Regularisation

Machine Learning for Process Engineers Workshop

Stellenbosch University

March 2022

Recap

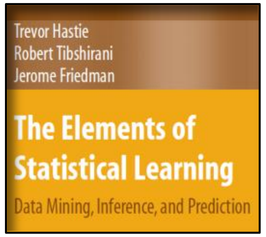
- Linear regression so far

$$\hat{y}(\mathbf{x}_i) = \sum_{j=0}^{p} x_{i,j}\beta_j = \mathbf{x}_i\boldsymbol{\beta}$$

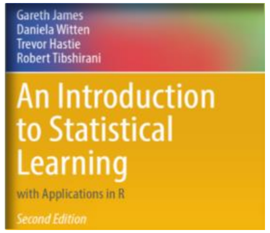
- Flexible models $\rightarrow$ more parameters $\rightarrow$ increased variance
  $\rightarrow$ reduced performance (sometimes)

- Feature selection: procedures to select relevant features
  - Forward selection: find the predictor $\mathbf{X}_j$ which most improves the model

Feature selection



p 57



p 227

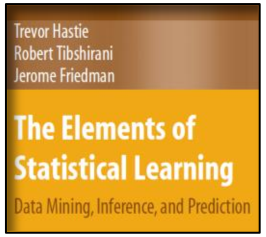
- Forward selection:
  1. Start with a constant term, $\hat{y}(\mathbf{x}_i) = \beta_0$ and the set $\mathcal{J}_0 = \{1\}$
  2. Find the predictor $\mathbf{X}_j$ which most improves the model according to a criterion
  3. Add predictor $\mathbf{X}_j$ to the set $\mathcal{J}_k = \mathcal{J}_{k-1} \cup j$, then refit the model such that

$$y(\mathbf{x}_i) = \sum_{j \in \mathcal{J}_k} \beta_j x_{i,j}$$

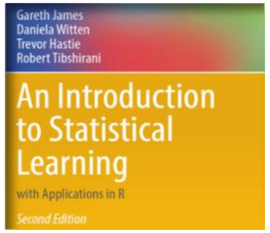
  4. Keep adding predictors until the criterion is optimized

- Typical criterion: adjusted R^2, Akaike Information Criterion, Bayesian Information Criterion, etc.

Feature selection



p 57



p 227

- Backwards selection:
  1. Start with a full model, $\hat{y}(\mathbf{x}_i) = \sum_{j=0}^{p} x_{i,j}\beta_j$ and the set $\mathcal{J}_0 = \{1,2,3 \dots p\}$
  2. Find j such that removing predictor $\mathbf{X}_j$ most improves the model according to a criterion
  3. Remove predictor $\mathbf{X}_j$ from the set $\mathcal{J}_k = \mathcal{J}_{k-1} \backslash j$ and refit the model such that
  $$y(\mathbf{x}_i) = \sum_{j \in \mathcal{J}_k} \beta_j x_{i,j}$$
  4. Keep removing predictors until the criterion is optimized

- Many hybrid version exist, see "Elements of Statistical Learning" (2009)and "Introduction to Statistical Learning" (2021) for a full treatment

In MATLAB: Example 6

- Open file "`MLforProcEng_Workshop_2.m`" and run the "**`%% Initialize`**" cell
- Go to the cell

  `%% Example 6:  Use feature selection…`

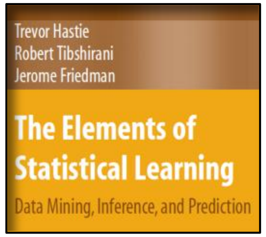
- The approach is exactly as before when fitting a p-order polynomial, but now replace "`fitlm`" with "`stepwiselm`"

In MATLAB: Example 6

```
Linear regression model:
    y ~ 1 + x1 + x3 + x5 + x7 + x9 + x11

Estimated Coefficients:
```

	Estimate	SE	tStat	pValue
(Intercept)	-0.013948	0.021743	-0.64149	0.52278
x1	5.9735	0.14314	41.733	5.0907e-62
x3	-6.4442	0.35421	-18.193	2.1442e-32
x5	2.9958	0.28601	10.475	2.0406e-17
x7	-0.7102	0.097203	-7.3064	9.2697e-11
x9	0.082462	0.014253	5.7855	9.6604e-08
x11	-0.0036621	0.00073734	-4.9667	3.0823e-06

Shifting to a different set of basis function



p 139, 212

- Polynomial regression

$$\hat{y}(t) = \beta_0 + \beta_1 t + \beta_2 t^2 \ldots \beta_p t^p = \sum_{j=0}^{p} t^j \beta_j = [1 \;\; t \;\; t^2 \;\; \ldots t^p]\boldsymbol{\beta}$$

- Orders of t were used as features, but any function $h(t)$ can be used

$$\hat{y}(t) = \sum_{j=1}^{p} h_j(t)\beta_j = \mathbf{x}\boldsymbol{\beta}$$

$$\mathbf{x} = \left[1 \;\; h_1(t) \;\; h_2(t) \;\; h_3(t) \;\; \ldots h_p(t)\right]$$

