Machine Learning Methods for Gene Expression Data

Day 4

Dennis Wylie, UT Bioinformatics Consulting Group

May 24, 2018

Outline

Support vector machines (SVMs)

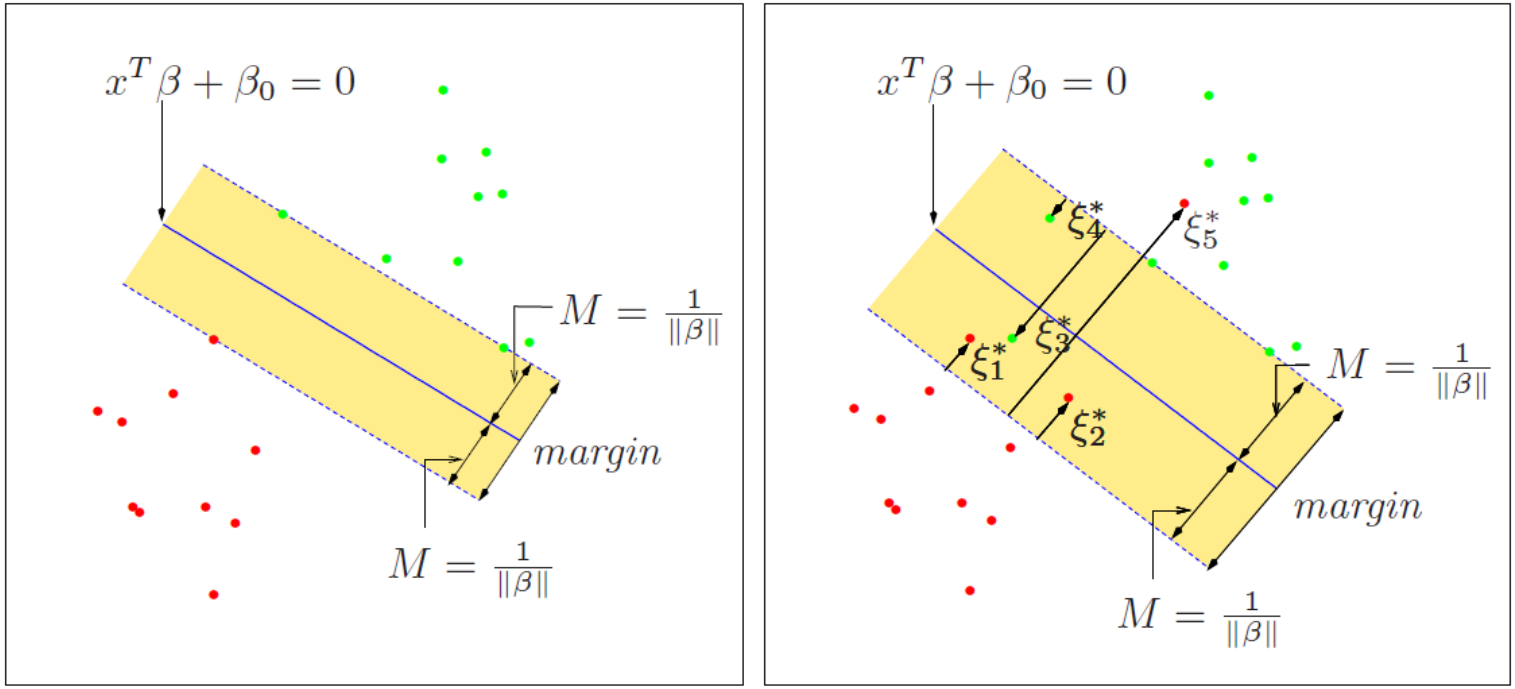


FIGURE 12.1. *Support vector classifiers. The left panel shows the separable case. The decision boundary is the solid line, while broken lines bound the shaded maximal margin of width* $2M = 2/\|\beta\|$. *The right panel shows the nonseparable (overlap) case. The points labeled* ξ_j^* *are on the wrong side of their margin by an amount* $\xi_j^* = M\xi_j$; *points on the correct side have* $\xi_j^* = 0$. *The margin is maximized subject to a total budget* $\sum \xi_i \leq$ *constant. Hence* $\sum \xi_j^*$ *is the total distance of points on the wrong side of their margin.*

Taken from Hastie *et al.* (2009).

Fitting SVMs

Here follow convention $y \in \{-1, 1\}$ instead of $y \in \{0, 1\}$.

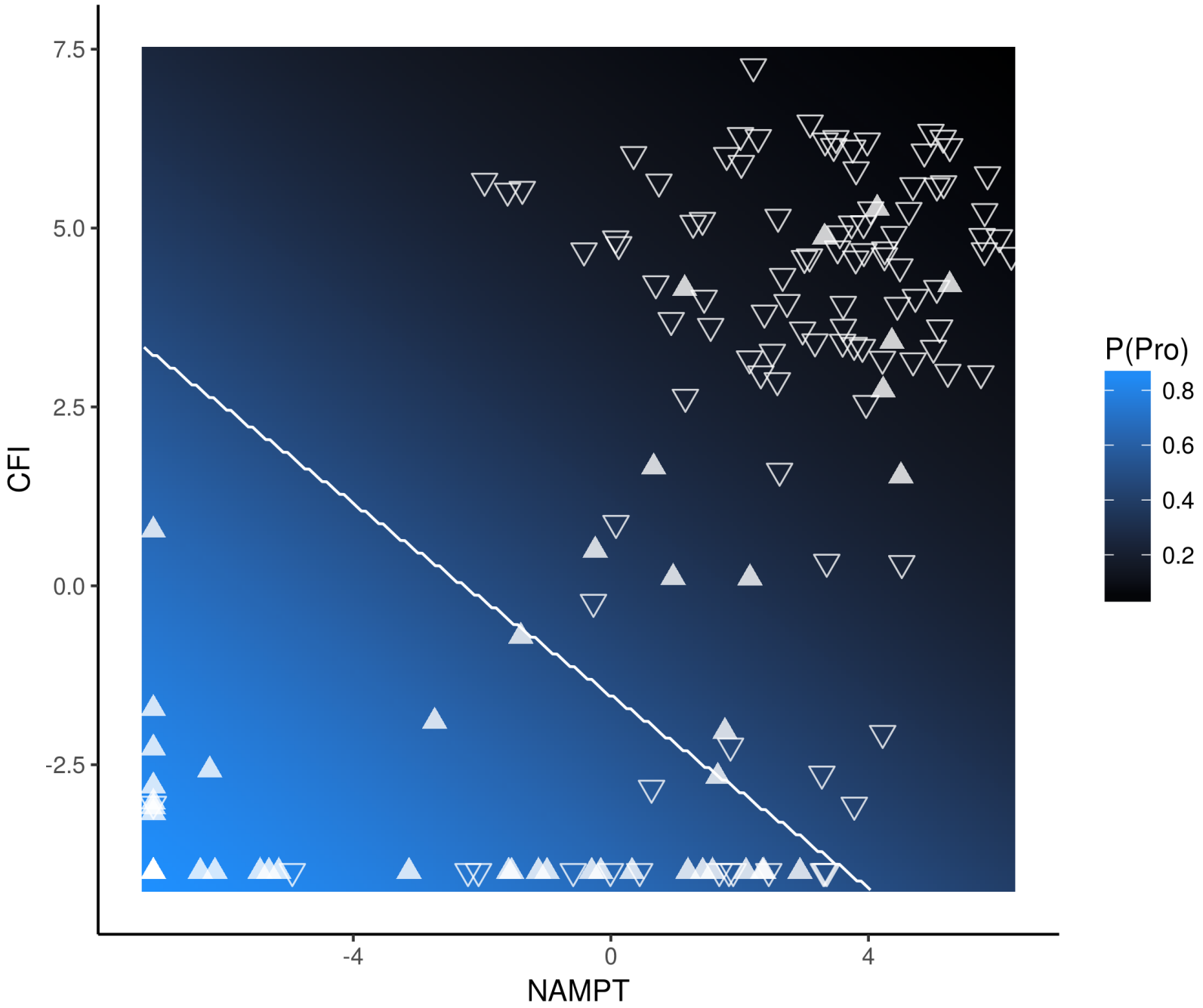
SVM parameters $\beta_0, \boldsymbol{\beta}, \underline{\xi}$ fit by optimization:

