Applied inductive learning

Ensemble methods and Feature selection

Pierre Geurts
Department of Electrical Engineering and Computer Science
University of Liège

November 2017

Reminder: bias/variance decomposition

$$E_{LS}\{E_{y|\underline{x}}\{(y-\hat{y}(\underline{x}))^2\}\} = \text{noise}(\underline{x}) + \text{bias}^2(\underline{x}) + \text{variance}(\underline{x})$$

- $\text{noise}(\underline{x}) = E_{y|\underline{x}}\{(y-h_B(\underline{x}))^2\}$

  Quantifies how much y varies from $h_B(\underline{x}) = E_{y|\underline{x}}\{y\}$, the Bayes model.
- $\text{bias}^2(\underline{x}) = (h_B(\underline{x}) - E_{LS}\{\hat{y}(\underline{x})\})^2$

  Measures the error between the Bayes model and the average model.
- $\text{variance}(\underline{x}) = E_{LS}\{(\hat{y} - E_{LS}\{\hat{y}(\underline{x})\})^2\}$

  Quantify how much $\hat{y}(\underline{x})$ varies from one learning sample to another.

Reminder: Bias and variance reduction techniques

- In the context of a given method:
  - Adapt the learning algorithm to find the best trade-off between bias and variance.
  - Not a panacea but the least we can do.
  - Example: pruning, weight decay.
- Ensemble methods:
  - Change the bias/variance trade-off.
  - Universal but destroys some features of the initial method.
  - Example: bagging, boosting.

Ensemble methods

- Combine the predictions of several models built with a learning algorithm in order to improve with respect to the use of a single model
- Two main families:
  - Averaging techniques
    - Grow several models independently and simply average their predictions
    - Ex: bagging, random forests
    - Decrease mainly variance
  - Boosting type algorithms
    - Grows several models sequentially
    - Ex: Adaboost, MART
    - Decrease mainly bias

Bagging (1)

- Suppose that we can generate several learning samples from the original data distribution $P(\underline{x}, y)$
- Let us study the following algorithm:
  - Draw *T* learning samples $\{LS_1, LS_2, \ldots, LS_T\}$
  - Learn a model $\hat{y}_{LS_i}$ from each LS_i
  - Compute the average model

$$\hat{y}_{ens}(\underline{x}) = \frac{1}{T}\sum_{i=1}^{T}\hat{y}_{LS_i}(\underline{x})$$

- How do the bias and variance of this algorithm relate to that of the original algorithm (building $\hat{y}_{LS}$ from one LS)

Bagging (2)

$$E_{LS}\{\mathrm{Err}(\underline{x})\} = E_{y|\underline{x}}\{(y - h_B(\underline{x}))^2\} + (h_B(\underline{x}) - E_{LS}\{\hat{y}(\underline{x})\})^2 + E_{LS}\{(\hat{y}(\underline{x}) - E_{LS}\{\hat{y}(\underline{x})\})^2\}$$

- Its bias is the same as the original algorithm

$$\begin{aligned} E_{LS_1,\ldots,LS_T}\{\hat{y}_{ens}(\underline{x})\} &= \frac{1}{T}\sum_i E_{LS_i}\{\hat{y}_{LS_i}(\underline{x})\} \\ &= E_{LS}\{\hat{y}_{LS}(\underline{x})\} \end{aligned}$$

- Variance is divided by T

$$\begin{aligned} &E_{LS_1,\ldots,LS_T}\{(\hat{y}_{ens}(\underline{x}) - E_{LS_1,\ldots,LS_T}\{\hat{y}_{ens}(\underline{x})\})^2\} \\ &\qquad = \frac{1}{T} E_{LS}\{(\hat{y}(\underline{x}) - E_{LS}\{\hat{y}_{ens}(\underline{x})\})^2\} \end{aligned}$$

$\Rightarrow$mean square error decreases

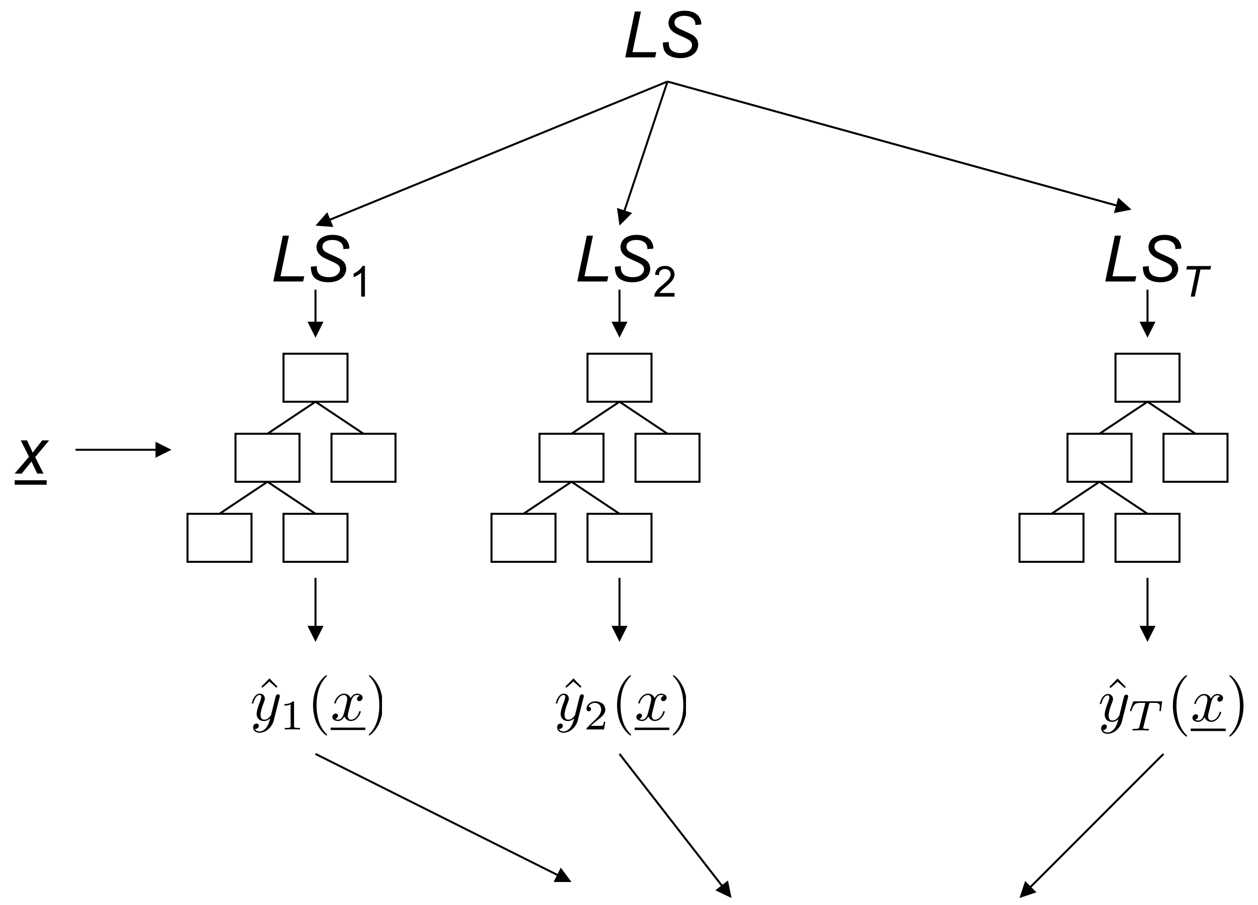
Bagging (3)