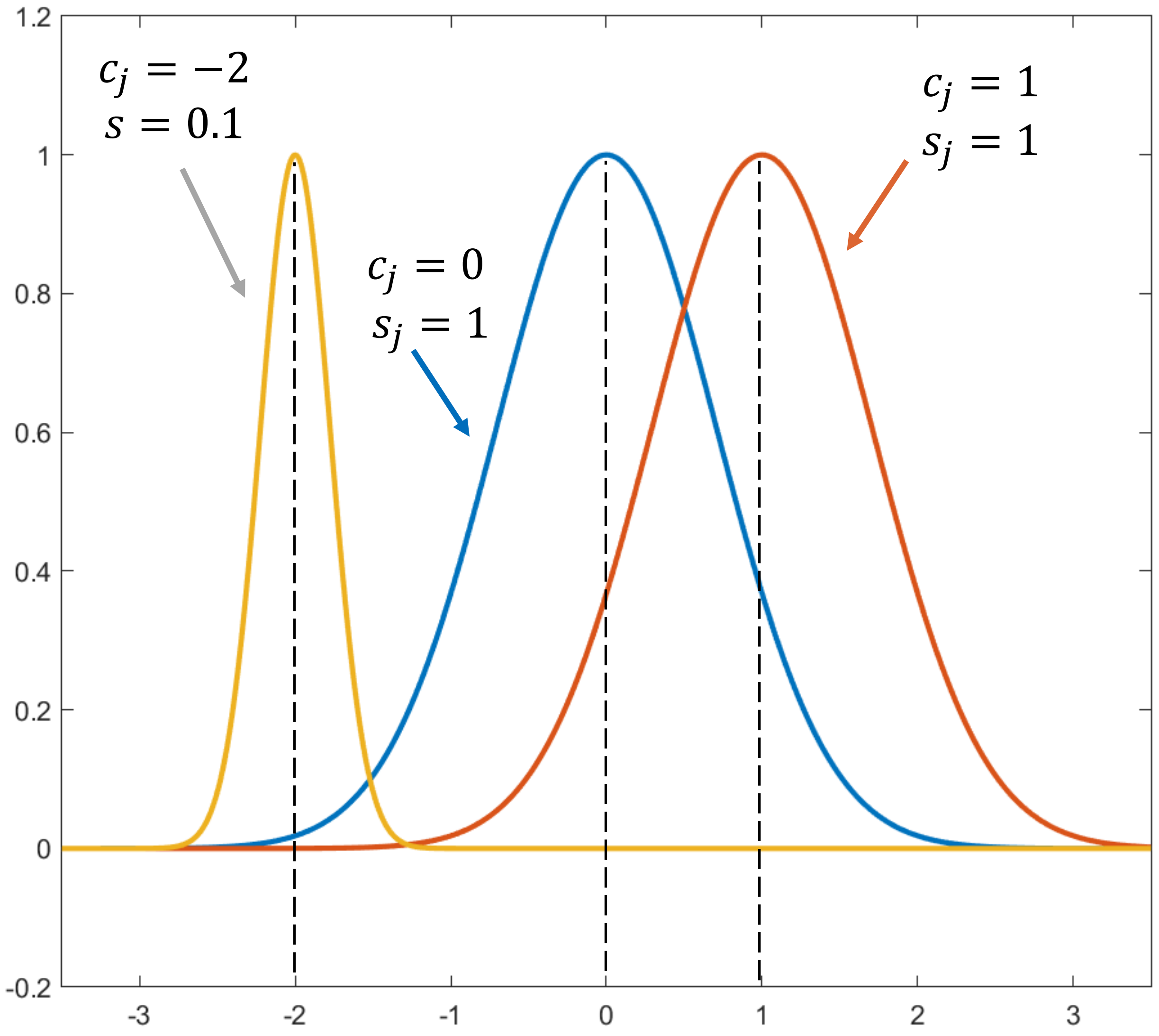
$$\hat{\mathbf{y}} = \mathbf{X}\boldsymbol{\beta}$$

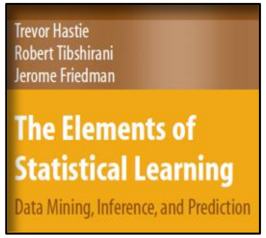
Shifting to a different set of basis functions

Commonly used basis functions: Gaussian with centroid c_j and shape factor s

$$h_j(t) = \exp\left(-\frac{(t-c_j)}{s_j}\right)^2$$

- For linear regression: the centroid locations and shape factors are pre-specified (not learned by data).





