STK4030 Summary

Kristoffer H. Hellton

23rd of November 2015

The plan for today

- General overview of the course
- Key concepts
- Relations between methods
- Some further extensions
- Uncovered topics
- What is relevant for the written exam!

Statistical learning

Spam or Email??

	george	you	your	hp	free	hpl	!	our	re	edu	remove
spam	0.00	2.26	1.38	0.02	0.52	0.01	0.51	0.51	0.13	0.01	0.28
email	1.27	1.27	0.44	0.90	0.07	0.43	0.11	0.18	0.42	0.29	0.01

Statistical learning

Spam or Email??

	george	you	your	hp	free	hpl	!	our	re	edu	remove
spam	0.00	2.26	1.38	0.02	0.52	0.01	0.51	0.51	0.13	0.01	0.28
email	1.27	1.27	0.44	0.90	0.07	0.43	0.11	0.18	0.42	0.29	0.01

- Input $x \in \mathcal{R}^p$, output y, given data $\{(\boldsymbol{x}_i, y_i), i = 1, ..., N\}$, want to build up a relation between $\boldsymbol{x}$ and y
  - Numerical: Regression

Statistical learning

Spam or Email??

	george	you	your	hp	free	hpl	!	our	re	edu	remove
spam	0.00	2.26	1.38	0.02	0.52	0.01	0.51	0.51	0.13	0.01	0.28
email	1.27	1.27	0.44	0.90	0.07	0.43	0.11	0.18	0.42	0.29	0.01

- Input $x \in \mathcal{R}^p$, output y, given data $\{(\boldsymbol{x}_i, y_i), i = 1, ..., N\}$, want to build up a relation between $\boldsymbol{x}$ and y
  - Numerical: Regression
  - Categorical: Classification

Statistical learning

Spam or Email??

	george	you	your	hp	free	hpl	!	our	re	edu	remove
spam	0.00	2.26	1.38	0.02	0.52	0.01	0.51	0.51	0.13	0.01	0.28
email	1.27	1.27	0.44	0.90	0.07	0.43	0.11	0.18	0.42	0.29	0.01

- Input $x \in \mathcal{R}^p$, output y, given data $\{(\boldsymbol{x}_i, y_i), i = 1, ..., N\}$, want to build up a relation between $\boldsymbol{x}$ and y
  - Numerical: Regression
  - Categorical: Classification

The aim is important!

- Prediction and learning, *OR*
- Explanation and inference.

Inference: fitting models and quantifying uncertainty, the traditional focus of statistics

Loss functions

- Typically
  - Squared error loss for regression
    - Optimal: regression function $\widehat{Y} = E[Y|X = x] \equiv f(\mathbf{x})$

Loss functions

- Typically
  - Squared error loss for regression
    - Optimal: regression function $\widehat{Y} = E[Y|X = x] \equiv f(\mathbf{x})$
  - Classification error for classification
    - Optimal: Bayes classifier $\widehat{G} = \arg\max_k \Pr(G = k|\mathbf{x})$, $\Pr(G = k|\mathbf{x}) = E[I(G = k)|\mathbf{x}]$

Loss functions

- Typically
  - Squared error loss for regression
    - Optimal: regression function $\widehat{Y} = E[Y|X = x] \equiv f(\mathbf{x})$
  - Classification error for classification
    - Optimal: Bayes classifier $\widehat{G} = \arg\max_k \Pr(G = k|\mathbf{x})$, $\Pr(G = k|\mathbf{x}) = E[I(G = k)|\mathbf{x}]$
- Other possibilities

Loss functions

- Typically
  - Squared error loss for regression
    - Optimal: regression function $\widehat{Y} = E[Y|X = x] \equiv f(\mathbf{x})$
  - Classification error for classification
    - Optimal: Bayes classifier $\widehat{G} = \arg\max_k \Pr(G = k|\mathbf{x})$, $\Pr(G = k|\mathbf{x}) = E[I(G = k)|\mathbf{x}]$
- Other possibilities
  - L_1 loss is more robust towards outliers
  - Exponential loss beneficial in classification (Adaboost) different weights

Regression

- Basic problem: Assume $y = f(\boldsymbol{x}) + \varepsilon, \varepsilon \sim (0, \sigma^2)$

Regression

- Basic problem: Assume $y = f(\boldsymbol{x}) + \varepsilon, \varepsilon \sim (0, \sigma^2)$
- Want to estimate $f(\boldsymbol{x}) = E[y|\boldsymbol{x}]$.

Regression

- Basic problem: Assume $y = f(\boldsymbol{x}) + \varepsilon, \varepsilon \sim (0, \sigma^2)$
- Want to estimate $f(\boldsymbol{x}) = E[y|\boldsymbol{x}]$.
- Need to consider: sample size, dimension, previous knowledge, ...

Regression

- Basic problem: Assume $y = f(\boldsymbol{x}) + \varepsilon, \varepsilon \sim (0, \sigma^2)$
- Want to estimate $f(\boldsymbol{x}) = E[y|\boldsymbol{x}]$.
- Need to consider: sample size, dimension, previous knowledge, ...
- One solution: Least squares $\sum_{i=1}^{n}(y_i - f(\boldsymbol{x}_i))^2$

Regression

- Basic problem: Assume $y = f(\boldsymbol{x}) + \varepsilon, \varepsilon \sim (0, \sigma^2)$
- Want to estimate $f(\boldsymbol{x}) = E[y|\boldsymbol{x}]$.
- Need to consider: sample size, dimension, previous knowledge, ...
- One solution: Least squares $\sum_{i=1}^{n}(y_i - f(\boldsymbol{x}_i))^2$
- Flexible f but additional restrictions/penalties:
    - Linear
    - Basis expansions
    - Additive
    - Tree structure
    - Smooth
    - Dimension reduction (Variable selection/PCA)
    - Penalties on parameters (Ridge/Lasso)
    - Selection: AIC, BIC, Crossvalidation

Classification

- Basic problem: Assume $y_i \in \{1, ..., K\}$ classes

Classification

- Basic problem: Assume $y_i \in \{1, ..., K\}$ classes
- Wish to estimate $\widehat{G} = \arg\max_k \Pr(G = k|\mathbf{x})$,
  $\Pr(G = k|\mathbf{x}) = E[I(G = k)|\mathbf{x}]$

Classification

- Basic problem: Assume $y_i \in \{1, ..., K\}$ classes
- Wish to estimate $\widehat{G} = \arg\max_k \Pr(G = k|\mathbf{x})$, $\Pr(G = k|\mathbf{x}) = E[I(G = k)|\mathbf{x}]$
- Define

$$y_{i,k} = \begin{cases} 1 & \text{if } y_i = k \\ 0 & \text{otherwise} \end{cases}$$