- In practice one can not drawn several LS ($P(\underline{x}, y)$ is unknown)
- Idea: use bootstrap sampling to generate several learning samples
- Bagging (bootstrap aggregating):
  - Drawn *T* bootstrap samples $\{B_1, B_2, \ldots, B_T\}$ from LS
  - Learn a model $\hat{y}_{B_i}$ from each B_i
  - Build the average model

$$\hat{y}_{ens}(\underline{x}) = \frac{1}{T} \sum_{i=1}^{T} \hat{y}_{B_i}(\underline{x})$$

- Variance is reduced but bias increases a bit (because the effective size of a bootstrap sample is about 30% smaller than the original LS)

Bagging (4)



LS
LS$_1$
LS$_2$
LS$_T$
$\underline{x}$
$\hat{y}_1(\underline{x})$
$\hat{y}_2(\underline{x})$
$\hat{y}_T(\underline{x})$

In regression: $\hat{y}(\underline{x}) = 1/k(\hat{y}_1(\underline{x}) + \ldots + \hat{y}_T(\underline{x}))$

In classification: $\hat{y}(\underline{x})$ = the majority class in $\{\hat{y}_1(\underline{x}), \ldots, \hat{y}_T(\underline{x})\}$

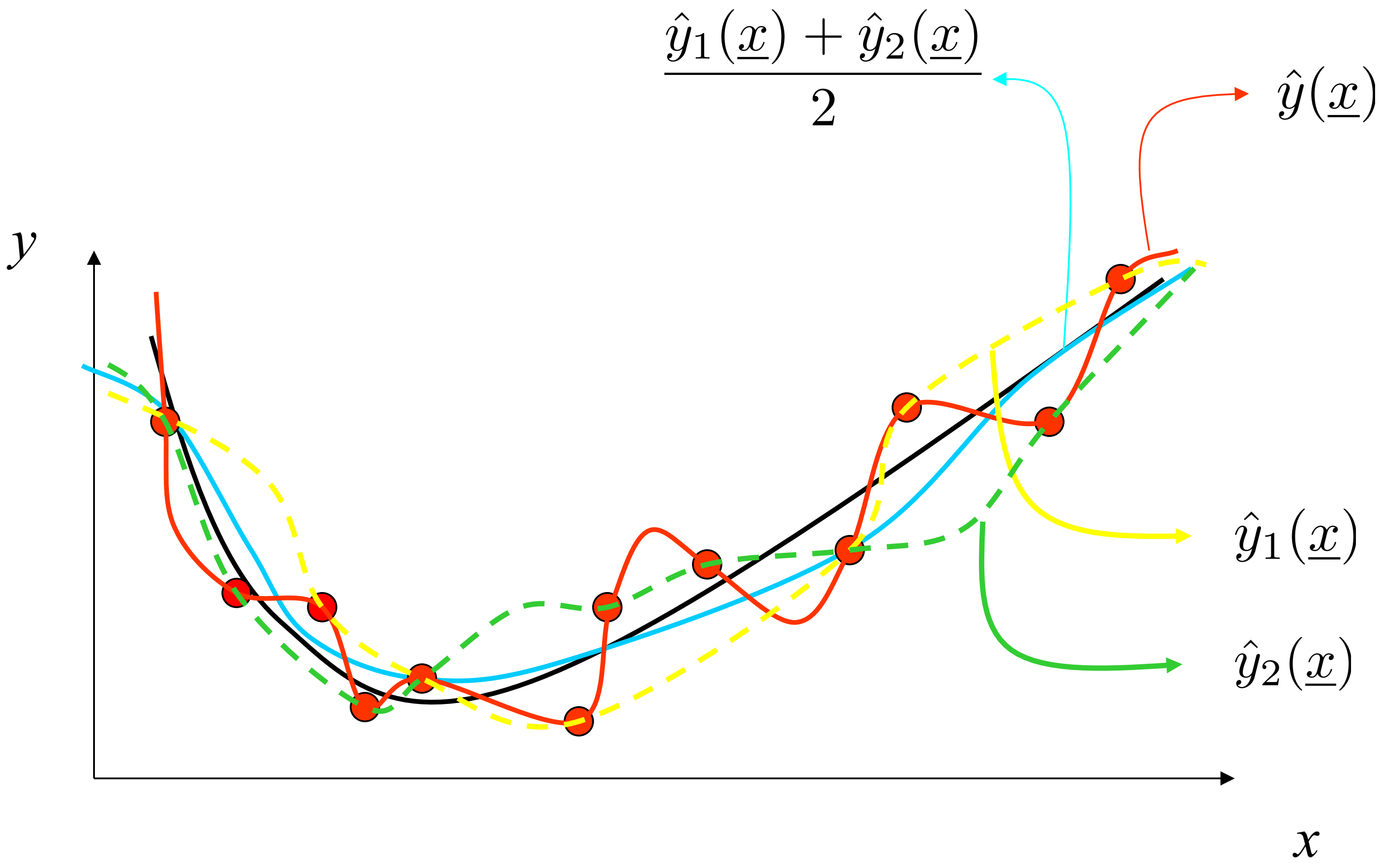
Bagging (5)

- Usually, bagging reduces very much the variance without increasing too much the bias.
- Application to regression trees (on Friedman's problem)

Method	E	Bias	Variance
3 Test regr. Tree	14.8	11.1	3.7
Bagged (T=25)	11.7	10.7	1.0
Full regr. Tree	10.2	3.5	6.7
Bagged (T=25)	5.3	3.8	1.5

- Strong variance reduction without increasing the bias (although the model is much more complex than a single tree)

Bagging (4)


$\frac{\hat{y}_1(\underline{x}) + \hat{y}_2(\underline{x})}{2}$
$\hat{y}(\underline{x})$
y
$\hat{y}_1(\underline{x})$
$\hat{y}_2(\underline{x})$
x

Other averaging techniques

- Perturb and Combine paradigm:
  - Perturb the data or the learning algorithm to obtain several models that are good on the learning sample.
  - Combine the predictions of these models
- Usually, these methods decrease the variance (because of averaging) but (slightly) increase the bias (because of the perturbation)
- Examples:
  - Bagging perturbs the learning sample.
  - Learn several neural networks with random initial weights

Method	E	Bias	Variance
MLP (10-10)	4.6	1.4	3.2
Average of 10 MLPs	2.0	1.4	0.6

  - Random forests.

Random forests (1)

- Perturb and combine algorithm specifically designed for trees
- Combine bagging and random attribute subset selection:
  - Build the tree from a bootstrap sample
  - Instead of choosing the best split among all attributes, select the best split among a random subset of k attributes

  (= bagging when k is equal to the number of attributes)
- There is a bias/variance tradeoff with k: The smaller k, the greater the reduction of variance but also the higher the increase of bias

Random forests (2)

- Application to our illustrative problem:

Method	E	Bias	Variance
Full regr. Tree	10.2	3.5	6.7
Bagging (k=10)	5.3	3.8	1.5
Random Forests (k=7)	4.8	3.8	1.0
Random Forests (k=5)	4.9	4.0	0.9
Random Forests (k=3)	5.6	4.7	0.8

- Other advantage: it decreases computing times with respect to bagging since only a subset of all attributes needs to be considered when splitting a node.