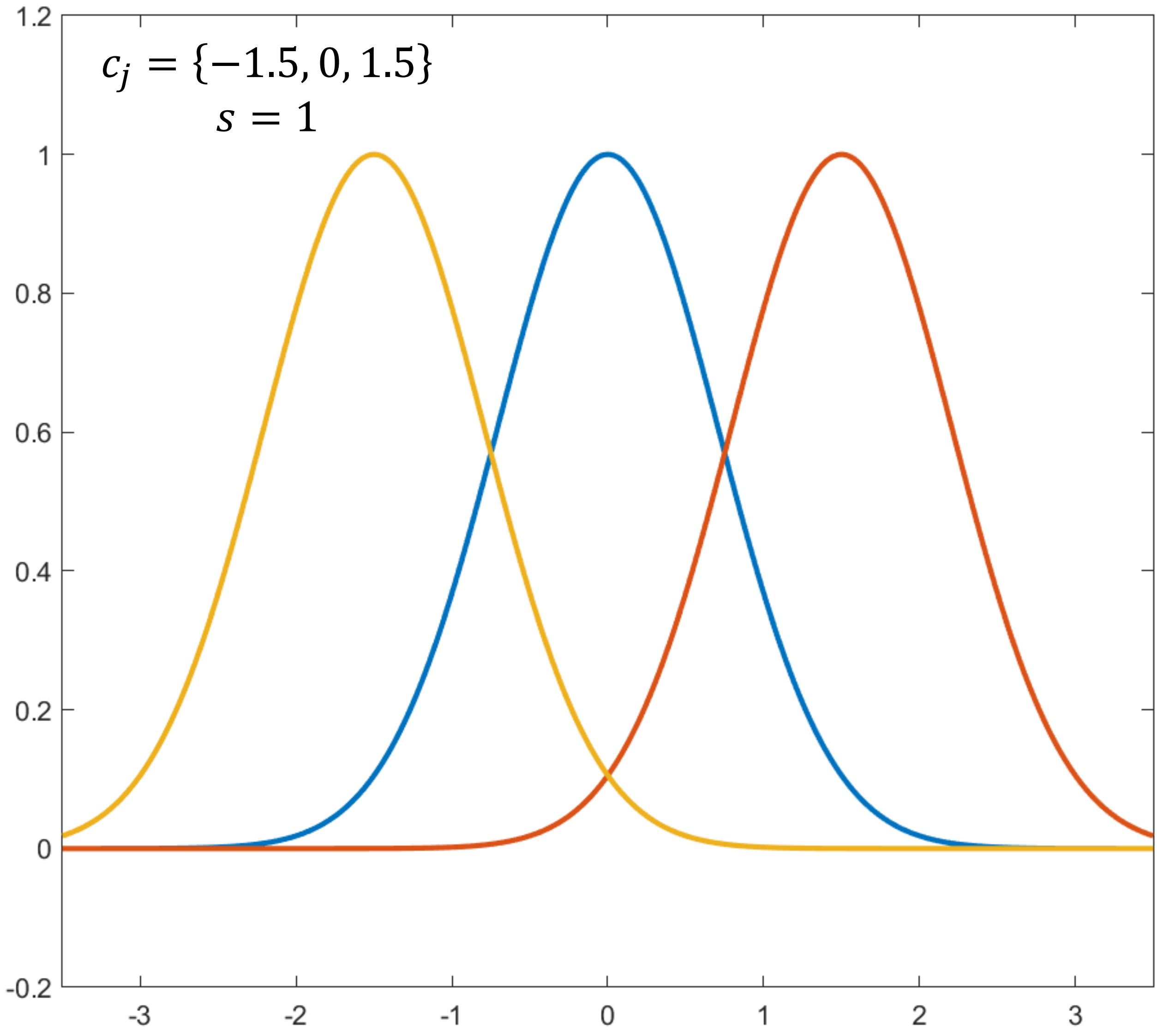
p 139, 212

Shifting to a different set of basis functions

Commonly used basis functions: Gaussian with centroid c_j and shape factor s

$$h_j(t) = \exp\left(-\frac{(t-c_j)}{s_j}\right)^2$$

- For linear regression: the centroid locations and shape factors are pre-specified (not learned by data).
- It is common for a single shape factor to be used, $s_j = s \, \forall \, j$ with equally spaced centroids