- Can model regression function $\hat{f}_k(x)$ for each k and classify $\hat{G}(\mathbf{x}) = \arg\max_k \hat{y}_k(\mathbf{x})$:

Classification

- Basic problem: Assume $y_i \in \{1, ..., K\}$ classes
- Wish to estimate $\widehat{G} = \arg\max_k \Pr(G = k|\mathbf{x})$,
  $\Pr(G = k|\mathbf{x}) = E[I(G = k)|\mathbf{x}]$
- Define

$$y_{i,k} = \begin{cases} 1 & \text{if } y_i = k \\ 0 & \text{otherwise} \end{cases}$$

- Can model regression function $\hat{f}_k(x)$ for each k and classify $\hat{G}(\mathbf{x}) = \arg\max_k \hat{y}_k(\mathbf{x})$:

Classification

- Basic problem: Assume $y_i \in \{1, ..., K\}$ classes
- Wish to estimate $\widehat{G} = \arg\max_k \Pr(G = k|\mathbf{x})$,
  $\Pr(G = k|\mathbf{x}) = E[I(G = k)|\mathbf{x}]$
- Define

$$y_{i,k} = \begin{cases} 1 & \text{if } y_i = k \\ 0 & \text{otherwise} \end{cases}$$

- Can model regression function $\hat{f}_k(x)$ for each k and classify $\hat{G}(\mathbf{x}) = \arg\max_k \hat{y}_k(\mathbf{x})$:
  - linear regression
  - logistic regression

Classification

- Basic problem: Assume $y_i \in \{1, ..., K\}$ classes
- Wish to estimate $\widehat{G} = \arg\max_k \Pr(G = k|\mathbf{x})$,
  $\Pr(G = k|\mathbf{x}) = E[I(G = k)|\mathbf{x}]$
- Define

$$y_{i,k} = \begin{cases} 1 & \text{if } y_i = k \\ 0 & \text{otherwise} \end{cases}$$

- Can model regression function $\hat{f}_k(x)$ for each k and classify $\hat{G}(\mathbf{x}) = \arg\max_k \hat{y}_k(\mathbf{x})$:
  - linear regression
  - logistic regression
- Can model $\Pr(G = k|\mathbf{x})$

Classification

- Basic problem: Assume $y_i \in \{1, ..., K\}$ classes
- Wish to estimate $\widehat{G} = \arg\max_k \Pr(G = k|\mathbf{x})$, $\Pr(G = k|\mathbf{x}) = E[I(G = k)|\mathbf{x}]$
- Define

$$y_{i,k} = \begin{cases} 1 & \text{if } y_i = k \\ 0 & \text{otherwise} \end{cases}$$

- Can model regression function $\hat{f}_k(x)$ for each k and classify $\hat{G}(\mathbf{x}) = \arg\max_k \hat{y}_k(\mathbf{x})$:
  - linear regression
  - logistic regression
- Can model $\Pr(G = k|\mathbf{x})$
  - Gaussian densities, equal $\Sigma_1 = \Sigma_2 \cdots$: LDA

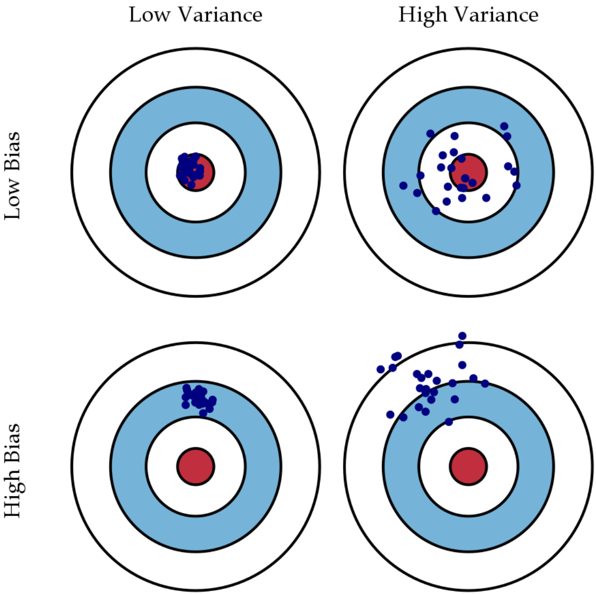
Classification

- Basic problem: Assume $y_i \in \{1, ..., K\}$ classes
- Wish to estimate $\widehat{G} = \arg\max_k \Pr(G = k|\mathbf{x})$,
  $\Pr(G = k|\mathbf{x}) = E[I(G = k)|\mathbf{x}]$
- Define

$$y_{i,k} = \begin{cases} 1 & \text{if } y_i = k \\ 0 & \text{otherwise} \end{cases}$$

- Can model regression function $\hat{f}_k(x)$ for each k and classify $\hat{G}(\mathbf{x}) = \arg\max_k \hat{y}_k(\mathbf{x})$:
  - linear regression
  - logistic regression
- Can model $\Pr(G = k|\mathbf{x})$
  - Gaussian densities, equal $\Sigma_1 = \Sigma_2 \cdots$: LDA
  - Gaussian densities, different $\Sigma_1 \neq \Sigma_2 \cdots$: QDA

Bias versus variance



Bias versus variance

$$\mathrm{Err}(\boldsymbol{x}_0) = \mathrm{E}[(Y - \hat{f}(x_0))^2 | X = x_0]$$

Bias versus variance

$$
\begin{aligned}
\mathrm{Err}(\boldsymbol{x}_0) &= \mathrm{E}[(Y - \hat{f}(x_0))^2 | X = x_0] \\
&= \sigma_\epsilon^2 + [\mathrm{E}\hat{f}(x_0) - f(x_0)]^2 + \mathrm{E}[\hat{f}(x_0) - E\hat{f}(x_0)]^2
\end{aligned}
$$

Bias versus variance

$$
\begin{aligned}
\mathrm{Err}(\boldsymbol{x}_0) &= \mathrm{E}[(Y - \hat{f}(x_0))^2 | X = x_0] \\
&= \sigma_\epsilon^2 + [\mathrm{E}\hat{f}(x_0) - f(x_0)]^2 + \mathrm{E}[\hat{f}(x_0) - E\hat{f}(x_0)]^2 \\
&= \text{Irreducible error} + \text{Bias}^2 + \text{Variance}
\end{aligned}
$$

Bias versus variance

$$\begin{aligned}\text{Err}(\boldsymbol{x}_0) &= \text{E}[(Y - \hat{f}(x_0))^2 | X = x_0] \\ &= \sigma_\epsilon^2 + [\text{E}\hat{f}(x_0) - f(x_0)]^2 + \text{E}[\hat{f}(x_0) - E\hat{f}(x_0)]^2 \\ &= \text{Irreducible error} + \text{Bias}^2 + \text{Variance}\end{aligned}$$

