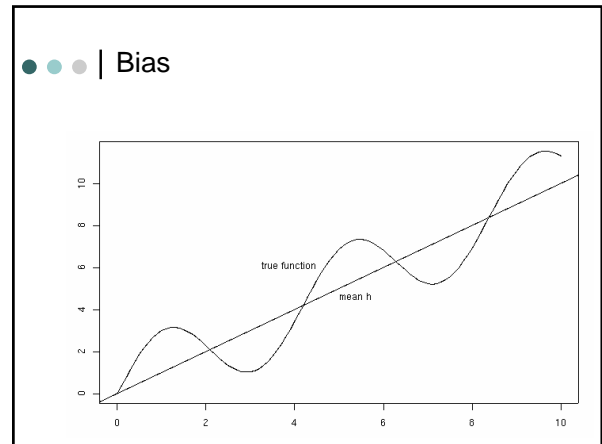
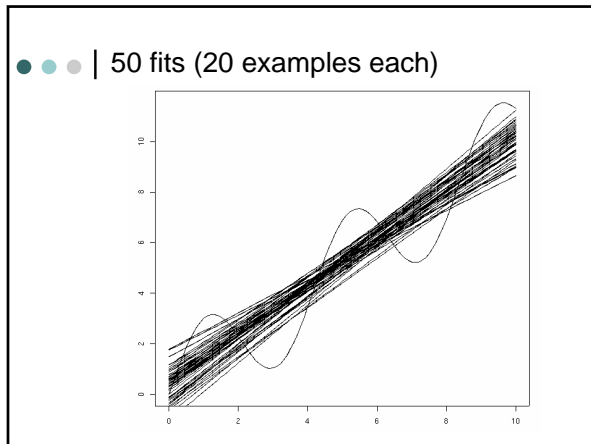
$$= \text{Var}(h(x^*)) + \text{Bias}(h(x^*))^2 + E[\epsilon^2]$$

$$= \text{Var}(h(x^*)) + \text{Bias}(h(x^*))^2 + \sigma^2$$

Expected prediction error = Variance + Bias² + Noise²

● ● ● | Bias, Variance, and Noise

- Variance: $E[(h(x^*) - \overline{h(x^*)})^2]$
Describes how much $h(x^*)$ varies from one training set S to another
- Bias: $[\overline{h(x^*)} - f(x^*)]$
Describes the average error of $h(x^*)$.
- Noise: $E[(y^* - f(x^*))^2] = E[\epsilon^2] = \sigma^2$
Describes how much y^* varies from $f(x^*)$



- ● ● | Measuring Bias and Variance
- In practice (unlike in theory), we have only ONE training set S .
 - We can simulate multiple training sets by bootstrap replicates
 - $S' = \{x \mid x \text{ is drawn at random with replacement from } S\}$ and $|S'| = |S|$.

- ● ● | Procedure for Measuring Bias and Variance
- Construct B bootstrap replicates of S (e.g., $B = 200$): S_1, \dots, S_B
 - Apply learning algorithm to each replicate S_b to obtain hypothesis h_b
 - Let $T_b = S \setminus S_b$ be the data points that do not appear in S_b (out of bag points)
 - Compute predicted value $h_b(x)$ for each x in T_b

● ● ● | Estimating Bias and Variance (continued)

- For each data point x , we will now have the observed corresponding value y and several predictions y_1, \dots, y_K .
- Compute the average prediction \bar{h} .
- Estimate bias as $(h - \bar{h})$
- Estimate variance as $\sum_k (y_k - \bar{h})^2 / (K - 1)$
- Assume noise is 0

● ● ● | Approximations in this Procedure

- Bootstrap replicates are not real data
- We ignore the noise
 - If we have multiple data points with the same x value, then we can estimate the noise
 - We can also estimate noise by pooling y values from nearby x values

● ● ● | Ensemble Learning

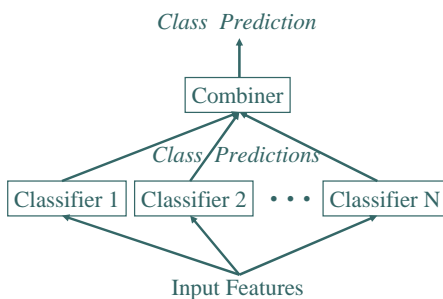
- what is an ensemble?
- why use an ensemble?
- selecting component classifiers
- selecting combining mechanism
- some results

● ● ● | Ensemble Learning Methods

- Given training sample S
- Generate multiple hypotheses, h_1, h_2, \dots, h_L .
- Optionally: determining corresponding weights w_1, w_2, \dots, w_L
- Classify new points according to

$$\sum_i w_i h_i > \theta$$

● ● ● | A Classifier Ensemble



● ● ● | Key Ensemble Questions

- Which components to combine?
- different learning algorithms
 - same learning algorithm trained in different ways
 - same learning algorithm trained the same way
- How to combine classifications?
- majority vote
 - weighted (confidence of classifier) vote
 - weighted (confidence in classifier) vote
 - learned combiner
- What makes a good (accurate) ensemble?