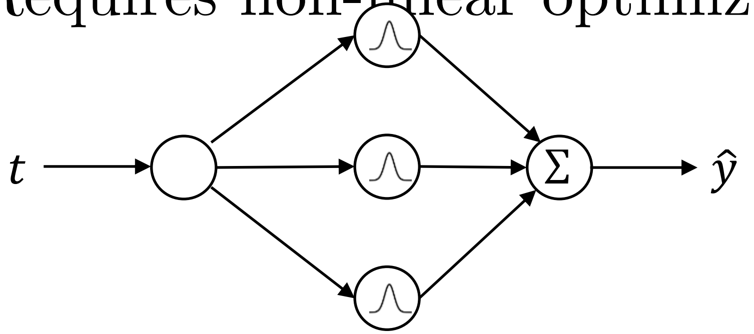


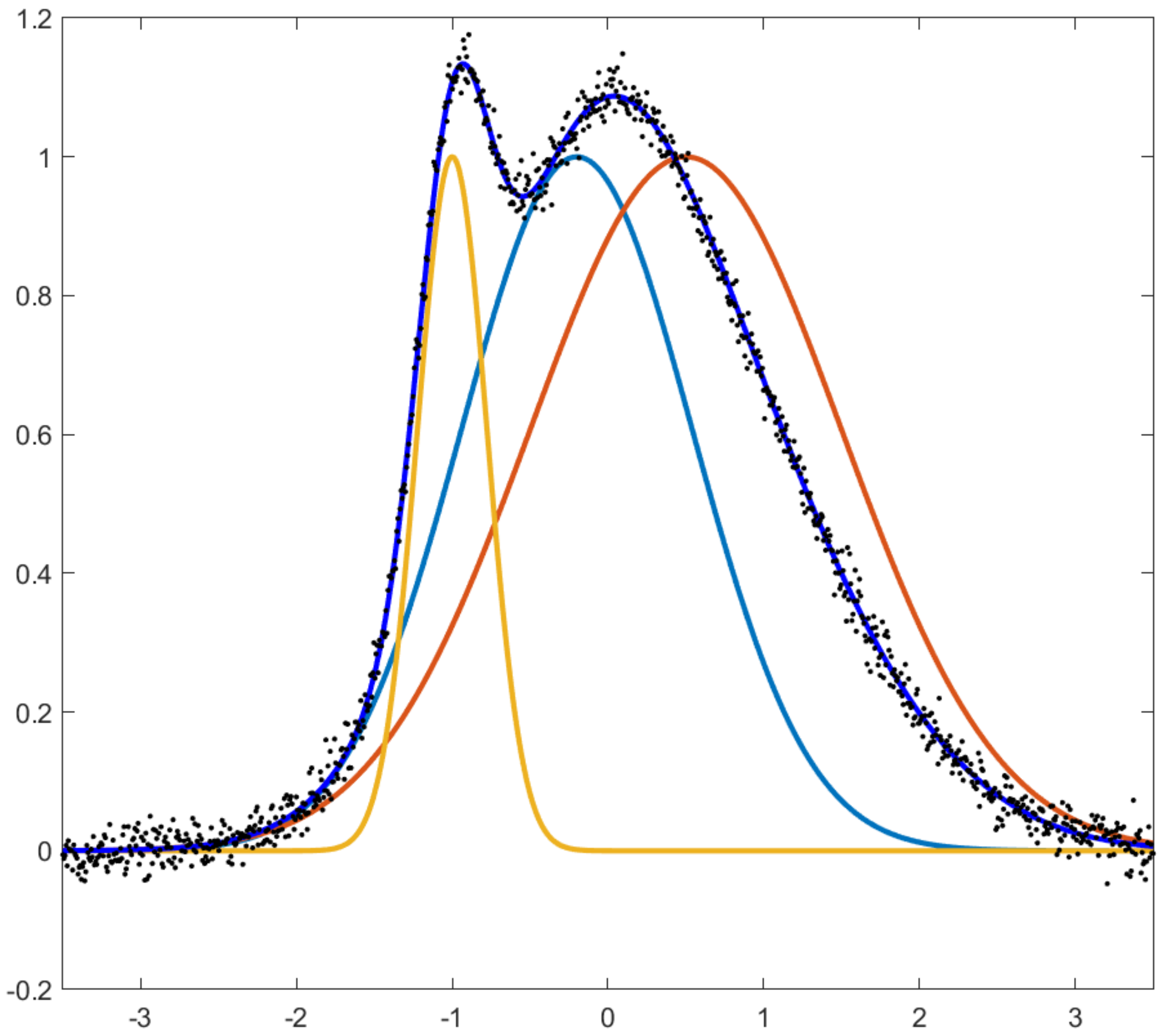
Shifting to a different set of basis functions

Commonly used basis functions: Gaussian with centroid c_j and shape factor s

$$h_j(t) = \exp\left(-\frac{\left(t - c_j\right)}{s_j}\right)^2$$

- Allowing the centroid and shape factors to be learnt
  - Radial basis function neural network
  - Requires non-linear optimization





In MATLAB: Example 7

- Go to the cell

  ```
  % Example 7:  Use an alternative model…
  ```

- The custom function "`CreateGaussDesignMatrix(t, c)`" will generate the design matrix at data points $t_1, t_2 \ldots t_N$ with centroids $c_1, c_2, \ldots c_p$ and a defaults shape factor s

$$\mathbf{X} = \begin{bmatrix} h_1(t_1) & h_2(t_1) & \ldots & h_p(t_1) \\ h_1(t_2) & h_2(t_2) & \ldots & h_p(t_2) \\ \vdots & \vdots & \ddots & \vdots \\ h_1(t_N) & h_2(t_N) & \ldots & h_p(t_N) \end{bmatrix}$$

- Create the design matrix using the new basis functions, then fit the model exactly as before using "`fitlm`". Plot the model using "`predict`" at the points "`Fit.t`"

In Python: Example 7

- Go to the cell

#%% Example 7: Use an alternative model using "Gaussian" radial basis functions (G-RBFs)

The custom function "`CreateGaussDesignMatrix(t, c)`" will generate the design matrix at data points $t_1, t_2 \dots t_N$ with centroids $c_1, c_2, \dots c_p$ and a defaults shape factor s