- Too simple model: Variance small, Bias large

Bias versus variance

$$
\begin{aligned}
\mathrm{Err}(\boldsymbol{x}_0) &= \mathrm{E}[(Y - \hat{f}(x_0))^2 | X = x_0] \\
&= \sigma_\epsilon^2 + [\mathrm{E}\hat{f}(x_0) - f(x_0)]^2 + \mathrm{E}[\hat{f}(x_0) - E\hat{f}(x_0)]^2 \\
&= \text{Irreducible error} + \text{Bias}^2 + \text{Variance}
\end{aligned}
$$

- Too simple model: Variance small, Bias large
- Too complex model: Bias small, variance large

Bias versus variance

$$
\begin{aligned}
\mathrm{Err}(\boldsymbol{x}_0) &= \mathrm{E}[(Y - \hat{f}(x_0))^2 | X = x_0] \\
&= \sigma_\epsilon^2 + [\mathrm{E}\hat{f}(x_0) - f(x_0)]^2 + \mathrm{E}[\hat{f}(x_0) - E\hat{f}(x_0)]^2 \\
&= \text{Irreducible error} + \text{Bias}^2 + \text{Variance}
\end{aligned}
$$

- Too simple model: Variance small, Bias large
- Too complex model: Bias small, variance large
- Trade-off: bias $\leftrightarrow$ variance

Bias versus variance

$$
\begin{aligned}
\mathrm{Err}(\boldsymbol{x}_0) &= \mathrm{E}[(Y - \hat{f}(x_0))^2 | X = x_0] \\
&= \sigma_\epsilon^2 + [\mathrm{E}\hat{f}(x_0) - f(x_0)]^2 + \mathrm{E}[\hat{f}(x_0) - E\hat{f}(x_0)]^2 \\
&= \text{Irreducible error} + \text{Bias}^2 + \text{Variance}
\end{aligned}
$$

- Too simple model: Variance small, Bias large
- Too complex model: Bias small, variance large
- Trade-off: bias $\leftrightarrow$ variance
- Same data for fitting and evaluation too optimistic! (Over-fitting)

Example: K nearest neighbor

The best fit to training data: $K = 1$. Bad for test data.

Example: K nearest neighbor

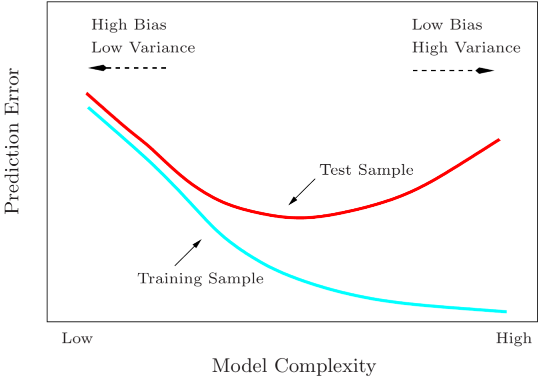
The best fit to training data: $K = 1$. Bad for test data.
Need to find an optimal model complexity

$$\mathrm{Err}(\boldsymbol{x}_0) = \sigma_\epsilon^2 + [\frac{1}{k}\sum_{l=1}^{k} f(x_{(l)}) - f(x_0)]^2 + \frac{\sigma_\epsilon^2}{k}.$$

Example: K nearest neighbor

The best fit to training data: $K = 1$. Bad for test data.
Need to find an optimal model complexity

$$\text{Err}(\boldsymbol{x}_0) = \sigma_\epsilon^2 + [\frac{1}{k}\sum_{l=1}^{k} f(x_{(l)}) - f(x_0)]^2 + \frac{\sigma_\epsilon^2}{k}.$$



Model assessment and selection

- How to choose between models/methods?

Model assessment and selection

- How to choose between models/methods?
- Want low prediction error on new data. But using the same data to fit and evaluate gives too optimistic prediction error

Model assessment and selection

- How to choose between models/methods?
- Want low prediction error on new data. But using the same data to fit and evaluate gives too optimistic prediction error
- Need independent evaluations for
  - selecting

Model assessment and selection

- How to choose between models/methods?
- Want low prediction error on new data. But using the same data to fit and evaluate gives too optimistic prediction error
- Need independent evaluations for
  - selecting
    - model/method
    - smoothness/complexity parameter

Model assessment and selection

- How to choose between models/methods?
- Want low prediction error on new data. But using the same data to fit and evaluate gives too optimistic prediction error
- Need independent evaluations for
  - selecting
    - model/method
    - smoothness/complexity parameter
  - evaluating the true prediction error for the final model/method

Training/validation/test sets

- Usual to divide data into
  - Training set: Selecting/fitting model/method
  - Test set: Evaluating final model

Training/validation/test sets

- Usual to divide data into
  - Training set: Selecting/fitting model/method
  - Test set: Evaluating final model
- Sometimes training set divided into two:
  - Training set: Fitting of model/method
  - Validation set: Selection of model/method

  But training/validation/test sets can each be too small

Cross-validation

- Cross-validation merges training/validation:

Cross-validation

- Cross-validation merges training/validation:
  1. Divide training data randomly into K groups.

Cross-validation

- Cross-validation merges training/validation:
    1. Divide training data randomly into K groups.
    2. For a specific model parameter value, repeat for $k = 1, \ldots, K$:

Cross-validation

- Cross-validation merges training/validation:
  1. Divide training data randomly into K groups.
  2. For a specific model parameter value, repeat for $k = 1, \ldots, K$:
     - Fit the model on $K - 1$ groups, leaving out group k,

Cross-validation

- Cross-validation merges training/validation:
  1. Divide training data randomly into K groups.
  2. For a specific model parameter value, repeat for $k = 1, \ldots, K$:
     - Fit the model on $K - 1$ groups, leaving out group k,
     - predict response of group k to find prediction error,

Cross-validation

- Cross-validation merges training/validation:
  1. Divide training data randomly into K groups.
  2. For a specific model parameter value, repeat for $k = 1, \ldots, K$:
     - Fit the model on $K - 1$ groups, leaving out group k,
     - predict response of group k to find prediction error,
  3. average prediction error over all data.
- Repeat over a grid of parameter values, select the parameter value with lowest error.

Cross-validation

- Cross-validation merges training/validation:
  1. Divide training data randomly into K groups.
  2. For a specific model parameter value, repeat for $k = 1, \ldots, K$:
     - Fit the model on $K - 1$ groups, leaving out group k,
     - predict response of group k to find prediction error,
  3. average prediction error over all data.
- Repeat over a grid of parameter values, select the parameter value with lowest error.
- Number of folds: 2-fold/N-fold (leave-one-out). More data gives less bias, but a larger overlap of folds gives correlated predictions and larger variance.