● ● ● | What Makes a Good Ensemble?

Krogh and Vedelsby, 1995

Can show that the accuracy of an ensemble is mathematically related:

$$\hat{E} = \bar{E} - D$$

\hat{E} is the error of the entire ensemble

\bar{E} is the average error of the component classifiers

D is a term measuring the diversity of the components

Effective ensembles have accurate and diverse components

● ● ● | Ensemble Mechanisms - Components

- Separate learning methods
 - not often used
 - very effective in certain problems (e.g., protein folding, Rost and Sander, Zhang)
- Same learning method
 - generally still need to vary something externally
 - exception, some good results with neural networks
 - most often, data set used for training varied:
 - Bagging (Bootstrap Aggregating), Breiman
 - Boosting, Freund & Schapire
 - Ada, Freund & Schapire
 - Arcing, Breiman

● ● ● | Ensemble Mechanisms - Combiners

- Voting
- Averaging (if predictions not 0,1)
- Weighted Averaging
 - base weights on confidence in component
- Learning combiner
 - Stacking, Wolpert
 - general combiner
 - RegionBoosting, Maclin
 - piecewise combiner

● ● ● | Bagging

Varies data set

Each training set a *bootstrap* sample

bootstrap sample - select set of examples (with replacement) from original sample

Algorithm:

for $k = 1$ to # of samples

$train' =$ bootstrap sample of train set

create classifier using $train'$ as training set

combine classifications using simple voting

● ● ● | Bagging: Bootstrap Aggregating

- For $b = 1, \dots, B$ do
 - $S_b =$ bootstrap replicate of S
 - Apply learning algorithm to S_b to learn h_b
- Classify new points by unweighted vote:
 - $[\sum_b h_b(x)]/B > 0$

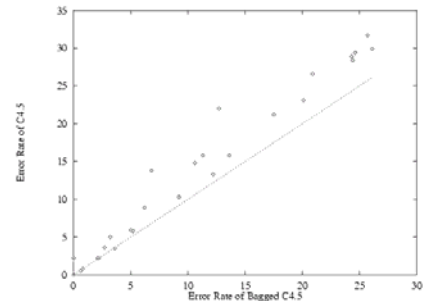
● ● ● | Bagging

- Bagging makes predictions according to $y = \sum_b h_b(x) / B$
- Hence, bagging's predictions are $h(x)$

Estimated Bias and Variance of Bagging

- If we estimate bias and variance using the same B bootstrap samples, we will have:
 - Bias = $(h - y)$ [same as before]
 - Variance = $\sum_k (y - h)^2 / (K - 1) = 0$
- Hence, according to this approximate way of estimating variance, bagging removes the variance while leaving bias unchanged.
- In reality, bagging only *reduces* variance and tends to slightly increase bias

Bagging Decision Trees (Freund & Schapire)



Bias/Variance Heuristics

- Models that fit the data poorly have high bias: "inflexible models" such as linear regression, regression stumps
- Models that can fit the data very well have low bias but high variance: "flexible" models such as nearest neighbor regression, regression trees
- This suggests that bagging of a flexible model can reduce the variance while benefiting from the low bias

Weak Learning

Schapire showed that a set of weak learners (learners with $> 50\%$ accuracy, but not much greater) could be combined into a strong learner

Idea: weight the data set based on how well we have predicted data points so far

- data points predicted accurately - low weight
- data points mispredicted - high weight

Result: focuses components on portion of data space not previously well predicted

Boosting - Ada

Varies weights on training data

Algorithm:

- for each data points: weight w_i to 1..#datapoints
- for $k = 1$ to #classifiers
 - generate $classifier_k$ with current weighted train set
 - ϵ_k = weighted sum of w_i 's of misclassified points
 - $\beta_k = (1 - \epsilon_k) / \epsilon_k$
 - multiply weights of all misclassified points by β_k
 - normalize weights to sum to 1
- combine: weighted vote, weight for $classifier_k$ is $\log(\beta_k^{-1})$

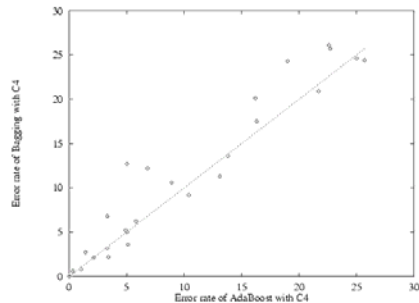
Boosting

Input: a set S , of m labeled examples: $S = \{(x_i, y_i), i = 1, 2, \dots, m\}$,
 labels $y_i \in Y = \{1, \dots, K\}$
 Learn (a learning algorithm)
 a constant L .