$$\underset{\beta_0, \boldsymbol{\beta}, \underline{\xi}}{\arg\min} \left\{ \frac{1}{2} \|\boldsymbol{\beta}\|^2 + C \sum_{i=1}^{n} \xi_i \right\}$$

subject to constraints

$$y_i(\boldsymbol{\beta} \cdot \mathbf{x}_i + \beta_0) \geq 1 - \xi_i$$
$$\xi_i \geq 0$$

Linear SVM



Nonlinear SVMs

Can fit SVM in nonlinearly transformed feature space.

For certain transformations, so-called "kernel trick" can be used: Given a particular transformation h, the kernel

$$k(\mathbf{x}, \mathbf{x}') = \langle h(\mathbf{x}), h(\mathbf{x}') \rangle$$

is actually all that is needed to fit SVM.

Nonlinear SVMs

Can fit SVM in nonlinearly transformed feature space.

For certain transformations, so-called "kernel trick" can be used: Given a particular transformation h, the kernel

$$k(\mathbf{x}, \mathbf{x}') = \langle h(\mathbf{x}), h(\mathbf{x}') \rangle$$

is actually all that is needed to fit SVM.

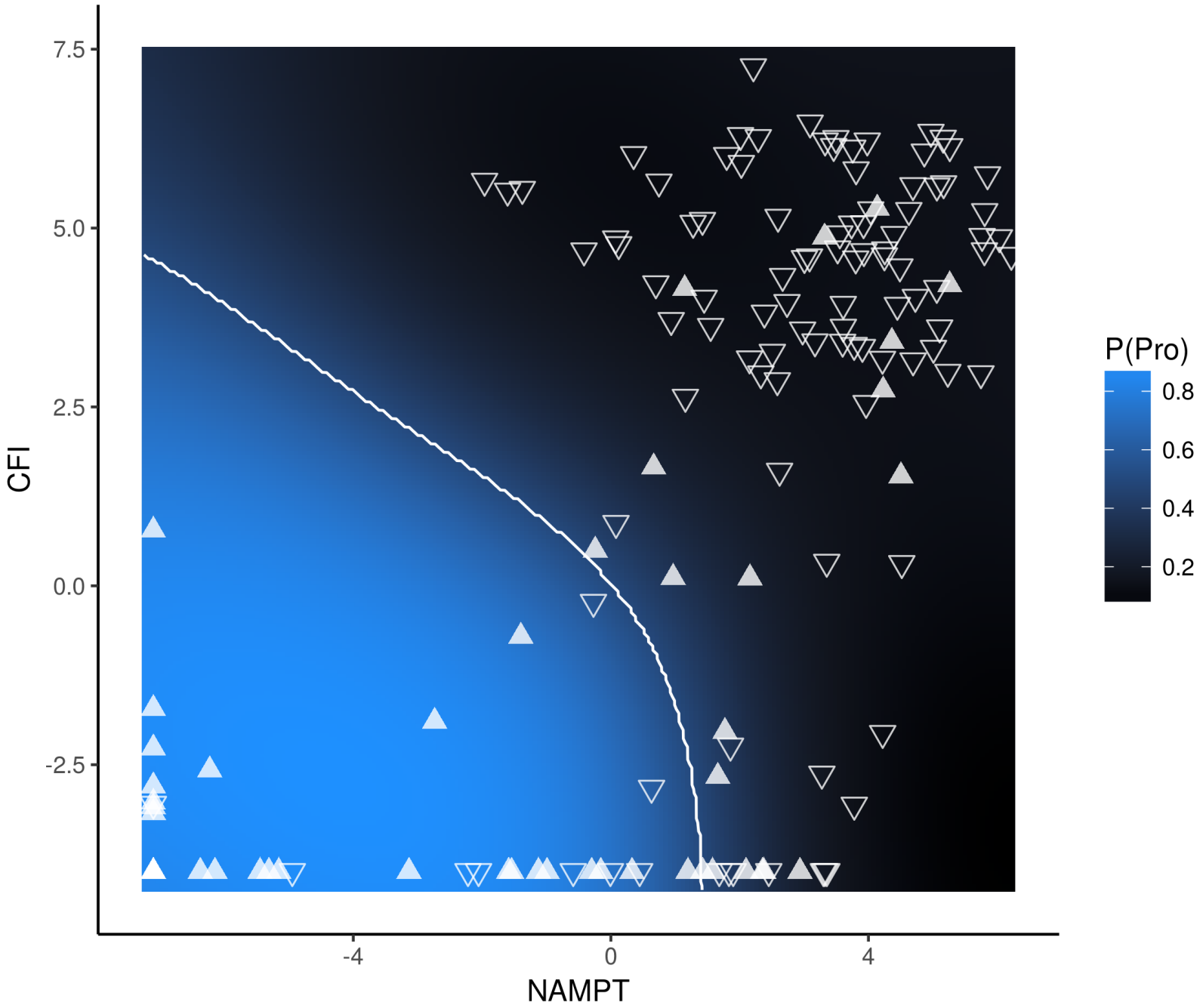
Most popular h:
special transformation designed to produce radial basis kernel

$$k(\mathbf{x}, \mathbf{x}') = \exp\left(-\gamma \|\mathbf{x} - \mathbf{x}'\|^2\right)$$