Cross-validation

- Cross-validation merges training/validation:
  1. Divide training data randomly into K groups.
  2. For a specific model parameter value, repeat for $k = 1, \ldots, K$:
     - Fit the model on $K - 1$ groups, leaving out group k,
     - predict response of group k to find prediction error,
  3. average prediction error over all data.
- Repeat over a grid of parameter values, select the parameter value with lowest error.
- Number of folds: 2-fold/N-fold (leave-one-out). More data gives less bias, but a larger overlap of folds gives correlated predictions and larger variance.
- Leave-one-out CV is more computationally intensive, but no randomness by fold division.

Cross-validation

- Cross-validation merges training/validation:
  1. Divide training data randomly into K groups.
  2. For a specific model parameter value, repeat for $k = 1, \dots, K$:
     - Fit the model on $K - 1$ groups, leaving out group k,
     - predict response of group k to find prediction error,
  3. average prediction error over all data.
- Repeat over a grid of parameter values, select the parameter value with lowest error.
- Number of folds: 2-fold/N-fold (leave-one-out). More data gives less bias, but a larger overlap of folds gives correlated predictions and larger variance.
- Leave-one-out CV is more computationally intensive, but no randomness by fold division.
- Compromise: 10-fold or 5-fold crossvalidation.

Linear models - regression

Linear in inputs: $f(\boldsymbol{x}) = \beta_0 + \sum_{j=1}^{p} \beta_j \boldsymbol{x}_j$:

- simple, easy to interpret

Linear models - regression

Linear in inputs: $f(\boldsymbol{x}) = \beta_0 + \sum_{j=1}^{p} \beta_j \boldsymbol{x}_j$:

- simple, easy to interpret
- gives better predictions than nonlinear (even when true) if
  - small sample size $n/$ large dimension p
  - small signal-to-noise ratio

Linear models - regression

Linear in inputs: $f(\boldsymbol{x}) = \beta_0 + \sum_{j=1}^{p} \beta_j \boldsymbol{x}_j$:

- simple, easy to interpret
- gives better predictions than nonlinear (even when true) if
  - small sample size n/ large dimension p
  - small signal-to-noise ratio
- can handle non-linearity by transformation

Linear models - regression

Linear in inputs: $f(\boldsymbol{x}) = \beta_0 + \sum_{j=1}^{p} \beta_j \boldsymbol{x}_j$:

- simple, easy to interpret
- gives better predictions than nonlinear (even when true) if
  - small sample size n/ large dimension p
  - small signal-to-noise ratio
- can handle non-linearity by transformation

But large p or small n gives problems:

- prediction accuracy

Linear models - regression

Linear in inputs: $f(\boldsymbol{x}) = \beta_0 + \sum_{j=1}^{p} \beta_j \boldsymbol{x}_j$:

- simple, easy to interpret
- gives better predictions than nonlinear (even when true) if
  - small sample size n/ large dimension p
  - small signal-to-noise ratio
- can handle non-linearity by transformation

But large p or small n gives problems:

- prediction accuracy
- interpretation of "big picture"

Linear models - regression

Linear in inputs: $f(\boldsymbol{x}) = \beta_0 + \sum_{j=1}^{p} \beta_j \boldsymbol{x}_j$:

- simple, easy to interpret
- gives better predictions than nonlinear (even when true) if
  - small sample size n/ large dimension p
  - small signal-to-noise ratio
- can handle non-linearity by transformation

But large p or small n gives problems:

- prediction accuracy
- interpretation of “big picture”
- Solution: restrictions on parameters

Linear models - regression

Linear in inputs: $f(\boldsymbol{x}) = \beta_0 + \sum_{j=1}^{p} \beta_j \boldsymbol{x}_j$:

- simple, easy to interpret
- gives better predictions than nonlinear (even when true) if
  - small sample size n/ large dimension p
  - small signal-to-noise ratio
- can handle non-linearity by transformation

But large p or small n gives problems:

- prediction accuracy
- interpretation of "big picture"
- Solution: restrictions on parameters
  - Ridge/Lasso penalty

Linear models - regression

Linear in inputs: $f(\boldsymbol{x}) = \beta_0 + \sum_{j=1}^{p} \beta_j \boldsymbol{x}_j$:

- simple, easy to interpret
- gives better predictions than nonlinear (even when true) if
  - small sample size n/ large dimension p
  - small signal-to-noise ratio
- can handle non-linearity by transformation

But large p or small n gives problems:

- prediction accuracy
- interpretation of “big picture”
- Solution: restrictions on parameters
  - Ridge/Lasso penalty
  - Best subset selection

Linear models - regression

Linear in inputs: $f(\boldsymbol{x}) = \beta_0 + \sum_{j=1}^{p} \beta_j \boldsymbol{x}_j$:

- simple, easy to interpret
- gives better predictions than nonlinear (even when true) if
  - small sample size n/ large dimension p
  - small signal-to-noise ratio
- can handle non-linearity by transformation

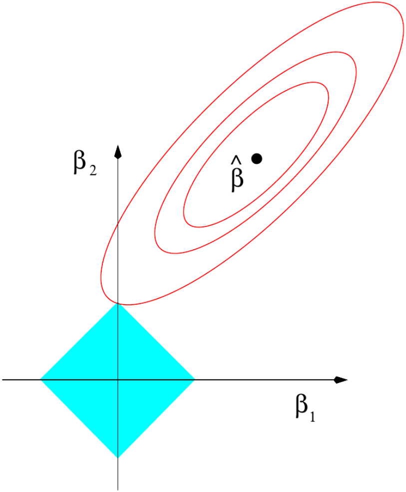
But large p or small n gives problems:

- prediction accuracy
- interpretation of "big picture"
- Solution: restrictions on parameters
  - Ridge/Lasso penalty
  - Best subset selection
  - Dimension reduction: PCR/PLS

Lasso regression

$$PRSS_{\lambda}^{lasso}(\beta) = \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j\right)^2 + \lambda \sum_{j=1}^{p} |\beta_j|$$

Penalized regression method,
RSS + penalty

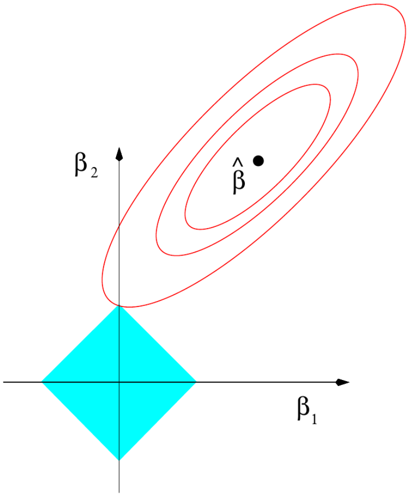


Lasso regression

$$PRSS_{\lambda}^{lasso}(\beta) = \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j \right)^2 + \lambda \sum_{j=1}^{p} |\beta_j|$$

Penalized regression method,
RSS + penalty

- Variable selection: β's can exactly zero.