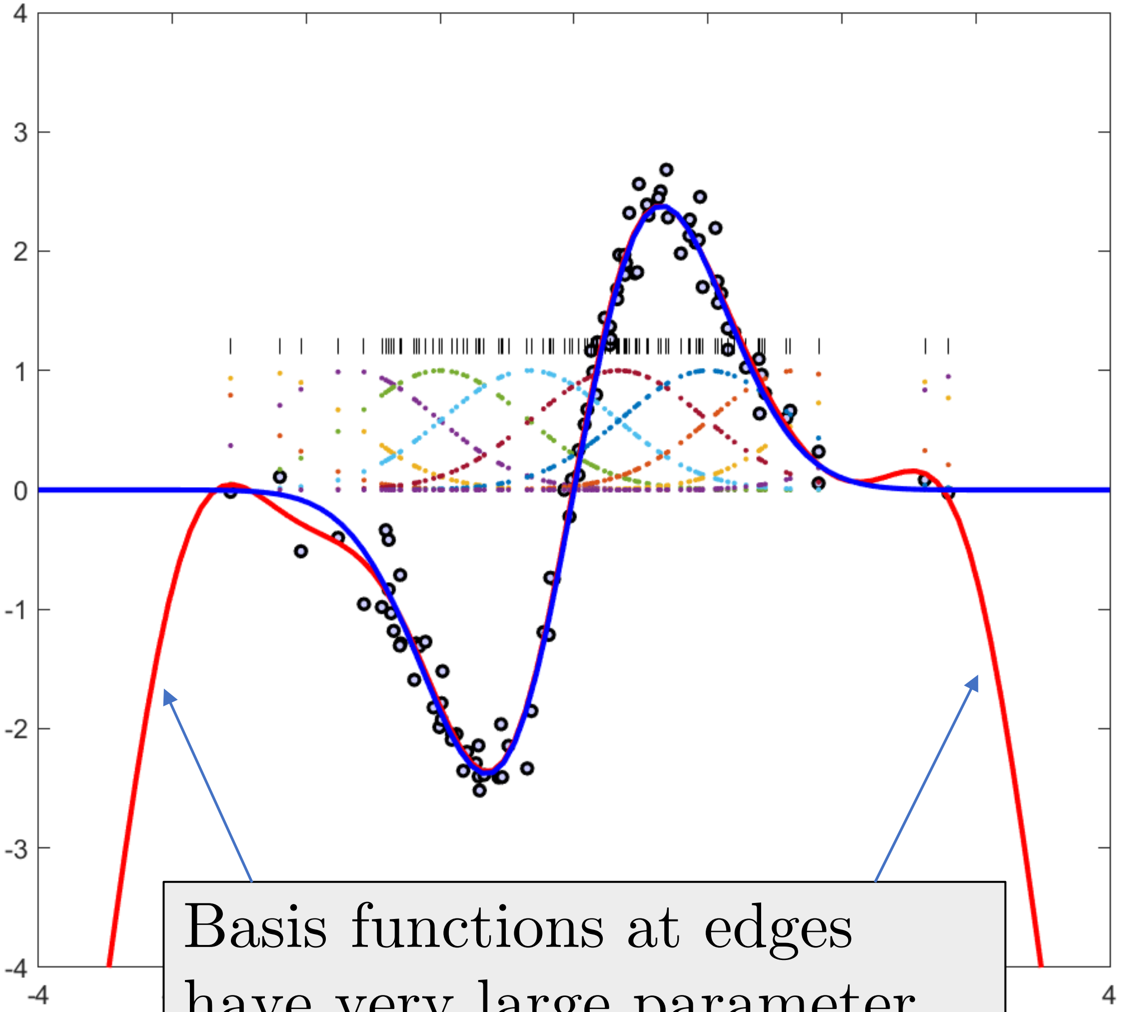
$$\mathbf{X} = \begin{bmatrix} h_1(t_1) & h_2(t_1) & \dots & h_p(t_1) \\ h_1(t_2) & h_2(t_2) & \dots & h_p(t_2) \\ \vdots & \vdots & \ddots & \vdots \\ h_1(t_N) & h_2(t_N) & \dots & h_p(t_N) \end{bmatrix}$$

- Create the design matrix using the new basis functions, then fit the model exactly as before using "`linear_model.LinearRegression().fit(X_train, Data.y)`".

- Plot the model using "`predict`" at the points "`Fit.t`"

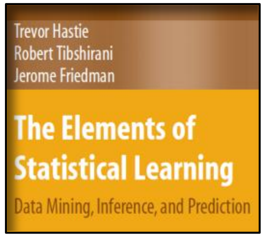
In MATLAB: Example 7

```
%% Example 7:  Use an alternative model…
clf
Data = GenerateData(f, sig_eps, 100,true,…
                    [-4 4 -4 4],'on');

c = linspace(-3, 3, 10);
X_train = CreateGaussDesignMatrix(Data.t, c);
⋮

% Fit the model using linear regression
mdl = fitlm(X_train, Data.y);

% Evaluate the function at the equally spaced
% "Fit.t" points and plot the function
X_fit = CreateGaussDesignMatrix(Fit.t, c);
Fit.RBF = predict(mdl, X_fit);
plot(Fit.t, Fit.RBF, 'r', Fit.t, Fit.f, 'b',…
'LineWidth', 2);
```

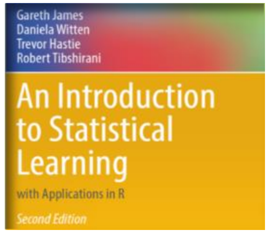


Basis functions at edges have very large parameter values, trying to fit noisy data points

Shrinkage methods (regularization)



p 61



p 237

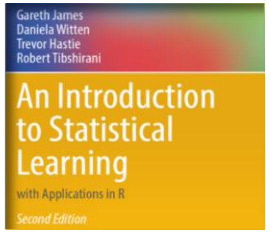
- Large coefficients → high variance
- Introduce a penalty on large coefficients
- New objective function to minimize:

$$\hat{\boldsymbol{\beta}} = \arg\min_{\boldsymbol{\beta}} \left[\sum_{i=1}^{N} L(y_i, \hat{y}(\mathbf{x}_i, \boldsymbol{\beta}) + \lambda \sum_{j=1}^{p} \beta_j^2 \right]$$

- The squared penalty (above) often referred to as the L_2 penalty
- Optimization is called “ridge regression”

Shrinkage methods (regularization)

p 179



p 248

Theoretical basis?

$$\widehat{\boldsymbol{\beta}} = \arg\min_{\boldsymbol{\beta}} \left[\sum_{i=1}^{N} L(y_i, \hat{y}(\mathbf{x}_i, \boldsymbol{\beta}) + \lambda \sum_{j=1}^{p} \beta_j^2 \right]$$