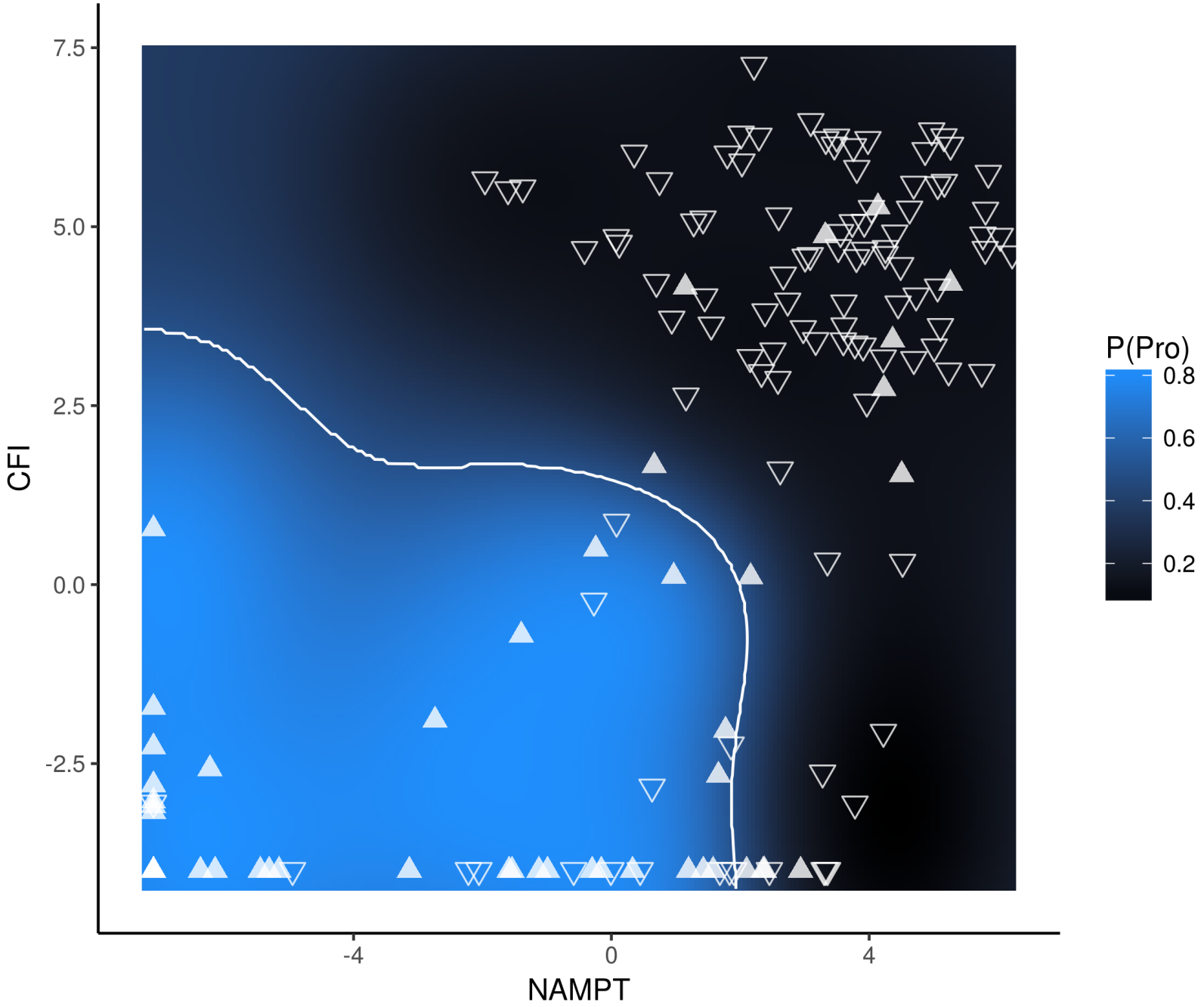
Inution: classify sample with features $\mathbf{x}$ based on

- (known) classes of similar training data $\mathbf{x}_i$
- where "similarity" quantified by kernel $k(\mathbf{x}, \mathbf{x}_i)$.