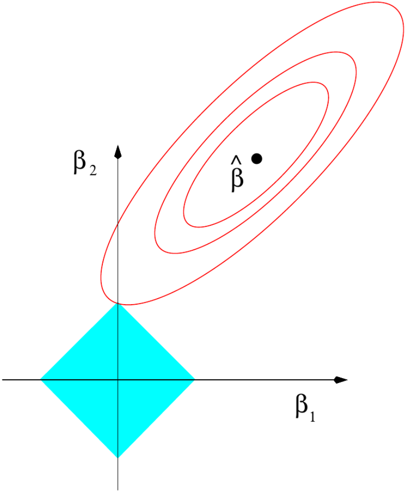


Lasso regression

$$PRSS_\lambda^{lasso}(\beta) = \sum_{i=1}^{n}\left(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j\right)^2 + \lambda \sum_{j=1}^{p} |\beta_j|$$

Penalized regression method,
RSS + penalty

- Variable selection: β's can exactly zero.
- Indifferent to the choice among correlated variables.



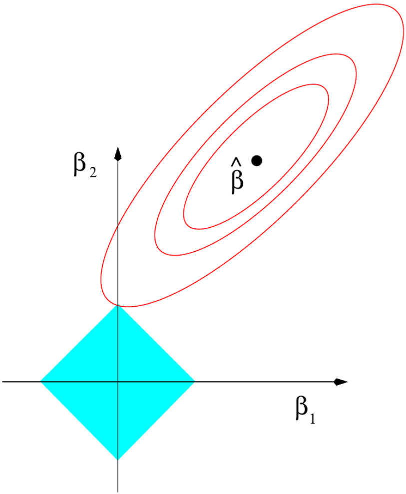
Lasso regression

$$PRSS_\lambda^{lasso}(\beta) = \sum_{i=1}^n \left(y_i - \beta_0 - \sum_{j=1}^p x_{ij}\beta_j\right)^2 + \lambda \sum_{j=1}^p |\beta_j|$$

Penalized regression method,
RSS + penalty

- Variable selection: β's can exactly zero.
- Indifferent to the choice among correlated variables.

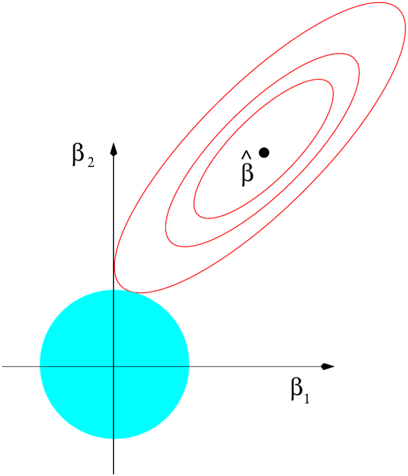
Not linear in **y**, must be calculated by algorithm; least angle regression (LAR).



Ridge regression

$$PRSS_{\lambda}^{ridge}(\beta) = \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j \right)^2 + \lambda \sum_{j=1}^{p} \beta_j^2$$

Penalized regression method,
RSS + penalty

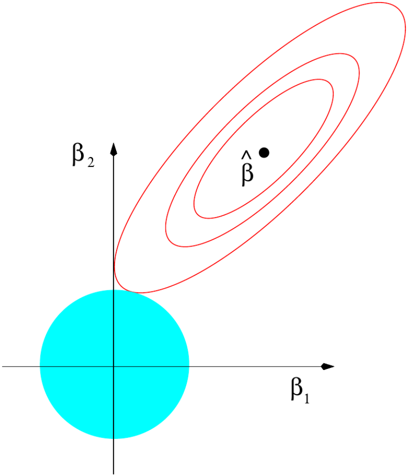


Ridge regression

$$PRSS_\lambda^{ridge}(\beta) = \sum_{i=1}^n \left(y_i - \beta_0 - \sum_{j=1}^p x_{ij}\beta_j\right)^2 + \lambda \sum_{j=1}^p \beta_j^2$$

Penalized regression method,
RSS + penalty

- No variable selection: all β's are non-zero.

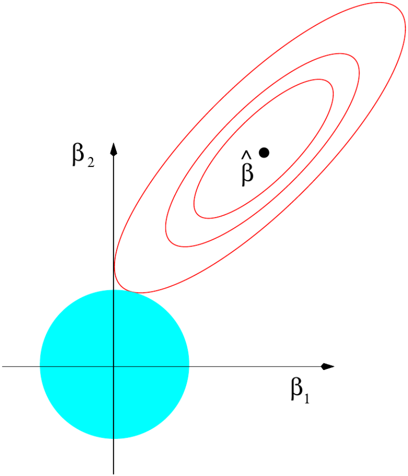


Ridge regression

$$PRSS_{\lambda}^{ridge}(\beta) = \sum_{i=1}^{n}\left(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j\right)^2 + \lambda \sum_{j=1}^{p} \beta_j^2$$

Penalized regression method,
RSS + penalty

- No variable selection: all β's are non-zero.
- Coefficients of correlated variables are shrunken toward each other.



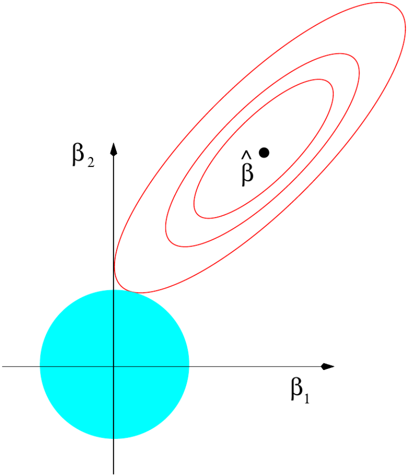
Ridge regression

$$PRSS_{\lambda}^{ridge}(\beta) = \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j \right)^2 + \lambda \sum_{j=1}^{p} \beta_j^2$$

Penalized regression method,
RSS + penalty

- No variable selection: all β's are non-zero.
- Coefficients of correlated variables are shrunken toward each other.

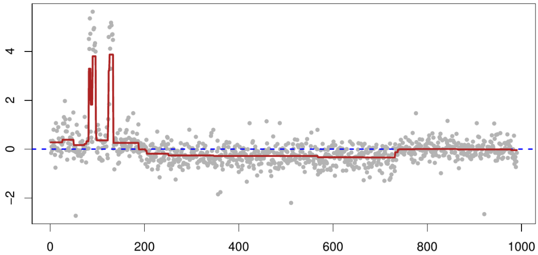
Ridge and lasso combined in elastic-net (sum of both penalties).