Ambiguity decomposition

(Krogh and Vedelsby, 1995)

Let us assume *T* models $\{\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_T\}$ and their average

$$\hat{y}_{ens}(\underline{x}) = \frac{1}{T}\sum_i \hat{y}_i(\underline{x})$$

We have the following decomposition:

$$\frac{1}{T}\sum_i E_{y|\underline{x}}\{(y - \hat{y}_i(\underline{x}))^2\} = E_{y|\underline{x}}\{(y - \hat{y}_{ens}(\underline{x}))^2\} + \frac{1}{T}\sum_i (y_i(\underline{x}) - \hat{y}_{ens}(\underline{x}))^2$$

Meaning that

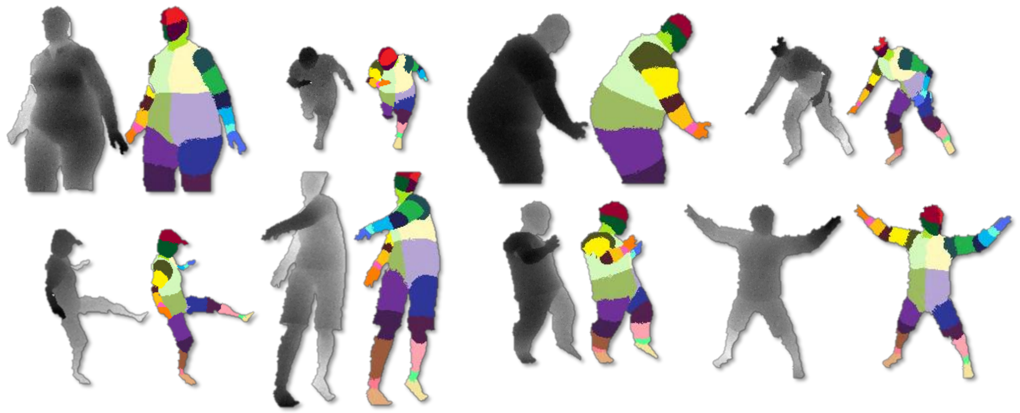
$$E_{y|\underline{x}}\{(y - \hat{y}_{ens}(\underline{x}))^2\} = \frac{1}{T}\sum_i E_{y|\underline{x}}\{(y - \hat{y}_i(\underline{x}))^2\} - \frac{1}{T}\sum_i (y_i(\underline{x}) - \hat{y}_{ens}(\underline{x}))^2$$

$\Rightarrow$ The average model is always better than the individual models in the mean

(This is not true in classification)

Application: Kinect

- Ensemble of randomized decision trees are used in Microsoft's Xbox Kinect for body part labeling:



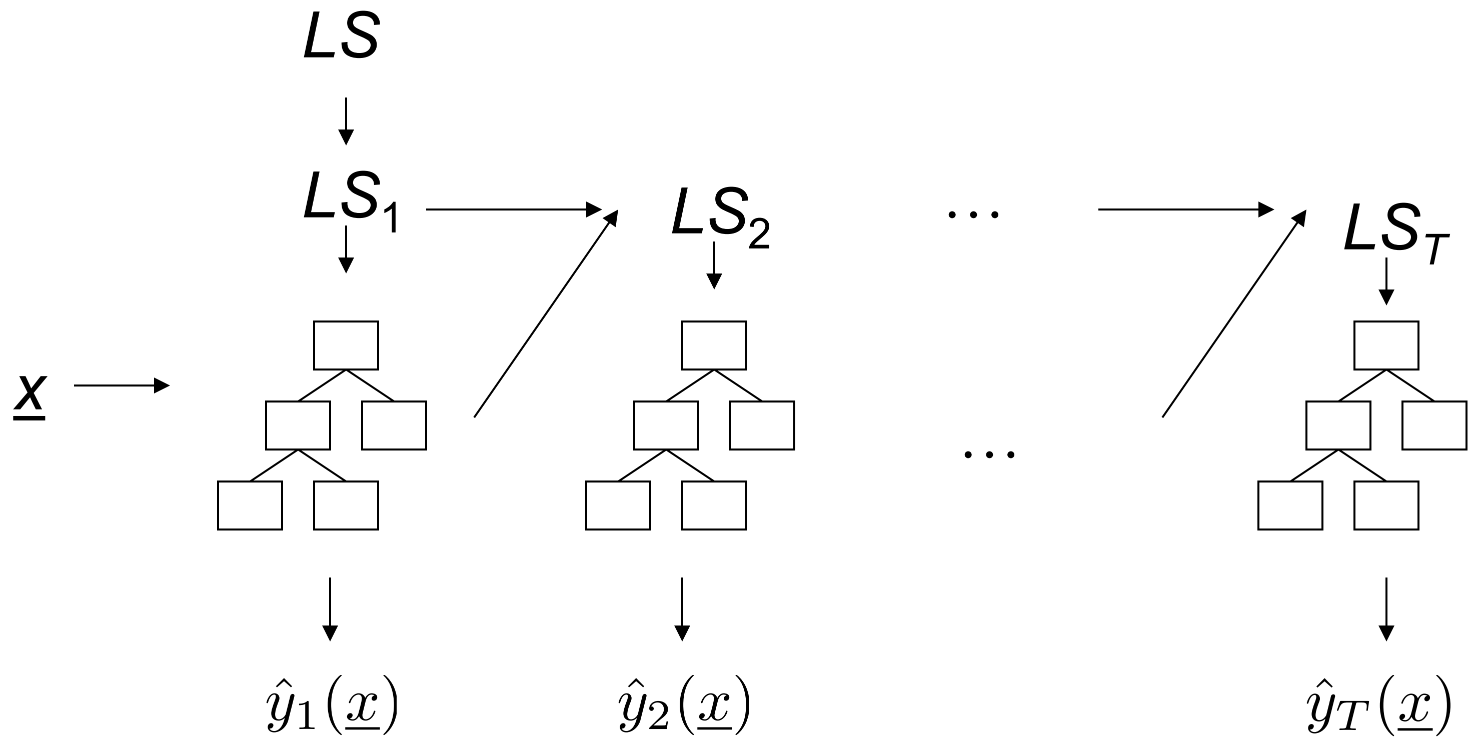
- Each sample corresponds to a single pixel and is described by depth differences between neighbor pixels
- Final model is implemented on GPU to get very fast predictions (200 frames per second)

http://research.microsoft.com/pubs/145347/BodyPartRecognition.pdf

Boosting methods (1)

- The motivation of boosting is to combine the ouputs of many « weak » models to produce a powerful ensemble of models.
- Weak model = a model that has a high bias (strictly, in classification, a model slightly better than random guessing)
- Differences with previous ensemble methods:
  - Models are built sequentially on modified versions of the data
  - The predictions of the models are combined through a weighted sum/vote

Boosting methods (2)


LS
LS_1
LS_2
$\cdots$
LS_T
$\underline{x}$
$\cdots$
$\hat{y}_1(\underline{x})$
$\hat{y}_2(\underline{x})$
$\hat{y}_T(\underline{x})$

In regression: $\hat{y}(\underline{x}) = \beta_1 \hat{y}_1(\underline{x}) + \beta_2 \hat{y}_2(\underline{x}) + \ldots + \beta_T \hat{y}_T(\underline{x})$

In classification: $\hat{y}(\underline{x})$ = the majority class in $\{\hat{y}_1(\underline{x}), \ldots, \hat{y}_T(\underline{x})\}$ according to the weights $\{\beta_1, \ldots, \beta_T\}$

Adaboost (1)