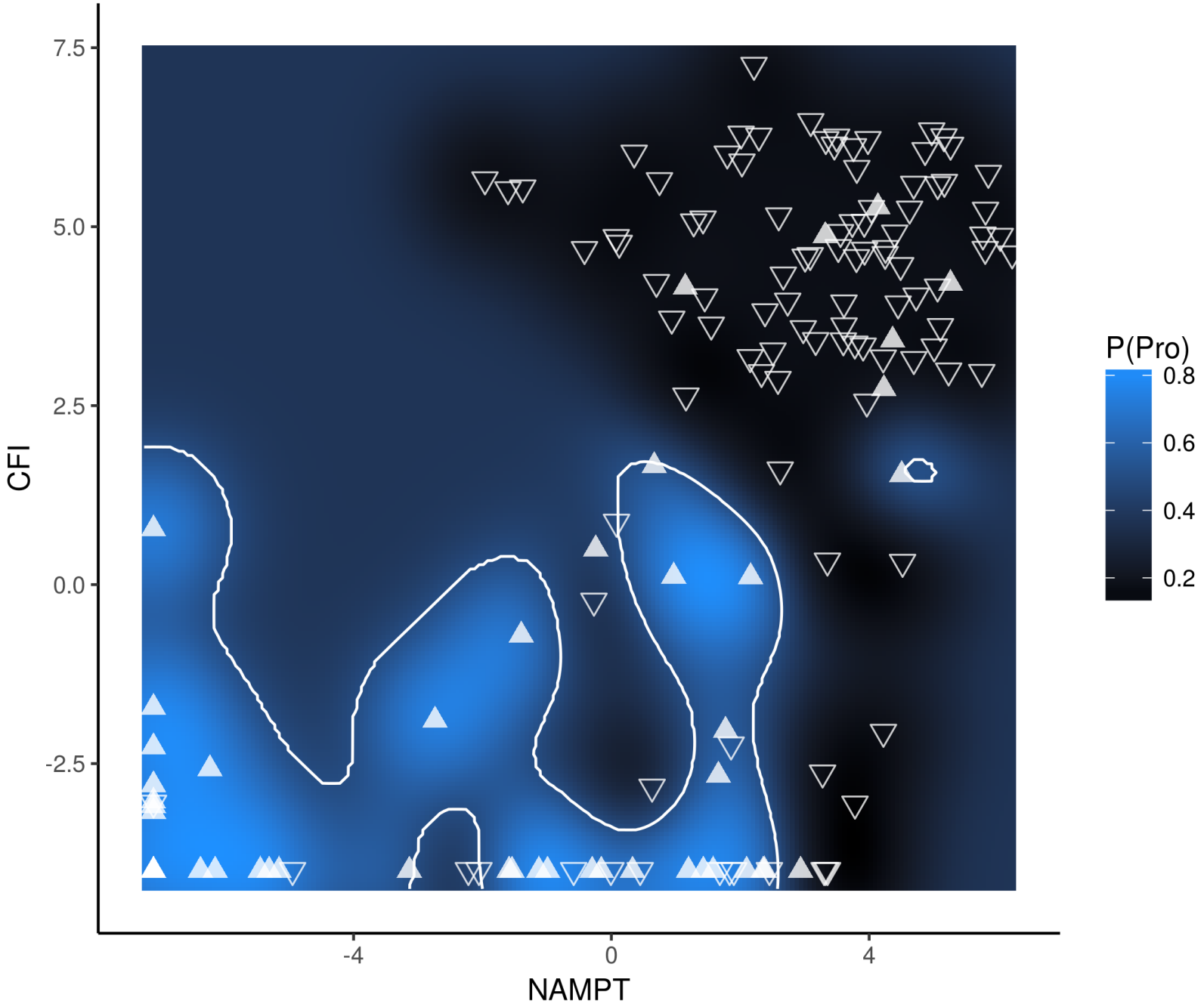
Radial SVM: $C = 1, \gamma = 0.5$



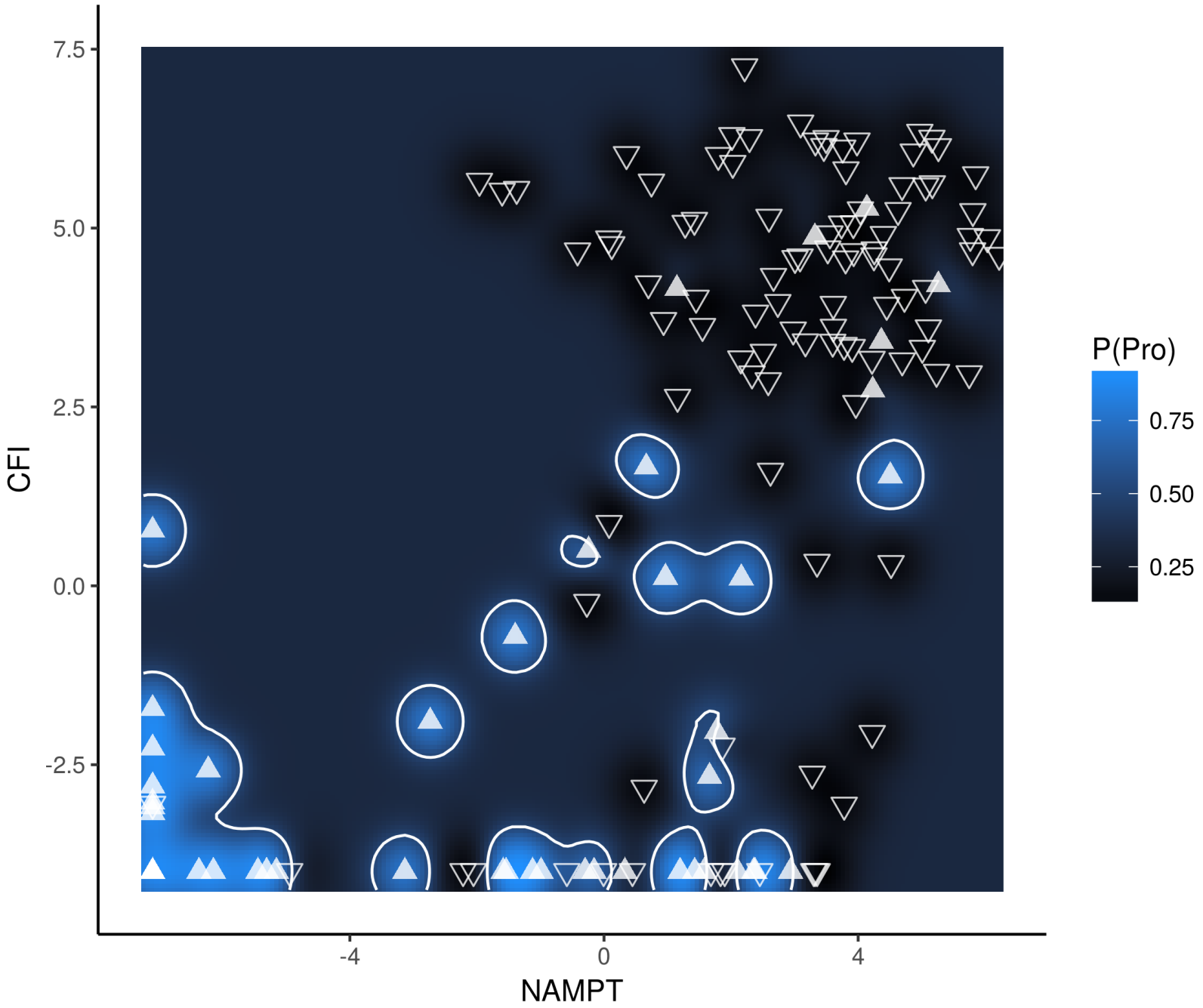
Radial SVM: $C = 1, \gamma = 2.5$



Radial SVM: $C = 1, \gamma = 12.5$



Radial SVM: $C = 1, \gamma = 62.5$



Uncertainty in Model Parameters

Often want to characterize uncertainty in model parameters.

Linear models: useful analytical results on confidence intervals

- other modeling strategies not always so lucky.

If gathering data were sufficiently cheap, could just replicate:

- the experiment which generated data
- and the modeling process

many times to

- empirically estimate the distribution of fit model parameters.

Bootstrapping

Bootstrapping: simulate such replication

- using just the one data set we actually have.

Bootstrapping

Bootstrapping: simulate such replication

- using just the one data set we actually have.

1. Generate a case-resampled data set $\underline{\mathbf{X}}^{\text{boot}}$ by drawing n random integers $R = \{r_i \in \{1, \ldots, n\}\}$ with replacement and setting

$$x_{gi}^{\text{boot}} = x_{gr_i}$$
$$y_i^{\text{boot}} = y_{r_i}$$

   Note that the r_i will generally not be unique!
2. Apply modeling strategy M to $(\underline{\mathbf{X}}^{\text{boot}}, \underline{y}^{\text{boot}})$ to obtain fitted parameters $\boldsymbol{\theta}^{\text{boot}}$.
3. Use model $(M, \boldsymbol{\theta}^{\text{boot}})$ to estimate parameter or statistic $\hat{\Omega}^{\text{boot}}$ of interest.
4. Repeat steps 1-3 B times, obtaining values Ω_b^{boot} for $b \in \{1, \ldots, B\}$ using models $(M, \boldsymbol{\theta}_b^{\text{boot}})$.