The future!

The future of penalized regression:

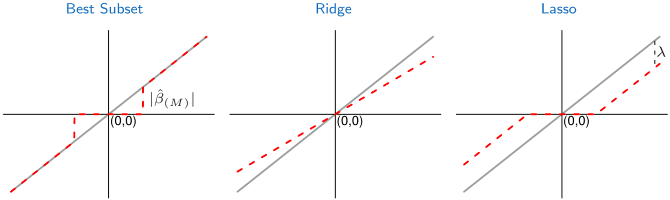
- Fused lasso for functional data: $\lambda \sum_{j=1}^{p-1} |\beta_{j+1} - \beta_j|$
- Group lasso for covariates in predefined groups.
- SLOPE (Sorted L-One Penalized Estimation), lasso with multiple testing



Best subset selection: Find the subset of size k with smallest RSS (up to $p \leq 30 \sim 40$), and select k with crossvalidation.

Best subset selection: Find the subset of size k with smallest RSS (up to $p \leq 30 \sim 40$), and select k with crossvalidation.

Estimator	Formula
Best subset (size M)	$\hat{\beta}_j \cdot I(\lvert\hat{\beta}_j\rvert \geq \lvert\hat{\beta}_{(M)}\rvert)$
Ridge	$\hat{\beta}_j/(1+\lambda)$
Lasso	$\text{sign}(\hat{\beta}_j)(\lvert\hat{\beta}_j\rvert - \lambda)_+$



For orthogonal design: $p = n, \mathbf{X}^T\mathbf{X} = I$, the methods can be seen as different *thresholding* of the OLS estimates.

PCR/PLS

Principal component regression (PCR); uses the m first principal components $\mathbf{D}_m\mathbf{U}_m$ ($\mathbf{X} = \mathbf{UDV}^T$)

$$\hat{\beta}_m^{PCR} = \mathbf{V}_m\hat{\theta},$$

where $\hat{\theta}$ is the OLS estimate based on $\mathbf{D}_m\mathbf{U}_m$.

PCR/PLS

Principal component regression (PCR); uses the m first principal components $\mathbf{D}_m\mathbf{U}_m$ ($\mathbf{X} = \mathbf{UDV}^T$)

$$\hat{\beta}_m^{PCR} = \mathbf{V}_m\hat{\theta},$$

where $\hat{\theta}$ is the OLS estimate based on $\mathbf{D}_m\mathbf{U}_m$. Connection to ridge: both are linear in $\mathbf{y}$ and shrink $\mathbf{u}_i^T\mathbf{y}$ differently.

PCR/PLS

Principal component regression (PCR); uses the m first principal components $\mathbf{D}_m\mathbf{U}_m$ ($\mathbf{X} = \mathbf{UDV}^T$)

$$\hat{\beta}_m^{PCR} = \mathbf{V}_m\hat{\theta},$$

where $\hat{\theta}$ is the OLS estimate based on $\mathbf{D}_m\mathbf{U}_m$. Connection to ridge: both are linear in $\mathbf{y}$ and shrink $\mathbf{u}_i^T\mathbf{y}$ differently.

Partial least squares (PLS); constructs linear combinations of inputs, but based on the correlation with response $\mathbf{y}$,

PCR/PLS

Principal component regression (PCR); uses the m first principal components $\mathbf{D}_m\mathbf{U}_m$ ($\mathbf{X} = \mathbf{UDV}^T$)

$$\hat{\beta}_m^{PCR} = \mathbf{V}_m\hat{\theta},$$

where $\hat{\theta}$ is the OLS estimate based on $\mathbf{D}_m\mathbf{U}_m$. Connection to ridge: both are linear in $\mathbf{y}$ and shrink $\mathbf{u}_i^T\mathbf{y}$ differently.

Partial least squares (PLS); constructs linear combinations of inputs, but based on the correlation with response $\mathbf{y}$,

- the PLS directions are iteratively calculated by algorithm.
- Solution is nonlinear function of $\mathbf{y}$.

Linear models - classification

Linear classification method if decision boundary is linear

- Treat as regression problem

Linear models - classification

Linear classification method if decision boundary is linear

- Treat as regression problem
  - Linear regression

Linear models - classification

Linear classification method if decision boundary is linear

- Treat as regression problem
  - Linear regression
  - Logistic regression

Linear models - classification

Linear classification method if decision boundary is linear

- Treat as regression problem
  - Linear regression
  - Logistic regression
- Modeling of $p(\boldsymbol{x}|y)$ through Bayes classifier
  - Linear Discriminant Analysis (QDA)

Linear models - classification

Linear classification method if decision boundary is linear

- Treat as regression problem
  - Linear regression
  - Logistic regression
- Modeling of $p(\boldsymbol{x}|y)$ through Bayes classifier
  - Linear Discriminant Analysis (QDA)
  - Quadratic Discriminant Analysis (LDA)

Linear models - classification

Linear classification method if decision boundary is linear

- Treat as regression problem
  - Linear regression
  - Logistic regression
- Modeling of $p(\boldsymbol{x}|y)$ through Bayes classifier
  - Linear Discriminant Analysis (QDA)
  - Quadratic Discriminant Analysis (LDA)
- Direct search for decision boundary
  - Separating hyperplanes: percetrons, support vector machines

Discriminant Analysis

Assuming Gaussian densities for each class k

Discriminant Analysis

Assuming Gaussian densities for each class k

$$f_k(x) = \frac{1}{(2\pi)^{p/2}|\Sigma_k|^{p/2}} \exp\left\{-\frac{1}{2}(x-\mu_k)^T \Sigma_k^{-1}(x-\mu_k)\right\},$$

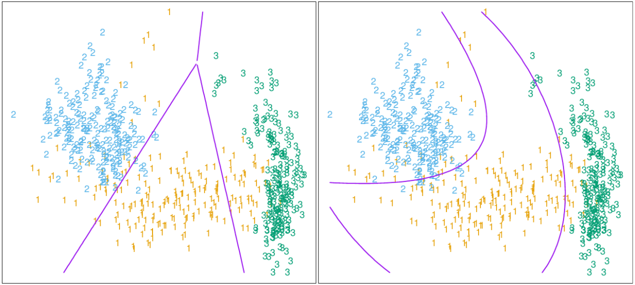
in Bayes classifier gives QDA and LDA.

Discriminant Analysis

Assuming Gaussian densities for each class k

$$f_k(x) = \frac{1}{(2\pi)^{p/2}|\Sigma_k|^{p/2}} \exp\left\{-\frac{1}{2}(x-\mu_k)^T\Sigma_k^{-1}(x-\mu_k)\right\},$$

in Bayes classifier gives QDA and LDA. Does not require normality, but will be optimal in the sense of the Bayes classifier.