- Assume that the learning algorithm accepts weighted objects

$$\{(x_1, y_1, w_1), (x_2, y_2, w_2), \ldots, (x_N, y_N, w_N)\}$$

- This is the case of many learning algorithms:
  - With trees, simply take into account the weights when counting objects
  - In neural networks, minimize the weighted squared error
- At each step, adaboost increases the weights of cases from the learning sample misclassified by the last model
- Thus, the algorithm focuses on the difficult cases from the learning sample
- In the weighted majority vote, adaboost gives higher influence to the more accurate models

Adaboost (2)

- Input: a learning algorithm and a learning sample
$$\{(x_i, y_i) : i = 1, \ldots, N\}$$
- Initialize the weights $w_i = 1/N, i = 1, \ldots, N$
- For *t*=1 to *T*
  - Build a model $\hat{y}_t(x)$ with the learning algorithm using weights *w*$_i$
  - Compute the weighted error:
$$\text{err}_t = \frac{\sum_i w_i I(y_i \neq \hat{y}_t(x_i))}{\sum_i w_i}$$
  - Compute $\beta_t = \log((1 - \text{err}_t)/\text{err}_t))$
  - Change weights according to
$$w_i \leftarrow w_i \exp[\beta_t I(y_i \neq \hat{y}_t(x_i))]$$
  - Normalize them so that $\sum_i w_i = 1$

Adaboost (3)

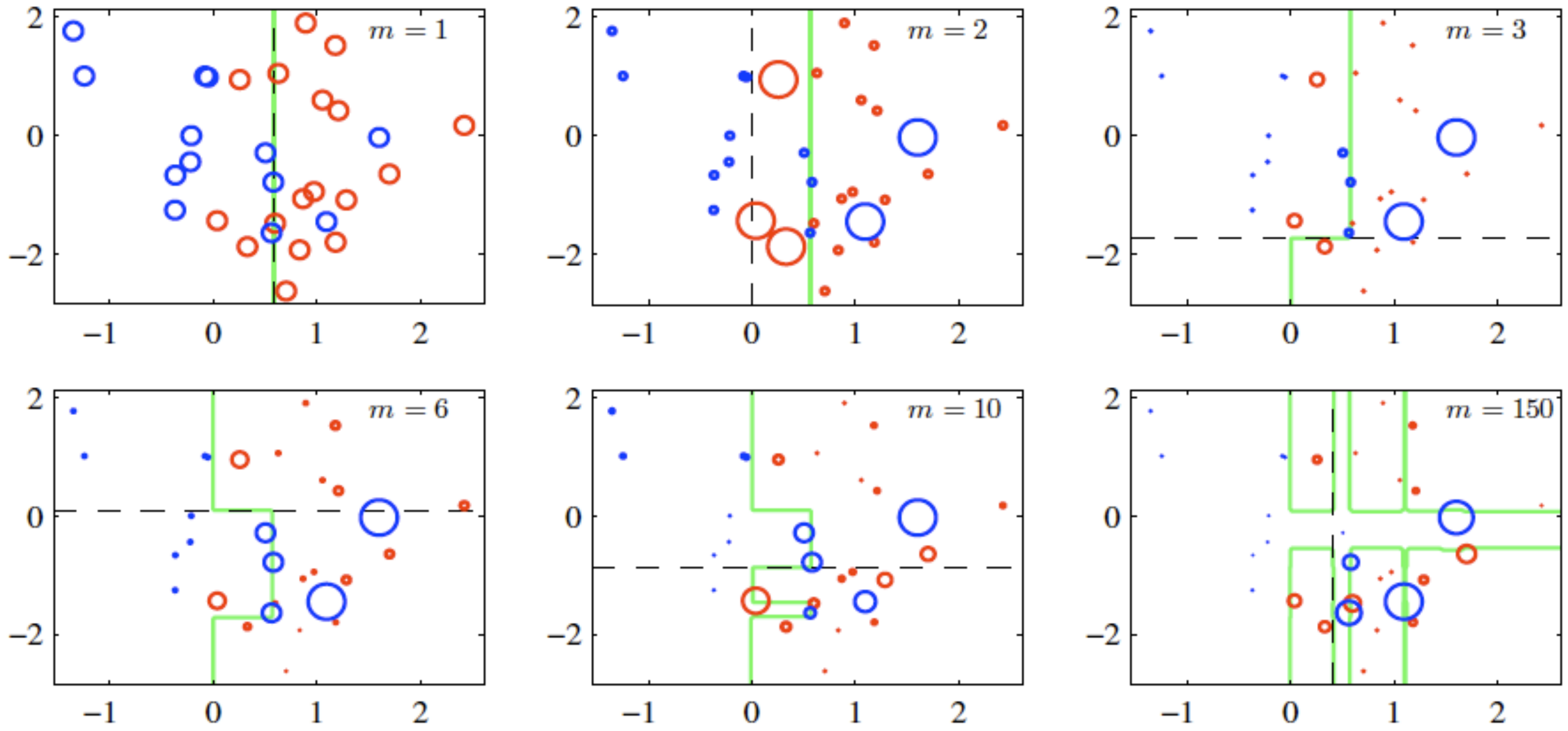


Figure 14.2 Illustration of boosting in which the base learners consist of simple thresholds applied to one or other of the axes. Each figure shows the number m of base learners trained so far, along with the decision boundary of the most recent base learner (dashed black line) and the combined decision boundary of the ensemble (solid green line). Each data point is depicted by a circle whose radius indicates the weight assigned to that data point when training the most recently added base learner. Thus, for instance, we see that points that are misclassified by the $m = 1$ base learner are given greater weight when training the $m = 2$ base learner.

(Bishop, p. 660)

Least squares boosting

(a boosting algorithm for regression)

- Input: a learning sample $\{(x_i, y_i) : i = 1, \ldots, N\}$
- Initialize

$$\hat{y}_0(x) = 1/N \sum_i y_i;\ r_i = y_i, i = 1, \ldots, N$$

- For *t*=1 to *T*:
  - For *i*=1 to *N*, compute the residuals

$$r_i \leftarrow r_i - \hat{y}_{t-1}(x_i)$$

  - Build a regression tree from the learning sample

$$\{(x_i, r_i) : i = 1, \ldots, N\}$$

- Return the model

$$\hat{y}(x) = \hat{y}_0(x) + \hat{y}_1(x) + \ldots + \hat{y}_T(x)$$

A generic boosting algorithm

- Goal: Find $\hat{y}(\underline{x}) = \beta_1 \hat{y}_1(\underline{x}) + \beta_2 \hat{y}_2(\underline{x}) + \ldots + \beta_T \hat{y}_T(\underline{x})$
  that minimizes $\sum_{i=1}^{N} L(y_i, \hat{y}(\underline{x_i}))$
- Forward stage-wise additive modeling:
  1. Initialize $\hat{y}(\underline{x}) = 0$
  2. For *t*=1 to *T*:
     a) Compute $(\beta_t, \hat{y}_t) = \arg\min_{\beta, \hat{y}'} \sum_i L(y_i, \hat{y}(\underline{x_i}) + \beta \hat{y}'(\underline{x_i}))$
     b) Set $\hat{y}(\underline{x}) \leftarrow \hat{y}(\underline{x}) + \beta_t \hat{y}_t(\underline{x})$
- Examples:
  – $L(y, y') = (y - y')^2 \Rightarrow$ Least squares boosting
  – $L(y, y') = \exp(-yy') \Rightarrow$ Adaboost *(try to prove it)*