Bootstrapping for Performance Estimation

Bootstrapping can also be used as an alternative to cross-validation for estimation of prediction error Ω.

How to do this?

- Might try to estimate distribution of prediction error $\{\hat{\Omega}_b^{\text{full}}\}$
- on full (original) training set $\underline{\mathbf{X}}$.

Bootstrapping for Performance Estimation

Bootstrapping can also be used as an alternative to cross-validation for estimation of prediction error Ω.

How to do this?

- Might try to estimate distribution of prediction error $\{\hat{\Omega}_b^{\text{full}}\}$
- on full (original) training set $\underline{\mathbf{X}}$.

However, since bootstrap training sets were drawn from $\underline{\mathbf{X}}$, $\{\hat{\Omega}_b^{\text{full}}\}$ will suffer from resubstitution bias.

Bootstrapping for Performance Estimation

Bootstrapping can also be used as an alternative to cross-validation for estimation of prediction error Ω.

How to do this?

- Might try to estimate distribution of prediction error $\{\hat{\Omega}_b^{\text{full}}\}$
- on full (original) training set $\underline{\mathbf{X}}$.

However, since bootstrap training sets were drawn from $\underline{\mathbf{X}}$, $\{\hat{\Omega}_b^{\text{full}}\}$ will suffer from resubstitution bias.

Instead we could follow cross-validation methodology:

- use only models $(M, \boldsymbol{\theta}_b)$ for which
- sample i not used in resampled training set R_b

$$\hat{\Omega}^{\text{loo-boot}} = \frac{1}{n} \sum_i \frac{1}{|\{b \mid i \notin R_b\}|} \sum_{\{b | i \notin R_b\}} \hat{\Omega}(M, \boldsymbol{\theta}_b, y_i)$$

.632 Bootstrap

While $\{\hat{\Omega}_b^{\text{full}}\}$ are generally overly optimistic,

- $\hat{\Omega}^{\text{loo-boot}}$ may be too pessimistic . . .

.632 Bootstrap

While $\{\hat{\Omega}_b^{\text{full}}\}$ are generally overly optimistic,

- $\hat{\Omega}^{\text{loo-boot}}$ may be too pessimistic ...

Why? Because each bootstrap case-resampled training set

$$R_b = \{r_{bi} \mid i \in \{1, \ldots, n\}\}$$

generally contains only a fraction $1 - \frac{1}{e} \approx 0.632$ of the true training samples (albeit with some showing up multiple times!).

.632 Bootstrap

While $\{\hat{\Omega}_b^{\text{full}}\}$ are generally overly optimistic,

- $\hat{\Omega}^{\text{loo-boot}}$ may be too pessimistic ...

Why? Because each bootstrap case-resampled training set

$$R_b = \{r_{bi} \mid i \in \{1, \ldots, n\}\}$$

generally contains only a fraction $1 - \frac{1}{e} \approx 0.632$ of the true training samples (albeit with some showing up multiple times!).

Repeating training samples doesn't generally improve models ...

- learning models using only $\approx 63.2\%$ of the available data.

.632 Bootstrap

While $\{\hat{\Omega}_b^{\text{full}}\}$ are generally overly optimistic,

- $\hat{\Omega}^{\text{loo-boot}}$ may be too pessimistic ...

Why? Because each bootstrap case-resampled training set

$$R_b = \{r_{bi} \mid i \in \{1, \ldots, n\}\}$$

generally contains only a fraction $1 - \frac{1}{e} \approx 0.632$ of the true training samples (albeit with some showing up multiple times!).

Repeating training samples doesn't generally improve models ...

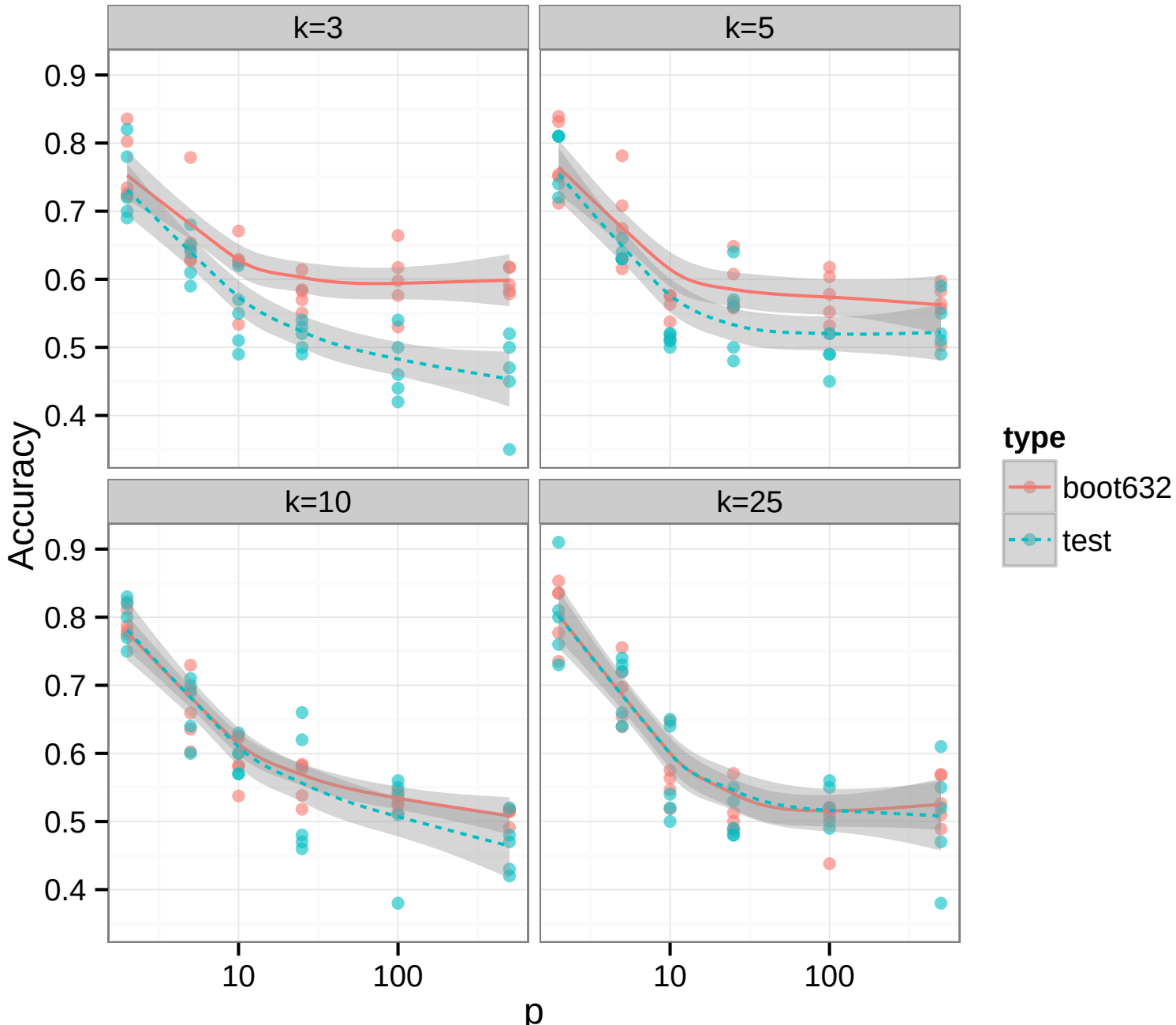
- learning models using only $\approx 63.2\%$ of the available data.