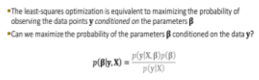
- Recall: the squared error originates from the assumption of normally distributed ε
- The least-squares optimization is equivalent to maximizing the likelihood of the parameters, that is maximizing the probability of observing the data points $\mathbf{y}$ *conditioned on* the parameters $\boldsymbol{\beta}$

$$\ln p(\mathbf{y}|\mathbf{X}; \boldsymbol{\beta}) \propto -\frac{1}{2\sigma^2} \sum_{i=1}^{N} \left(\boldsymbol{y}_i - \widehat{\boldsymbol{y}}(\boldsymbol{x}_i; \boldsymbol{\beta}) \right)^2$$

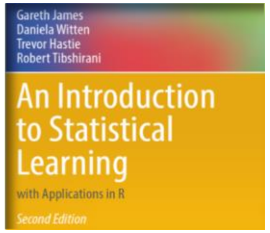
Shrinkage methods (regularization)

- The least-squares optimization is equivalent to maximizing the probability of observing the data points $\mathbf{y}$ *conditioned on* the parameters $\boldsymbol{\beta}$
- Can we maximize the probability of the parameters $\boldsymbol{\beta}$ conditioned on the data $\mathbf{y}$?

$$p(\boldsymbol{\beta}|\mathbf{y},\mathbf{X}) = \frac{p(\mathbf{y}|\mathbf{X},\boldsymbol{\beta})p(\boldsymbol{\beta})}{p(\mathbf{y}|\mathbf{X})}$$



p 179



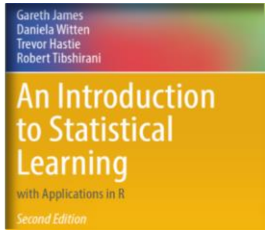
p 248

Shrinkage methods (regularization)

- The least-squares optimization is equivalent to maximizing the probability of observing the data points $\mathbf{y}$ *conditioned on* the parameters $\boldsymbol{\beta}$
- Can we maximize the probability of the parameters $\boldsymbol{\beta}$ conditioned on the data $\mathbf{y}$?

$$p(\boldsymbol{\beta}|\mathbf{y},\mathbf{X}) = \frac{p(\mathbf{y}|\mathbf{X},\boldsymbol{\beta})p(\boldsymbol{\beta})}{p(\mathbf{y}|\mathbf{X})}$$

p 179



p 248

Shrinkage methods (regularization)

- The least-squares optimization is equivalent to maximizing the probability of observing the data points $\mathbf{y}$ *conditioned on* the parameters $\boldsymbol{\beta}$
- Can we maximize the probability of the parameters $\boldsymbol{\beta}$ conditioned on the data $\mathbf{y}$?

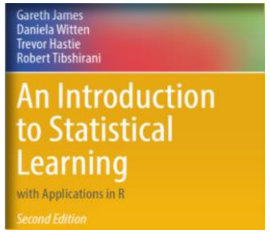
$$p(\boldsymbol{\beta}|\mathbf{y},\mathbf{X}) = \frac{p(\mathbf{y}|\mathbf{X},\boldsymbol{\beta})p(\boldsymbol{\beta})}{p(\mathbf{y}|\mathbf{X})}$$

- If we use

$$p(\boldsymbol{\beta}) = \mathcal{N}\left(0, \sigma_\beta I\right)$$

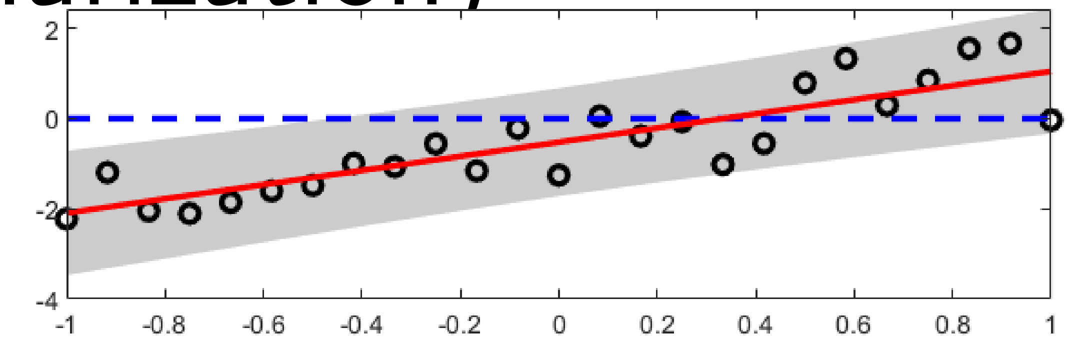
$$\ln p(\boldsymbol{\beta}|\mathbf{y},\mathbf{X}) \propto -\sum_{\boldsymbol{i}=\mathbf{1}}^{\boldsymbol{N}} \left(\boldsymbol{y_i} - \hat{\boldsymbol{y}}(\boldsymbol{x_i};\boldsymbol{\beta})\right)^{\mathbf{2}} - \frac{\sigma^2}{\sigma_\beta^2} \sum_{j=1}^{p} \beta_j^2$$