A note about loss functions

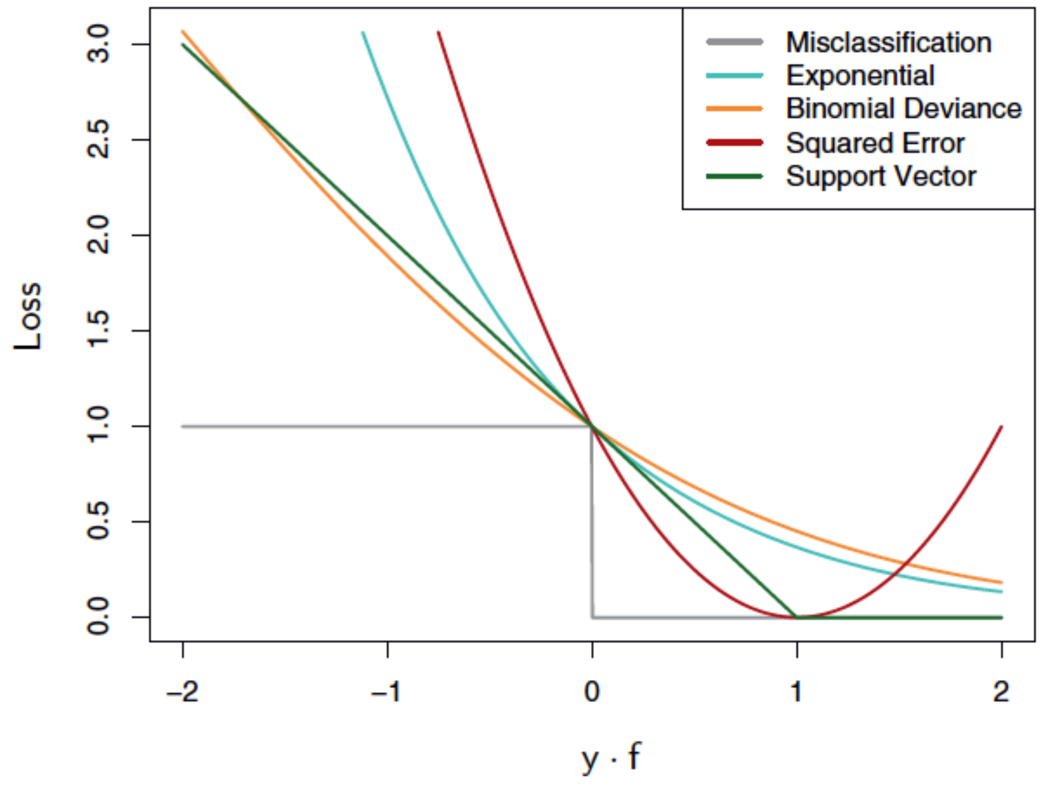


FIGURE 10.4. *Loss functions for two-class classification. The response is* $y = \pm 1$*; the prediction is* f*, with class prediction* $\text{sign}(f)$*. The losses are misclassification:* $I(\text{sign}(f) \neq y)$*; exponential:* $\exp(-yf)$*; binomial deviance:* $\log(1 + \exp(-2yf))$*; squared error:* $(y - f)^2$*; and support vector:* $(1 - yf)_+$ *(see Section 12.3). Each function has been scaled so that it passes through the point* $(0, 1)$.

Boosting methods

- There are many other types of boosting algorithms (eg. gradient boosting)
- Boosting decision/regression trees improves their accuracy often dramatically. However, boosting is more sensitive to noise than averaging techniques (overfitting).
- For boosting to work, the models need not to be perfect on the learning sample. With trees, there are two possible strategies:
  - Use pruned trees (pre-pruned or post-pruned by cross-validation)
  - Limit the number of tree tests (and split first the most impure nodes)
- $\Rightarrow$ there is again a bias/variance tradeoff with respect to the tree size.

Experiment with MART (=Least-squares boosting)

- On our illustrative problem:

Method	E	Bias	Variance
Full regr. Tree	10.2	3.5	6.7
Regr. Tree with 1 test	18.9	17.8	1.1
+ MART (T=50)	5.0	3.1	1.9
+ Bagging (T=50)	17.9	17.3	0.6
Regr. Tree with 5 tests	11.7	8.8	2.9
+ MART (T=50)	6.4	1.7	4.7
+ Bagging (T=50)	9.1	8.7	0.4

- Boosting reduces the bias but increases the variance. However, with respect to full trees, it decreases both bias and variance.

Other ensemble approaches (1)

- Bayesian model averaging

$$P(y|x, LS) = \sum_{h\in\mathcal{H}} P(y|h, LS)P(h|LS)$$

Prior knowledge about models (e.g., simple models are more probable)

Quality of the fit

$$\begin{aligned} P(h|LS) &\propto \overset{\downarrow}{P(h)}\,\overset{\downarrow}{P(LS|h)} \\ &\propto P(h)\sum_{\theta} P(LS|\theta, h)P(\theta|h) \end{aligned}$$

Other ensemble approaches (2)

- Stacking:

  learn a model to combine the models

  - Let $LS = \{(x_i, y_i) | i = 1, \ldots, N\}$
  - Let $A^t, t = 0, \ldots, T$ be *T*+1 learning algorithms
  - For $t = 1, \ldots, T$ do
    - Construct a model: $\hat{y}^t = A^t(LS)$
    - Compute predictions: $\hat{y}_i^t = \hat{y}^t(x_i)$
  - Set $LS^0 = \{(x_i^0, y_i)\}$ with $x_i^0 = (y_i^t)_{t=1}^T$
  - Return $\hat{y} = A^0(LS^0)$

Interpretability and efficiency of ensembles

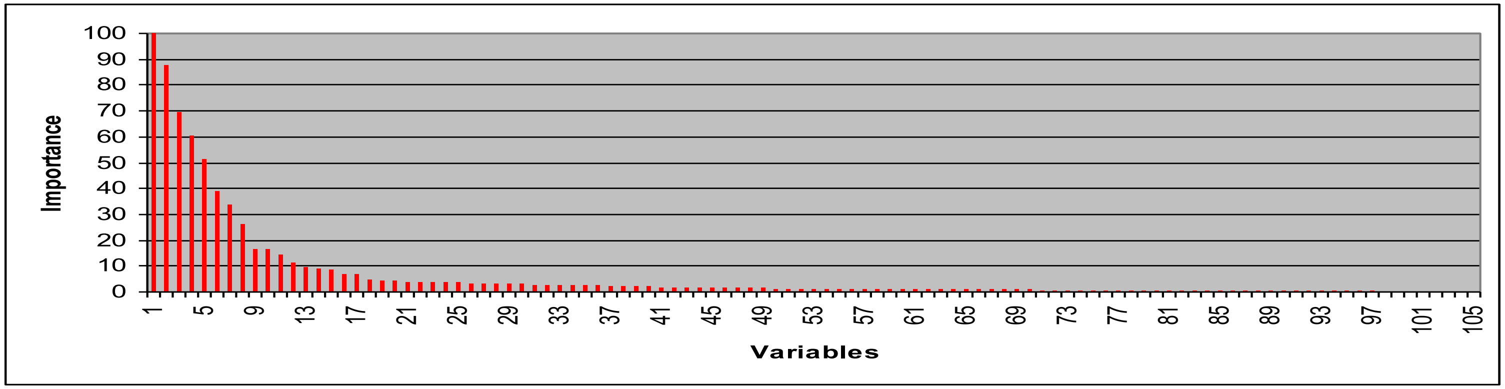
- Since we average several models, we loose the interpretability of the combined models and some efficiency
- However,
  - We still can use the ensembles to compute variable importance by averaging over all trees. Actually, this even stabilizes the estimates.
  - Averaging techniques can be parallelized and boosting type algorithm uses smaller trees. So, the increase of computing times is not so detrimental.