Efron & Tibshirani (1997) showed that

$$\Omega^{.632} = 0.368\,\Omega^{\text{resub}} + 0.632\,\Omega^{\text{loo-boot}}$$

works well in some situations.

.632 Bootstrap



.632+ Bootstrap

However, in cases where overfitting is more severe, Efron & Tibshirani (1997) recommend

$$\Omega^{.632+} = (1 - \hat{w})\,\Omega^{\text{resub}} + \hat{w}\,\Omega^{\text{loo-boot}}$$

where $\hat{w} \in [1 - \frac{1}{e}, 1]$ depends on the degree of overfitting.

There is a standard formula for calculating $\hat{w}$ for estimating prediction error using the .632+ bootstrap which you can look up; aside from Efron & Tibshirani (1997), Hastie *et al.* (2009) has a nice treatment.

Bagging: **B**ootstrap **Agg**regat**ing** Models

Might consider using set of B bootstrap case-resample trained models in place of a single model for making predictions.

Bagging: **B**ootstrap **Agg**regat**ing** Models

Might consider using set of B bootstrap case-resample trained models in place of a single model for making predictions.

For modeling strategy M yielding predictions $f_{M,\boldsymbol{\theta}}(\mathbf{x})$:

Repeat for $b \in \{1, \ldots, B\}$:

1. Generate $\underline{\mathbf{X}}_b$ by drawing n random integers $R_b = \{r_{bi} \in \{1, \ldots, n\}\}$ with replacement and setting $X_{bgi} = X_{gr_{bi}}$, $y_{bi} = y_{r_{bi}}$.
2. Fit M to $(\underline{\mathbf{X}}_b, \underline{y}_b)$ to obtain fitted parameters $\boldsymbol{\theta}_b$.

Bagged predictions for new data $\mathbf{x}$ using $(M, \{\boldsymbol{\theta}_b\})$:

$$f_{M,\{\boldsymbol{\theta}_b\}}(\mathbf{x}) = \frac{1}{B} \sum_{b=1}^{B} f_{M,\boldsymbol{\theta}_b}(\mathbf{x})$$

Bagging: What is it Good For?

From Breiman (1996):

> For unstable procedures bagging works well . . . The evidence, both experimental and theoretical, is that bagging can push a good but unstable procedure a significant step towards optimality. On the other hand, it can slightly degrade the performance of stable procedures.

In this context, "stability" is of the fit model parameters $\boldsymbol{\theta}$ with respect to the training data $\{\mathbf{x}_i, y_i\}$.

Bagging: What is it Good For?

From Breiman (1996):

> For unstable procedures bagging works well . . . The evidence, both experimental and theoretical, is that bagging can push a good but unstable procedure a significant step towards optimality. On the other hand, it can slightly degrade the performance of stable procedures.

In this context, "stability" is of the fit model parameters $\boldsymbol{\theta}$ with respect to the training data $\{\mathbf{x}_i, y_i\}$.

Perhaps the most celebrated application of bagging is in its application to generate **random forests** of **decision trees** . . .

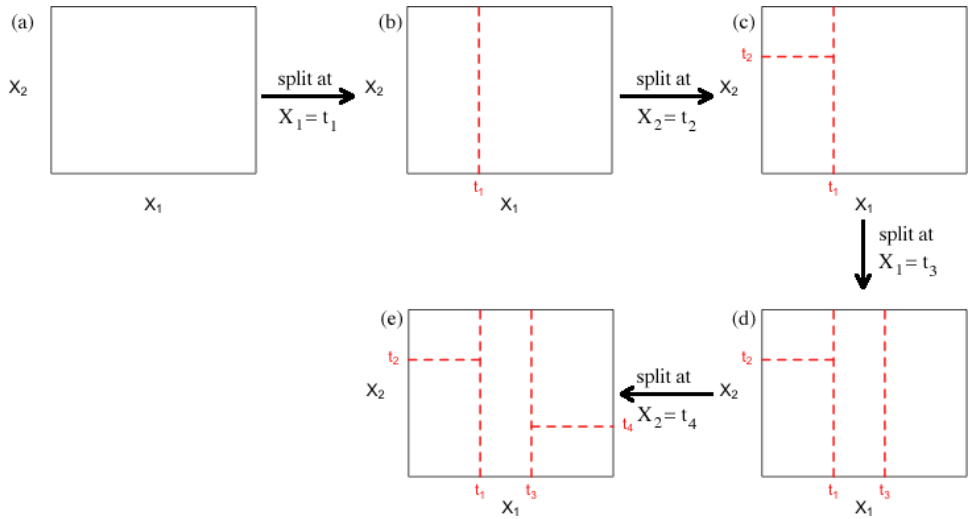
Decision Trees

$$\mathbb{P}(Y = y \mid \mathbf{X} = \mathbf{x}) = \sum_{k=1}^{K} p_k \, [\mathbf{x} \in R_k]$$

where the Iverson bracket expression

$$[\mathbf{x} \in R_k] = \begin{cases} 1 & \text{if } \mathbf{x} \in R_k, \\ 0 & \text{otherwise} \end{cases}$$

and the regions $R_k \in \mathbb{R}^p$ are defined by *recursive partitioning*.