p 179



p 248

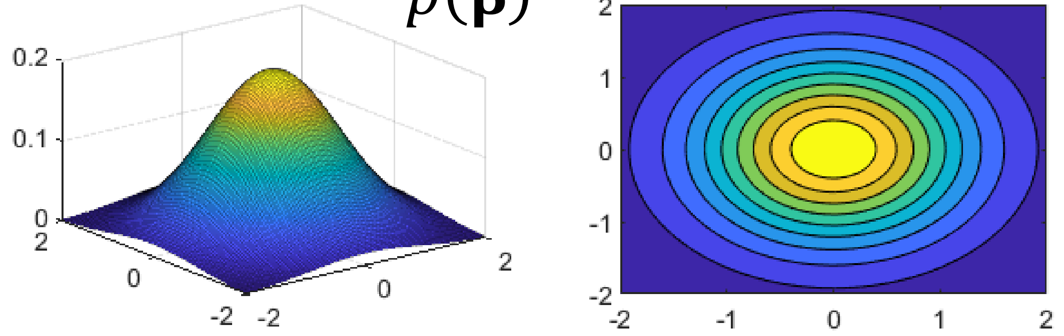
Shrinkage methods (regularization)



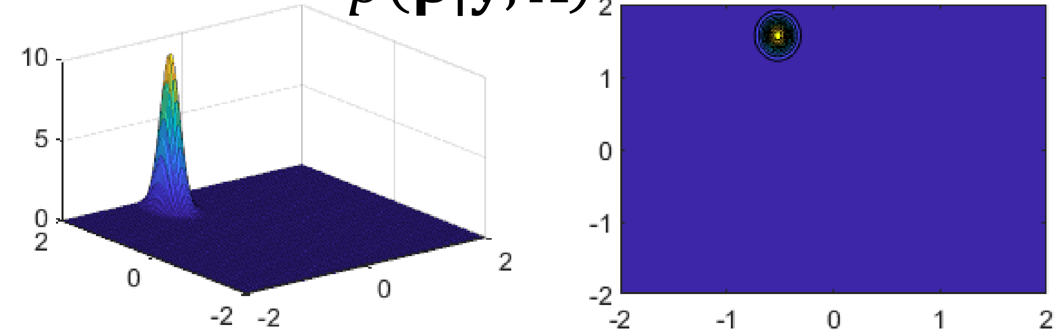
$$p(\boldsymbol{\beta}) = \mathcal{N}\left(0, \sigma_\beta I\right)$$

$$p(\boldsymbol{\beta}|\mathbf{y}, \mathbf{X}) = \frac{p(\mathbf{y}|\mathbf{X}, \boldsymbol{\beta})p(\boldsymbol{\beta})}{p(\mathbf{y}|\mathbf{X})}$$

$p(\boldsymbol{\beta})$



$p(\boldsymbol{\beta}|\mathbf{y}, \mathbf{X})$



The prior distribution $p(\boldsymbol{\beta})$ introduces bias into the model, while decreasing variance

(literally, biasing the model based on an existing knowledge of β)

Shrinkage methods (regularization)

$$\widehat{\boldsymbol{\beta}} = \arg\min_{\boldsymbol{\beta}} \left[\sum_{i=1}^{N} L(y_i, \hat{y}(\mathbf{x}_i, \boldsymbol{\beta}) + \lambda \sum_{j=1}^{p} \beta_j^2 \right]$$

- The regularization parameter λ biases the model towards $\beta_j \to \mathbf{0}$
- The increase in bias is associated with a decrease in variance
- Through λ, the amount of bias introduced can be varied gradually from $\lambda = \mathbf{0}$ (no bias) to $\lambda \to \infty$ (fixed model, no variance)

In MATLAB: Example 8

$$\widehat{\boldsymbol{\beta}} = \arg\min_{\boldsymbol{\beta}} \left[\sum_{i=1}^{N} (y_i - \mathbf{x}_i \boldsymbol{\beta}) + \lambda \sum_{j=1}^{p} \beta_j^2 \right]$$

- Go to the next cell

```
%% Example 8:  Regularize the G-RBF model using ridge regression
```

- Use "`lasso`" to fit a linear model to the data with regularization
  - `[beta, FitInfo] = lasso(X, y,'Lambda', 0.1,…`
    `'Alpha', 1e-6)`
  - `'Lambda', 0.1` sets the regularization parameter $\lambda = 0.1$
  - `'Alpha', 1e-6` ensures L_2 (squared) regularization, will be discussed next
  - `'Beta'` contains the parameters, excluding the intercept
  - `'FitInfo'` contains model information, incl. the intercept in `'FitInfo.Intercept'`

- Instead of "`predict(mdl, X_fit)`" to evaluate the model predictions, use

  "`FitInfo.Intercept + X_fit*beta`"

In Python: Example 8

$$\widehat{\boldsymbol{\beta}} = \arg\min_{\boldsymbol{\beta}} \left[\sum_{i=1}^{N} (y_i - \mathbf{x}_i \boldsymbol{\beta}) + \lambda \sum_{j=1}^{p} \beta_j^2 \right]$$

- Go to the next cell

  #%% Example 8: Regularize the G-RBF model using ridge regression

- Use "`linear_model.Ridge`" to fit a linear model to the data with regularization