Experiments on Golub's microarray data

- 72 objects, 7129 numerical attributes (gene expressions), 2 classes (ALL and AL)
- Leave-one-out error with several variants

Method	Error
1 decision tree	22.2% (16/72)
Random forests (k=85,T=500)	9.7% (7/72)
Extra-trees (s_{th}=0.5, T=500)	5.5% (4/72)
Adaboost (1 test node, T=500)	1.4% (1/72)

- Variable importance with boosting

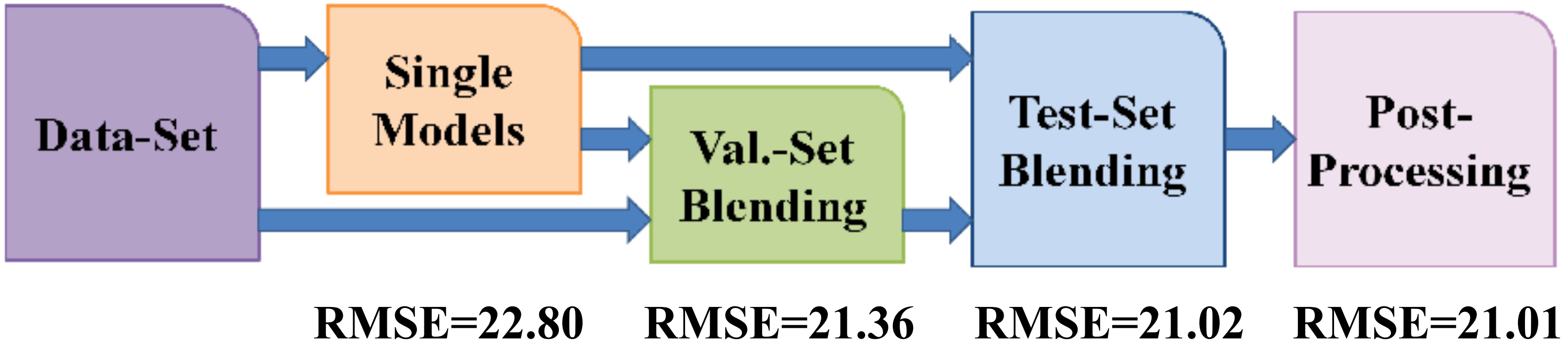


Conclusion

- Ensemble methods are very effective techniques to reduce bias and/or variance. They can transform a not so good method to a competitive method in terms of accuracy.
- Adaboost with trees is considered as one of the best « off-the-shelve » classification method.
- Interpretability of the model and efficiency of the method are difficult to preserve if we want to reduce variance significantly.
- There are other ways to tackle the variance/overfitting problem, e.g.:
  - Bayesian approaches (related to averaging techniques)
  - Support vector machines (they maintain a low variance by maximizing the classification margin)

Machine learning challenges

- Machine learning challenges are commonly won by « ensemble » solutions
- Netflix prize (1M$ reward):
  - Best solution combines 107 models obtained from different methods (stacking)
- Yahoo! 2011 KDD Cup (5,000$ reward):
  - Best solution uses two levels of stacking



Feature selection

- Techniques to reduce the number of features used by the learning algorithm
- Why?
  - Avoid overfitting and improve model performance
  - Improve interpretability
  - Provide faster and more cost-effective models
  - Reduce overall computing times (if the feature selection technique is fast)

Feature selection vs ranking

- Feature selection:
  - find a small (or the smallest) subset of features that maximizes accuracy
- Feature ranking:
  - sort the variable according to their relevance at predicting the output
- There are techniques in both families
- Feature selection can be obtained from a feature ranking:
  - Eg., select the top k features in a ranking

Some formalization

Let Y denote the class variable and $V=\{X_1,...,X_p\}$ the set of input variables:

- A feature X_i is:
  - **strongly relevant** iff $P(Y|X_i,V\setminus X_i) \neq P(Y|V\setminus X_i)$
  - **weakly relevant** iff it is not strongly relevant and $P(Y|X_i,S) \neq P(Y|S)$ for some subset $S \subset V$
  - **irrelevant** otherwise
- A subset $M \subseteq V$ of variables is a **markov boundary** for Y if it is **minimal** and $P(Y|M,V\setminus M) = P(Y|M)$
  - Features in M are either weakly or strongly relevant. They are all strongly relevant when the distribution P is strictly positive.
- Feature selection is often formulated as finding a Markov boundary M for Y

NB: All variables in a Markov boundary do not necessarily appear in the Bayes model (depending on the loss function)

Feature selection

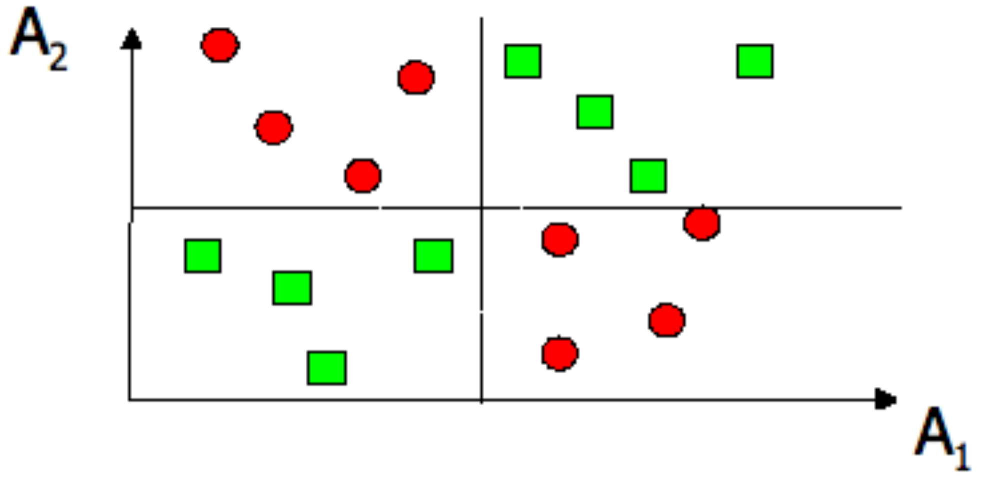
- Three main approaches:
  - **Filter:** a priori selection of the variables (ie, independently of the supervised learning algorithm)
  - **Embedded:** feature selection embedded in the learning algorithm
  - **Wrapper:** use CV to find the optimal set of features for a given algorithm

Filter techniques

- Main idea:
  - Associate a relevance score to each feature
  - Remove low-scoring features
- Often univariate scoring:
  - Any score measures used in decision trees
  - Statistical test (t-test, chi-square, etc.)
- But multivariate approaches exist (Relief, Markov blanket filter, decision trees, etc.)
- Optimal number of features can be determined by cross-validation

Univariate vs multivariate

Each feature is useless alone (low univariate scoring) but together they perfectly explain the classification



Filter techniques

- Advantages:
  - Univariate: fast and scalable
  - Independent of the SL algorithm
- Drawbacks:
  - Ignore the SL algorithm
  - Univariate: ignores feature dependencies
  - Multivariate: slower than univariate approaches