Images from http://www.unc.edu/courses/2010spring/ecol/562/001/images/lectures/lecture21

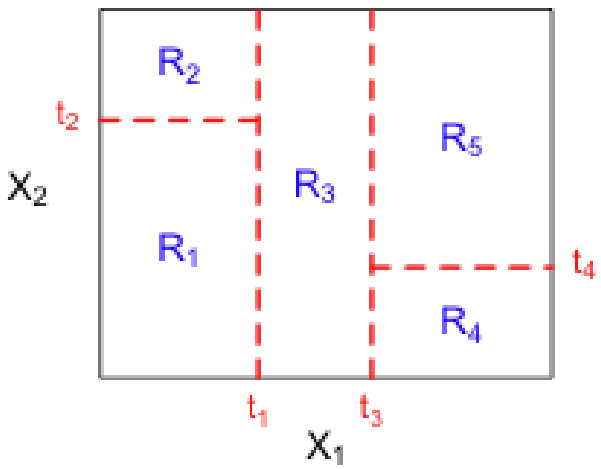
Decision Trees

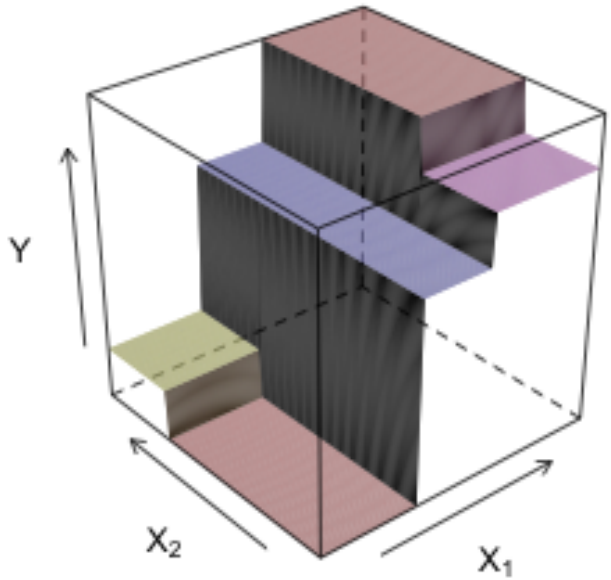
$$\mathbb{P}(Y = y \mid \mathbf{X} = \mathbf{x}) = \sum_{k=1}^{K} p_k \, [\mathbf{x} \in R_k]$$

where the Iverson bracket expression

$$[\mathbf{x} \in R_k] = \begin{cases} 1 & \text{if } \mathbf{x} \in R_k, \\ 0 & \text{otherwise} \end{cases}$$

and the regions $R_k \in \mathbb{R}^p$ are defined by *recursive partitioning*.





Images from http://www.unc.edu/courses/2010spring/ecol/562/001/images/lectures/lecture21

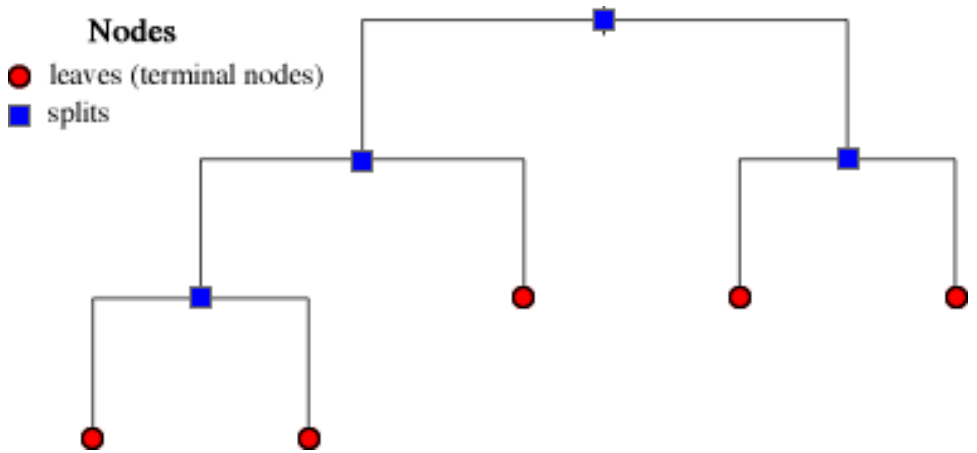
Decision Trees

$$\mathbb{P}(Y = y \mid \mathbf{X} = \mathbf{x}) = \sum_{k=1}^{K} p_k \ [\mathbf{x} \in R_k]$$

where the Iverson bracket expression

$$[\mathbf{x} \in R_k] = \begin{cases} 1 & \text{if } \mathbf{x} \in R_k, \\ 0 & \text{otherwise} \end{cases}$$

and the regions $R_k \in \mathbb{R}^p$ are defined by *recursive partitioning*.



Images from http://www.unc.edu/courses/2010spring/ecol/562/001/images/lectures/lecture21

Fitting a Decision Tree

Recursive partitioning generally selects both the variable X_g to split on and the value t at which to split that variable by minimizing an impurity measure Q such as

$$\begin{aligned} Q_{\text{misclassification}} &= \min(p_<, 1 - p_<) \\ &\quad + \min(p_>, 1 - p_>) \\ Q_{\text{gini}} &= p_<(1 - p_<) \\ &\quad + 2p_>(1 - p_>) \\ Q_{\text{deviance}} &= -p_< \log p_< - (1 - p_<)\log(1 - p_<) \\ &\quad - p_> \log p_> - (1 - p_>)\log(1 - p_>) \end{aligned}$$

where

$$p_< = \frac{|\{i \mid y_i = 1, x_{ig} < t\}|}{|\{i \mid x_{ig} < t\}|}$$

$$p_> = \frac{|\{i \mid y_i = 1, x_{ig} \geq t\}|}{|\{i \mid x_{ig} \geq t\}|}$$

Random Forests

A random forest is constructed by:

Repeat for $b \in \{1, \ldots, B\}$:

1. Generate $\underline{\mathbf{X}}_b$ by drawing n random integers $R_b = \{r_{bi} \in \{1, \ldots, n\}\}$ with replacement and setting $X_{bgi} = X_{gr_{bi}}$.
2. Recursively partition $\underline{\mathbf{X}}_b$ with the modification that for each partitioning step, $m < p$ of the features are randomly selected from a uniform distribution and the best split is selected from only this set of variables.
   - m random features redrawn for each new split.
   - Commonly $m \approx \sqrt{p}$.