Beyond lineary: Basis expansions

- Decompose in basis functions $f(\boldsymbol{x}) = \sum_{m=1}^{M} \beta_m h_m(\boldsymbol{x})$

Beyond lineary: Basis expansions

- Decompose in basis functions $f(\boldsymbol{x}) = \sum_{m=1}^{M} \beta_m h_m(\boldsymbol{x})$
- Different h_m functions:
  - Piecewise polynomials
  - Splines
  - Sigmaoids (Neural network)
  - Piecewise constant (trees)

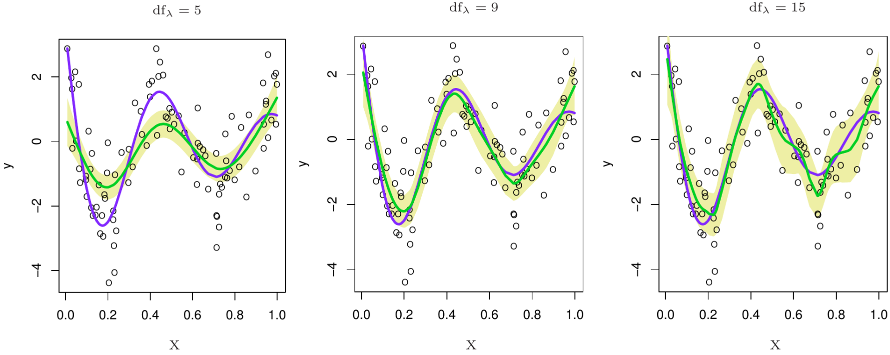
Beyond lineary: Basis expansions

- Decompose in basis functions $f(\boldsymbol{x}) = \sum_{m=1}^{M} \beta_m h_m(\boldsymbol{x})$
- Different h_m functions:
  - Piecewise polynomials
  - Splines
  - Sigmaoids (Neural network)
  - Piecewise constant (trees)
- Restrictions on model complexity
  - Effective number of parameters (splines, GAM)
  - Generalized crossvalidation
  - Ridge-type (Neural network)

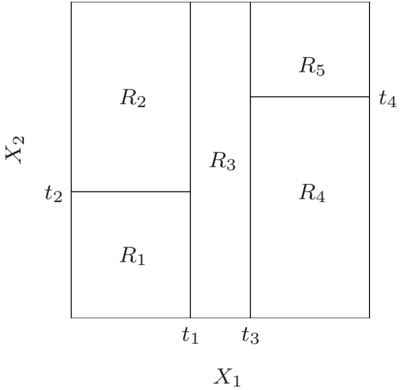
Smoothing splines

$$PRSS(f, \lambda) = \sum_{i=1}^{N}(y_i - f(x_i)^2 + \lambda \int (f''(t))^2 \, dt,$$

where the smoothing parameter λ controls the fit to the data.

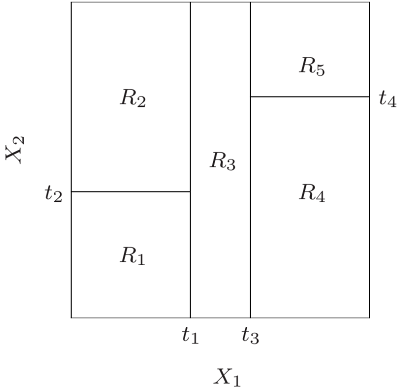


Trees



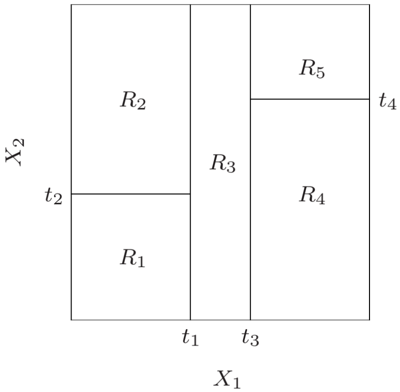
- $\mathcal{R}^p = R_1 \cup R_2 \cup \cdots \cup R_M$

Trees



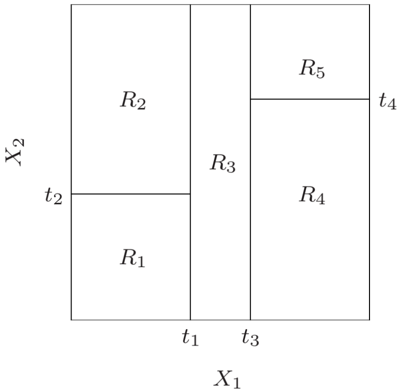
- $\mathcal{R}^p = R_1 \cup R_2 \cup \cdots \cup R_M$
- Model
  $f(\boldsymbol{x}) = \sum_{m=1}^{M} c_m I(\boldsymbol{x} \in R_m)$

Trees



- $\mathcal{R}^p = R_1 \cup R_2 \cup \cdots \cup R_M$
- Model
  $f(\mathbf{x}) = \sum_{m=1}^{M} c_m I(\mathbf{x} \in R_m)$
- Too flexible model, restrictions through
  - Sequential definition of R_m's
  - Splitting only through one variable at a time.
  - Pruning of tree

Trees



- $\mathcal{R}^p = R_1 \cup R_2 \cup \cdots \cup R_M$
- Model
  $f(\mathbf{x}) = \sum_{m=1}^{M} c_m I(\mathbf{x} \in R_m)$
- Too flexible model, restrictions through
  - Sequential definition of R_m's
  - Splitting only through one variable at a time.
  - Pruning of tree
- Classification:
  $\Pr(y = k|\mathbf{x}) = p_{m(\mathbf{x}),k}$ where $\mathbf{x} \in R_{m(\mathbf{x})}$. Estimate
  $\hat{p}_{m,k} = \frac{1}{N_m} \sum_{\mathbf{x}_i \in R_m} I(y_i = k)$

Boosting

- Forward stagewise additive modeling

Boosting

- Forward stagewise additive modeling
- Fitting many simple functions or base learner

$$f_M(x) = \sum_{i=1}^{M} \beta_m b(x, \gamma_m),$$

- by sequentially adding new basis functions to the expansion without adjusting earlier included parameter/coefficients.

Boosting

- Forward stagewise additive modeling
- Fitting many simple functions or base learner

$$f_M(x) = \sum_{i=1}^{M} \beta_m b(x, \gamma_m),$$

- by sequentially adding new basis functions to the expansion without adjusting earlier included parameter/coefficients.
- For squared error loss the basis function explaining the current residual is added at each iteration.

Boosting

- Forward stagewise additive modeling
- Fitting many simple functions or base learner

$$f_M(x) = \sum_{i=1}^{M} \beta_m b(x, \gamma_m),$$

- by sequentially adding new basis functions to the expansion without adjusting earlier included parameter/coefficients.
- For squared error loss the basis function explaining the current residual is added at each iteration.
- Boosting: improve the learners in an adaptive way to (slowly) remove bias.