Embedded

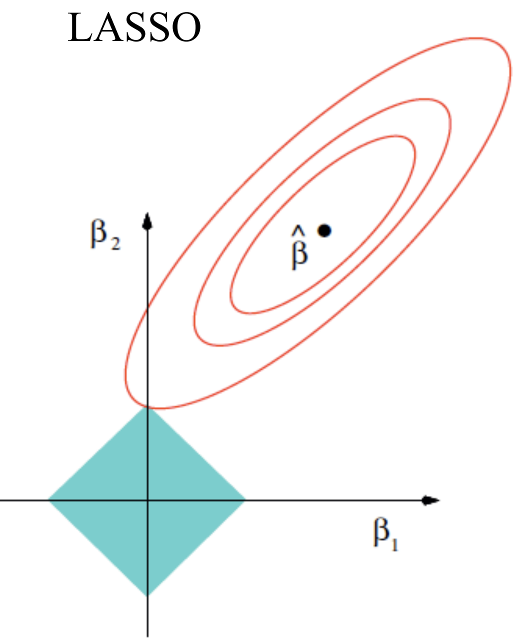
- Some supervised learning methods embed feature selection. The search for an optimal subset of features is built into the learning algorithm
- Examples:
  - Decision tree node splitting is a feature selection technique
  - Tree ensemble variable importance measures
  - Absolute weights in a linear SVM model

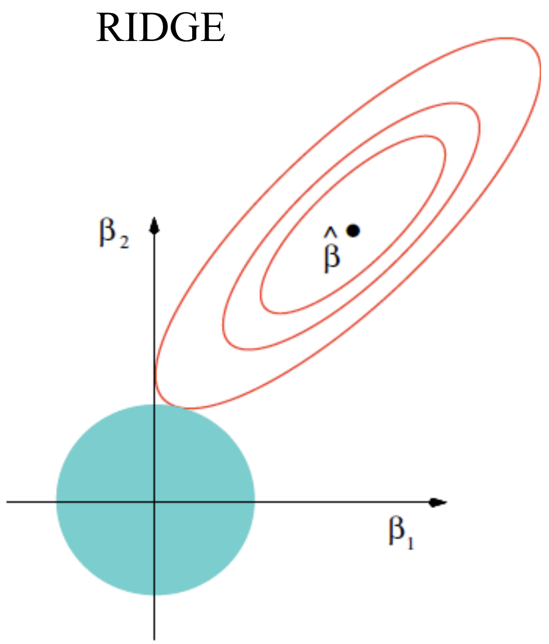
$$\hat{y}(\underline{x}) = \mathrm{sgn}(\sum_i w_i x_i + b)$$

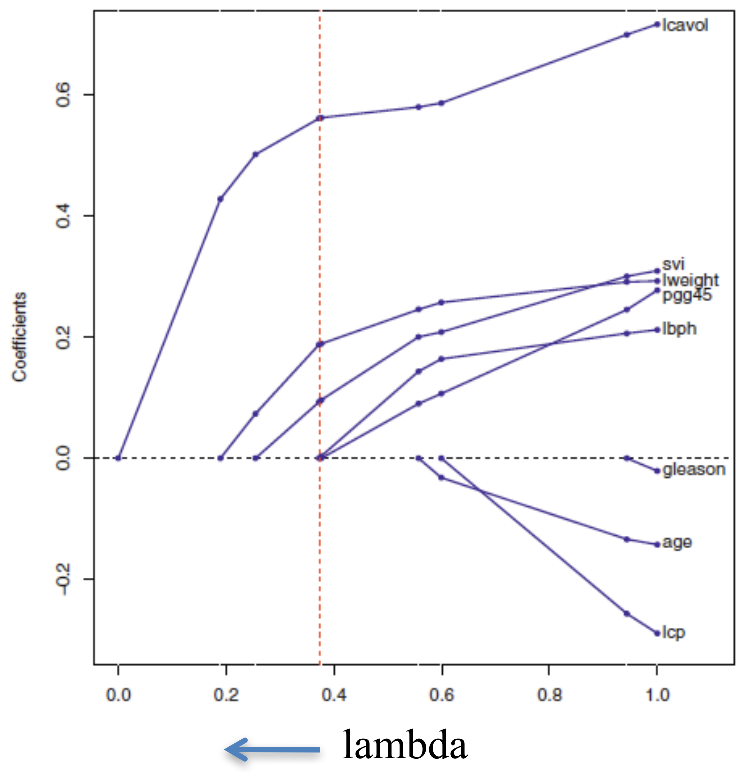
LASSO

Linear model learned with L1 penalization

$$\min_{\beta} \sum_{i=1}^{N} (y_i - (\beta_0 + \sum_j \beta_j x_j))^2 + \lambda \sum_j |\beta_j|$$







Embedded

- Advantages:
  - Usually computationally efficient
  - Well integrated with the learning algorithm (obviously)
  - Multivariate
- Drawbacks:
  - Specific to a given learning algorithm