Boosting

An alternative approach to model aggregation is boosting; one variant (Discrete AdaBoost, Friedman *et al.* (2000)) consists of:

1. Observation weights w_{1i} ($i \in \{1, \ldots, n\}$) are initialized: $w_{1i} = \frac{1}{n}$.
2. For each $b \in \{1, \ldots, B\}$, fit M to $(\underline{\mathbf{X}}, \underline{y}, \underline{w})$ and set

$$\text{err}_b = \frac{1}{\sum_i w_{bi}} \sum_i w_{bi} \left[y_i \neq f_{M,\boldsymbol{\theta}_b}(\mathbf{x}_i) \right]$$

$$\alpha_b = \log\left(\frac{1 - \text{err}_b}{\text{err}_b} \right)$$

$$w_{(b+1)i} = w_{bi} \exp\left(\alpha_b \left[y_i \neq f_{M,\boldsymbol{\theta}_b}(\mathbf{x}_i) \right] \right)$$

3. Define $f_{M,\{\boldsymbol{\theta}_b\}} = \text{sign}\left(\sum_b \alpha_b f_{M,\boldsymbol{\theta}_b} \right)$.

Boosting

An alternative approach to model aggregation is boosting; one variant (Discrete AdaBoost, Friedman *et al.* (2000)) consists of:

1. Observation weights w_{1i} ($i \in \{1, \ldots, n\}$) are initialized: $w_{1i} = \frac{1}{n}$.
2. For each $b \in \{1, \ldots, B\}$, fit M to $(\underline{\mathbf{X}}, \underline{y}, \underline{w})$ and set

$$\text{err}_b = \frac{1}{\sum_i w_{bi}} \sum_i w_{bi} \left[y_i \neq f_{M,\boldsymbol{\theta}_b}(\mathbf{x}_i) \right]$$

$$\alpha_b = \log\left(\frac{1 - \text{err}_b}{\text{err}_b} \right)$$

$$w_{(b+1)i} = w_{bi} \exp\left(\alpha_b \left[y_i \neq f_{M,\boldsymbol{\theta}_b}(\mathbf{x}_i) \right] \right)$$

3. Define $f_{M,\{\boldsymbol{\theta}_b\}} = \text{sign}\left(\sum_b \alpha_b f_{M,\boldsymbol{\theta}_b} \right)$.

- Requires M to be able to fit weighted data set.

Boosting

Training of b^{th} tree fit focused on those samples misclassified by first $b-1$ trees.

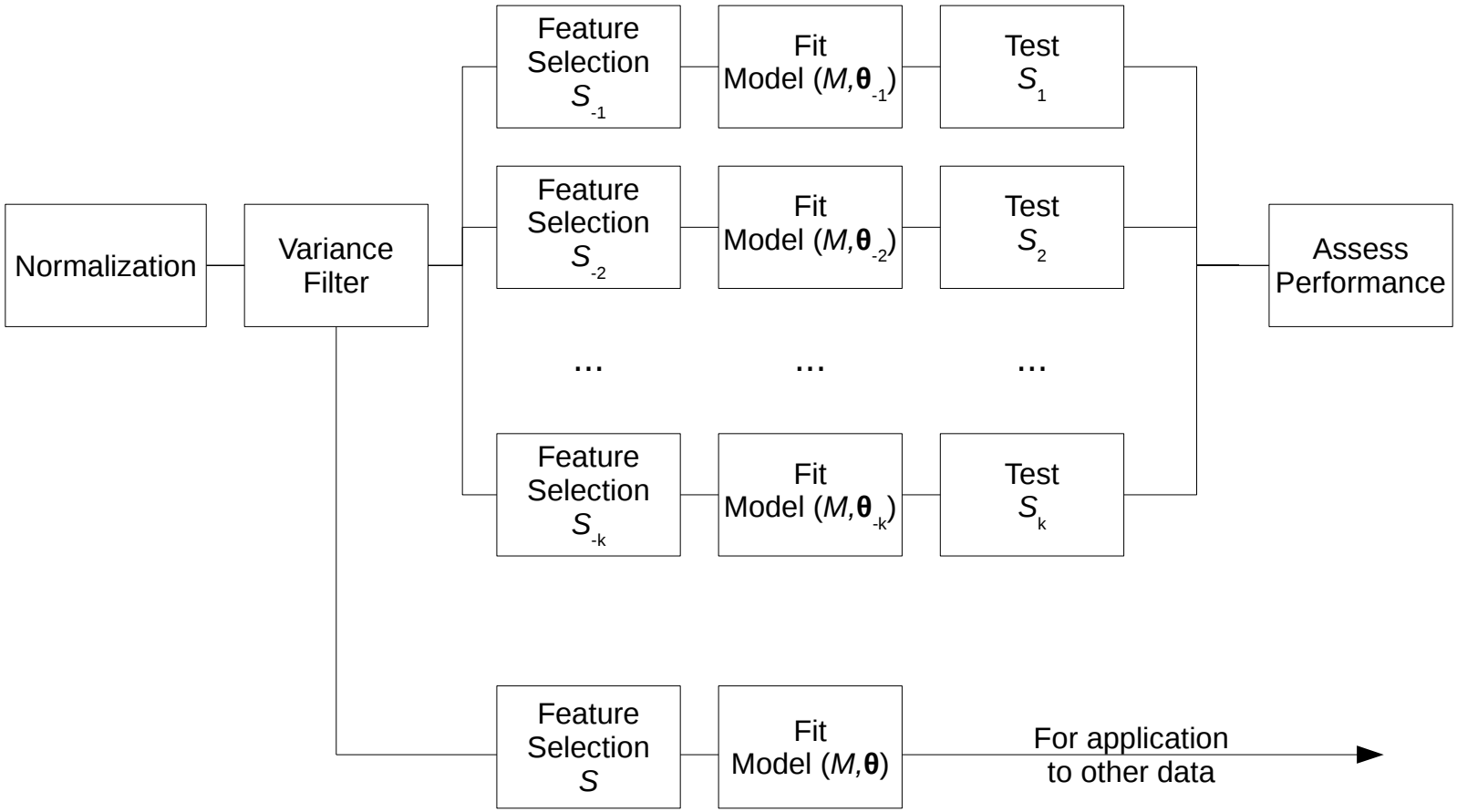
Individual trees cast votes weighted by α_b to determine outcome of final classifier.

Friedman *et al.* (2000) showed that from a statistical point of view boosting can be seen as a form of additive modeling with similarities to logistic regression.

Key concept which emerges across many variations of boosting is the importance of **slow learning**.

- often use very shallow trees (e.g., stumps)
- may be further facilitated by shrinkage and randomization

Cross-Validation Flow



References I

Breiman, Leo. 1996. Bagging predictors. *Machine Learning*, **24**(2), 123–140.

Efron, Bradley, & Tibshirani, Robert. 1997. Improvements on cross-validation: the 632+ bootstrap method. *Journal of the American Statistical Association*, **92**(438), 548–560.

Friedman, Jerome, Hastie, Trevor, Tibshirani, Robert, *et al.* . 2000. Additive logistic regression: a statistical view of boosting. *The Annals of Statistics*, **28**(2), 337–407.

Hastie, Trevor, Tibshirani, Robert, & Friedman, Jerome. 2009. *The Elements of Statistical Learning*. Springer.