Boosting

- Forward stagewise additive modeling
- Fitting many simple functions or base learner

$$f_M(x) = \sum_{i=1}^{M} \beta_m b(x, \gamma_m),$$

- by sequentially adding new basis functions to the expansion without adjusting earlier included parameter/coefficients.
- For squared error loss the basis function explaining the current residual is added at each iteration.
- Boosting: improve the learners in an adaptive way to (slowly) remove bias.
- Typical algorithms: AdaBoost for classification, GradientBoost for regression

Boosting

- Forward stagewise additive modeling
- Fitting many simple functions or base learner

$$f_M(x) = \sum_{i=1}^{M} \beta_m b(x, \gamma_m),$$

- by sequentially adding new basis functions to the expansion without adjusting earlier included parameter/coefficients.
- For squared error loss the basis function explaining the current residual is added at each iteration.
- Boosting: improve the learners in an adaptive way to (slowly) remove bias.
- Typical algorithms: AdaBoost for classification, GradientBoost for regression
- Boosting with shrinkage use implicit lasso-style penalty

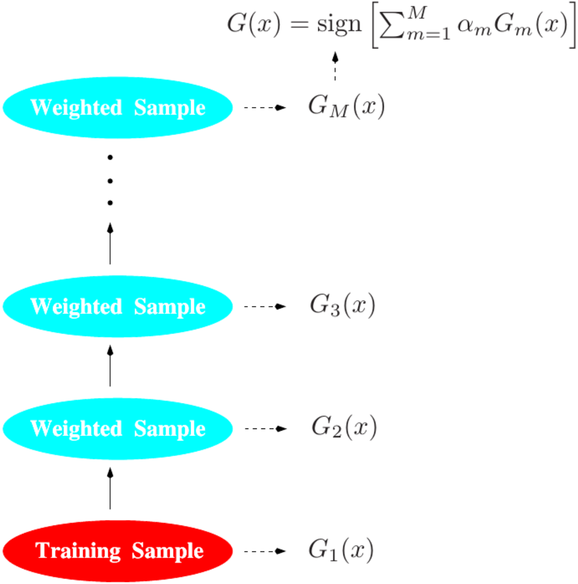


FIGURE 10.1. *Schematic of AdaBoost. Classifiers are trained on weighted versions of the dataset, and then combined to produce a final prediction.*

Bagging

- Bootstrap aggregation/bagging builds predictions $\hat{f}^{*b}$ over bootstrap samples (resampled with replacement),

Bagging

- Bootstrap aggregation/bagging builds predictions $\hat{f}^{*b}$ over bootstrap samples (resampled with replacement),
- and averages

$$\hat{f}_{bag}(x_i) = \frac{1}{B}\sum_{b=1}^{B} \hat{f}^{*b}(x),$$

only to reduce the variance of the prediction.

Bagging

- Bootstrap aggregation/bagging builds predictions $\hat{f}^{*b}$ over bootstrap samples (resampled with replacement),
- and averages
$$\hat{f}_{bag}(x_i) = \frac{1}{B}\sum_{b=1}^{B} \hat{f}^{*b}(x),$$
only to reduce the variance of the prediction.
- Improves the variance of nonlinear, high variance and low bias learners, such as trees.

Random forests

The variance of bagged predictions is trade-off between variance and correlation:

$$\text{var}\left[\hat{f}_{bag}(x)\right] = \rho\sigma^2 + \frac{1-\rho}{B}\sigma^2$$

- Bagged or bootstrap trees are (highly) correlated.
- Idea behind random forests: reduce the correlation without increasing the variance.
- Achieved through random selection of the input variables.

Random forests

The variance of bagged predictions is trade-off between variance and correlation:

$$\text{var}\left[\hat{f}_{bag}(x)\right] = \rho\sigma^2 + \frac{1-\rho}{B}\sigma^2$$

- Bagged or bootstrap trees are (highly) correlated.
- Idea behind random forests: reduce the correlation without increasing the variance.
- Achieved through random selection of the input variables.
- Before each split, select $m \leq p$ (typically $\sqrt{p}$ or $p/3$) of the input variables at random as candidates for splitting.

What have we not covered!

- Kernel smoothing and regression

What have we not covered!

- Kernel smoothing and regression
- BIC and bootstraping

What have we not covered!

- Kernel smoothing and regression
- BIC and bootstraping
- Baysian methods, EM algorithm, MCMC

What have we not covered!

- Kernel smoothing and regression
- BIC and bootstraping
- Baysian methods, EM algorithm, MCMC
- Generalized additive models

What have we not covered!

- Kernel smoothing and regression
- BIC and bootstraping
- Baysian methods, EM algorithm, MCMC
- Generalized additive models
- Neural networks

What have we not covered!

- Kernel smoothing and regression
- BIC and bootstraping
- Baysian methods, EM algorithm, MCMC
- Generalized additive models
- Neural networks
- Support vector machines

What have we not covered!

- Kernel smoothing and regression
- BIC and bootstraping
- Baysian methods, EM algorithm, MCMC
- Generalized additive models
- Neural networks
- Support vector machines
- Nearest-neighbors

What have we not covered!

- Kernel smoothing and regression
- BIC and bootstraping
- Baysian methods, EM algorithm, MCMC
- Generalized additive models
- Neural networks
- Support vector machines
- Nearest-neighbors
- Cluster analysis

What have we not covered!

- Kernel smoothing and regression
- BIC and bootstraping
- Baysian methods, EM algorithm, MCMC
- Generalized additive models
- Neural networks
- Support vector machines
- Nearest-neighbors
- Cluster analysis
- Independent component analysis

What have we not covered!

- Kernel smoothing and regression
- BIC and bootstraping
- Baysian methods, EM algorithm, MCMC
- Generalized additive models
- Neural networks
- Support vector machines
- Nearest-neighbors
- Cluster analysis
- Independent component analysis
- ....

Explanation versus prediction

- In this course: Focus on prediction

Explanation versus prediction

- In this course: Focus on prediction
- Explanation: Which covariates are important?

Explanation versus prediction

- In this course: Focus on prediction
- Explanation: Which covariates are important?
- Much more difficult question

Explanation versus prediction

- In this course: Focus on prediction
- Explanation: Which covariates are important?
- Much more difficult question
- Often: Prediction performance used as criterion for evaluating importance of covariate

Explanation versus prediction

- In this course: Focus on prediction
- Explanation: Which covariates are important?
- Much more difficult question
- Often: Prediction performance used as criterion for evaluating importance of covariate
- Problems:
  - Lack of predictive power might be due to too little data.
  - Predictive power may be because of indirect influence through other covariates