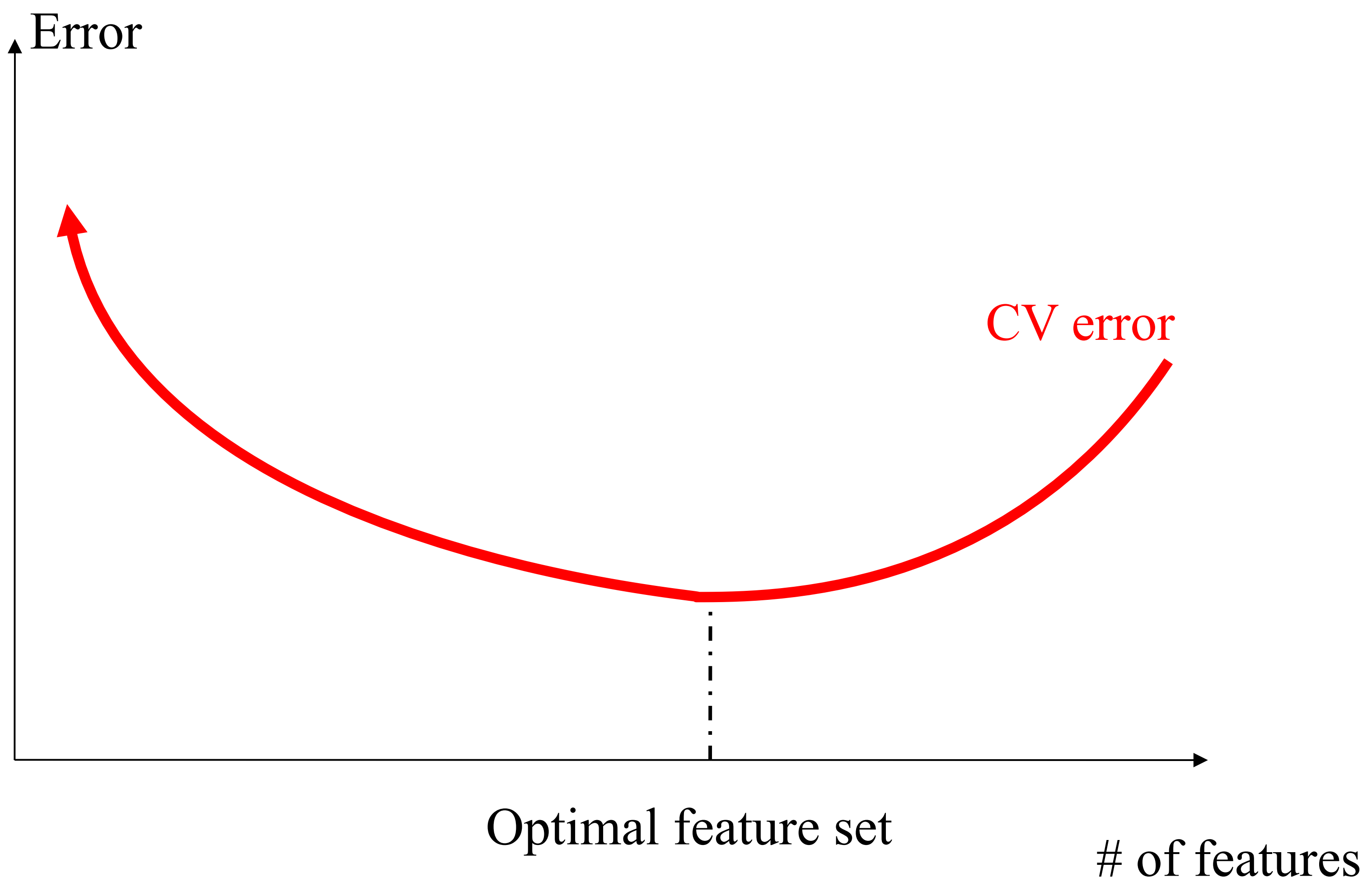
Wrapper methods

- Try to find a subset of features that maximizes the quality of the model induced by the learning algorithm.
- Quality of the model estimated by cross-validation
- As the number of subsets of *p* features is 2^p, all subsets can usually not be evaluated and heuristics are necessary
- Many approaches exist:
  - Forward or backward selection: add (remove) the variable that most decreases (less increases) the error
  - Optimization by genetic algorithms

Recursive feature elimination

- Popular wrapper, especially in bioinformatics
- Assume a learning algorithm that can rank the features (e.g., linear SVM, decision trees)
- Iterate (from the full feature set):
  - learn a model from the current feature set
  - rank the features with the model
  - remove the feature with the smallest ranking
- Keep the feature set that gives the lowest (CV) error

Recursive feature elimination



Wrapper methods

- Advantages:
  - Custom-tailored to the learning algorithm
  - Able to find interactions and remove redundant variables
- Drawbacks:
  - Prone to overfitting
    - It is often easy to find a small subset of noisy features that discriminates perfectly the classes
  - Expensive
    - We have to build a model for each subset of variables

Selection bias

- We often see this experiment:
  - From the dataset, select the N top variables using some filter
  - Evaluate an algorithm that uses these N variables as inputs by cross-validation (eg., LOO) on the dataset
- What is wrong with this protocol?

An artificial experiment

1000 variables

250 objects

A1	A2	...	A1000	Y
-0.86	0.17	...	0	C2
-2.3	-1.2	...	-0.42	C1
-0.37	-0.11	...	-0.64	C1
0.41	0.67	...	-0.8	C2
-0.51	-0.59	...	0.98	C2
-0.25	-0.27	...	-0.68	C1
-0.52	0.23	...	0.11	C1
-1.3	-0.2	...	0.14	C1
0.93	-0.78	...	-0.01	C2
-0.25	-0.29	...	0.69	C2
-0.6	0.92	...	-0.64	C1
0.22	-0.8	...	-0.5	C2
-0.62	0.2	...	0.08	C1
-0.3	0.8	...	0.02	C2
-0.91	0.44	...	-0.57	C1
0.76	0.65	...	-0.08	C1

Each variable randomly drawn from a N(0,1) distribution

The class is selected randomly (with P(C1)=P(C2)=0.5)

An artificial experiment

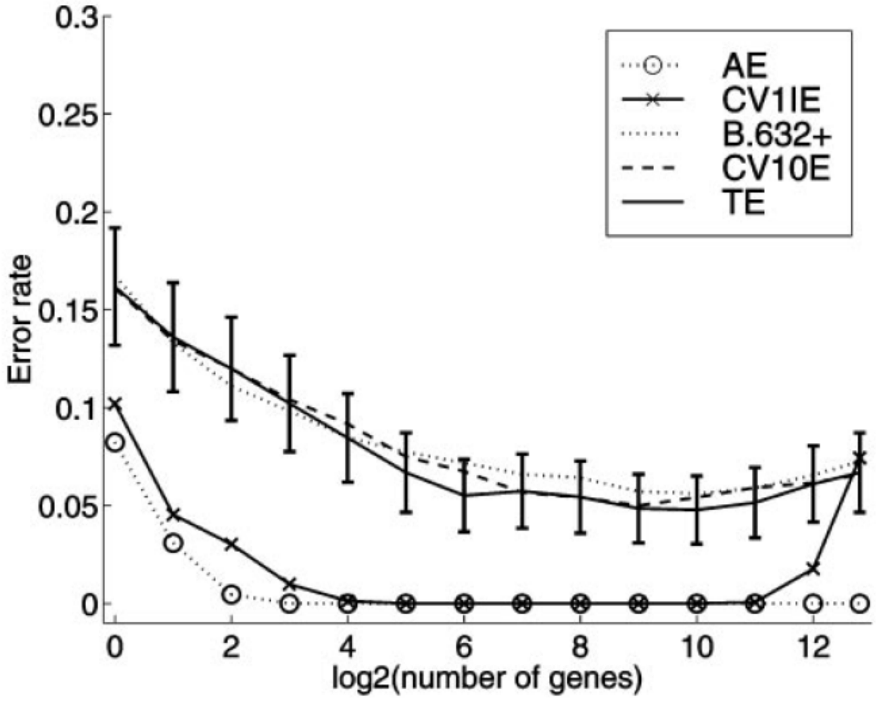
- Two trials:
  - Tree bagging without feature selection: 10-fold CV error: 52%
  - Tree bagging with the 20 top features (t-test): 10-fold CV error: 35%
- One could conclude that:
  - There are 20 interesting variables
  - From them, one can classify better than at random
- But, on a new set of 250 samples, the error is 52%

Selection bias

- We have both selected the variables and tested the model on the basis of the whole training set ⇒ overfitting
- Correct protocol:
  - Divide the LS into 10 folds
  - For i=1 to 10:
    - remove the i-th fold from the LS
    - select the top 20 variables from the remaining objects
    - learn the model using the 20 variables and the remaining objects
    - test the model on the i-th fold

Example on Golub et al's data

- SVM and recursive feature elimination



AE = Error on LS
CV1IE= internal LOO

CV10E= external 10-fold CV
TE= Error on an independent test sample
B.632+=another unbiased error estimate

Ambroise and McLachlan, PNAS, 2002

Further reading

- Ensemble methods
  - Hastie: 8.7, 8.8, 10.1-4, 15 (not in detail)
- Feature selection:
  - Hastie: 18
  - Guyon and Elisseeff, An introduction to variable and feature selection.
    http://www.jmlr.org/papers/volume3/guyon03a/guyon03a.pdf
    http://clopinet.com/fextract-book/IntroFS.pdf
  - Saeys et al. A review of feature selection techniques in bioinformatics
    http://dx.doi.org/10.1093/bioinformatics/btm344

Software

- Feature selection and ensembles (all):
  - scikit-learn
  - R
- Boosting (for large datasets)
  - XGBoost: http://xgboost.readthedocs.io/
  - Light-GBM: https://github.com/Microsoft/LightGBM