- [1] initialize for all i : $w_i(i) := 1/m$ initialize the weights
- [2] for $\ell = 1$ to L do
- [3] for all i : $p_i(i) := w_i(i) / (\sum_i w_i(i))$ compute normalized weights
- [4] $h_\ell := \text{Learn}(p_i)$ call Learn with normalized weights.
- [5] $\epsilon_\ell := \sum_i p_i(i) [h_\ell(x_i) \neq y_i]$ calculate the error of h_ℓ
- [7] if $\epsilon_\ell > 1/2$ then
- [8] $L := \ell - 1$
- [9] exit
- [10] $\beta_\ell := \epsilon_\ell / (1 - \epsilon_\ell)$
- [11] for all i : $w_{\ell+1}(i) := w_\ell(i) \beta_\ell^{1 - [h_\ell(x_i) \neq y_i]}$ compute new weights

Output: $h_f(x) = \text{argmax}_{y \in Y} \sum_{\ell=1}^L \left(\log \frac{1}{\beta_\ell} \right) [h_\ell(x) = y]$

● ● ● | Boosting vs Bagging (Freund & Schapire)



● ● ● | Boosting - Arcing

Sample data set (like Bagging), but probability of data point being chosen weighted (like Boosting)

m_i = #number of mistakes made on point i by previous classifiers

probability of selecting point i :

$$prob_i = \frac{1 + m_i^4}{\sum_{j=0}^N 1 + m_j^4}$$

Value 4 chosen empirically

Combine using voting

● ● ● | Some Results - BP, C4.5 Components

Dataset	C4.5	BP	BagC4	BagBP	AdaC4	AdaBP	ArcC4	ArcBP
letter	14.0	18.0	7.0	10.5	4.1	5.7	3.9	4.6
segment	3.7	6.6	3.0	5.4	1.7	3.5	1.5	3.3
promoter	12.8	5.3	10.6	4.0	6.8	4.5	6.4	4.6
kr-vs-kp	0.6	2.3	0.6	0.8	0.3	0.4	0.4	0.3
splice	5.9	4.7	5.4	3.9	5.1	4.0	5.3	4.2
breastc	5.0	3.4	3.7	3.4	3.5	3.8	3.5	4.0
housev	3.6	4.9	3.6	4.1	5.0	5.1	4.8	5.3

● ● ● | Some Theories on Bagging/Boosting

Error = noise error + Bias + Variance

Theories:

Bagging can reduce variance part of error

Boosting can reduce variance AND bias part of error

Bagging will hardly ever increase error

Boosting may increase error

Boosting susceptible to noise

Boosting increases margins

● ● ● | Combiner - Stacking

Idea:

generate component (level 0) classifiers with part of the data (half, three quarters)

train combiner (level 1) classifier to combine predictions of components using remaining data

retrain component classifiers with all of training data

In practice, often equivalent to voting

● ● ● | Combiner - RegionBoost

- Train "weight" classifier for each component classifier
- "weight" classifier predicts how likely point will be predicted correctly
- "weight" classifiers: k-Nearest Neighbor, Backprop
- Combiner, generate component classifier prediction and weight using corresponding "weight" classifier
- Small gains in accuracy

● ● ● | Sources of Bias and Variance

- Bias arises when the classifier cannot represent the true function – that is, the classifier underfits the data
- Variance arises when the classifier overfits the data
- There is often a tradeoff between bias and variance

● ● ● | Effect of Algorithm Parameters on Bias and Variance

- k-nearest neighbor: increasing k typically increases bias and reduces variance
- decision trees of depth D: increasing D typically increases variance and reduces bias
- RBF SVM with parameter σ : increasing σ increases bias and reduces variance

● ● ● | Effect of Bagging

- If the bootstrap replicate approximation were correct, then bagging would reduce variance without changing bias
- In practice, bagging can reduce both bias and variance
 - For high-bias classifiers, it can reduce bias
 - For high-variance classifiers, it can reduce variance

● ● ● | Effect of Boosting

- In the early iterations, boosting is primarily a bias-reducing method
- In later iterations, it appears to be primarily a variance-reducing method

● ● ● | Other Approaches

- Error Correcting Output Codes
- Mixture of Experts
- Cascading Classifiers
- many others...