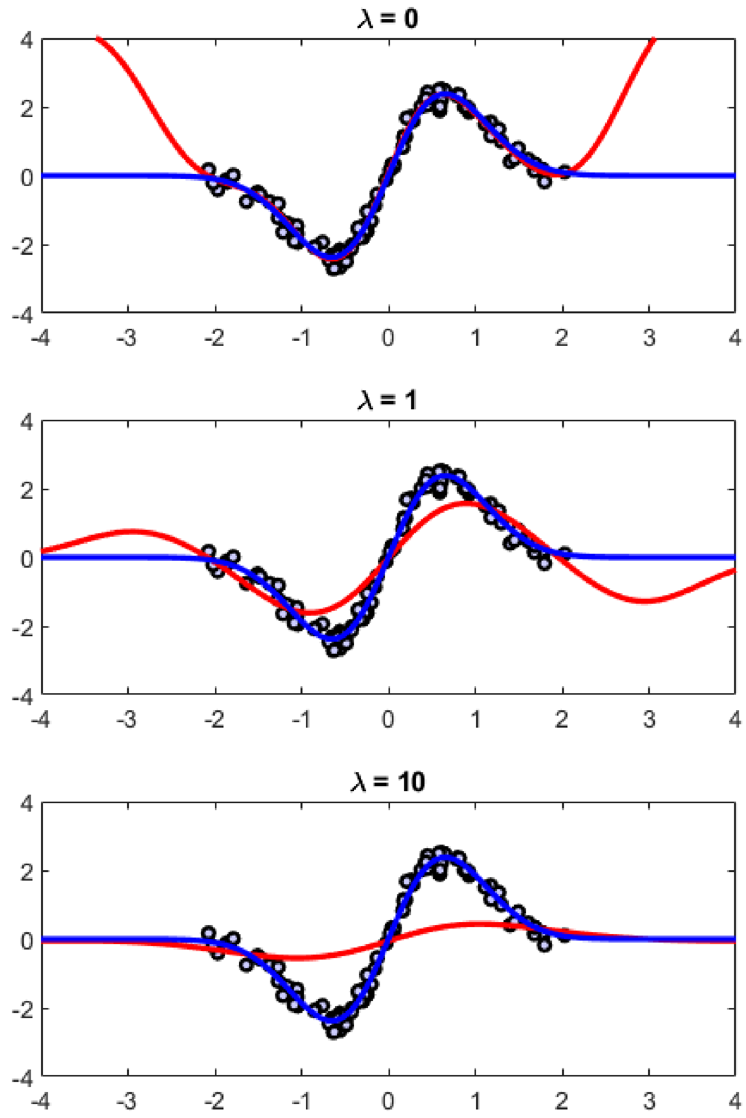
```
mdl = linear_model.Ridge(alpha = 0)
mdl.fit(X_train, Data.y)
```

  - `'Alpha', 0.1` sets the regularization parameter $\lambda = 0.1$

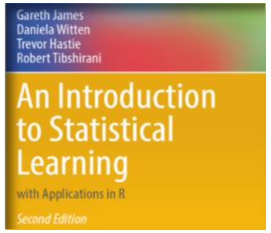
- Use "`mdl.predict(X_fit)`" as before

In MATLAB: Example 8

```
%% Example 8:  Regularize the G-RBF model…
clf
Data = GenerateData(f, sig_eps, 100,true,…
                    [-4 4 -4 4],'on');

c = linspace(-3, 3, 10);
X_train = CreateGaussDesignMatrix(Data.t, c);

[beta, FitInfo] = lasso(X_train, Data.y, …
                        'Lambda', 0.1, …
                        'Alpha', 1e-6);

beta0 = FitInfo.Intercept;

X_train = CreateGaussDesignMatrix(Fit.t, c);
Fit.RBF_ridge = beta0 + X_train*beta;

plot(Fit.t, Fit.RBF_ridge, 'r', …
     Fit.t, Fit.f, 'b', …
     'LineWidth', 2);
```



Shrinkage methods (regularization)

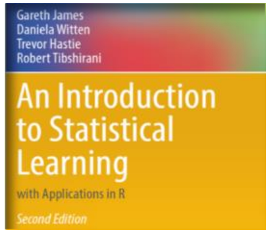


p 250

$$\widehat{\boldsymbol{\beta}} = \arg\min_{\boldsymbol{\beta}} \left[\sum_{i=1}^{N} L(y_i, \hat{y}(\mathbf{x}_i, \boldsymbol{\beta}) + \lambda \sum_{j=1}^{p} \beta_j^2 \right]$$

- How can λ be selected?

Shrinkage methods (regularization)

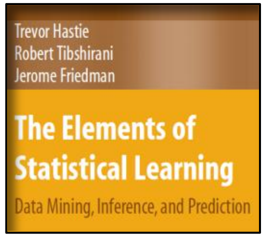


p 250

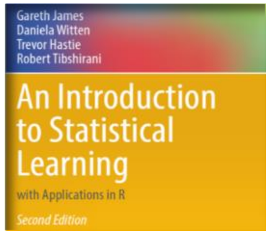
$$\hat{\boldsymbol{\beta}} = \arg\min_{\boldsymbol{\beta}} \left[\sum_{i=1}^{N} L(y_i, \hat{y}(\mathbf{x}_i, \boldsymbol{\beta})) + \lambda \sum_{j=1}^{p} \beta_j^2 \right]$$

- How can λ be selected? **Cross-validation** (or AIC, BIC, etc., but CV is preferred)
- Cross-validation is built into the MATLAB function “`lasso`”

Shrinkage methods (regularization)



p 68

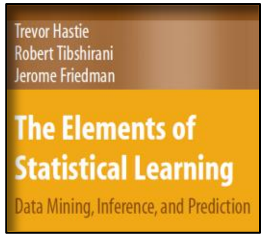


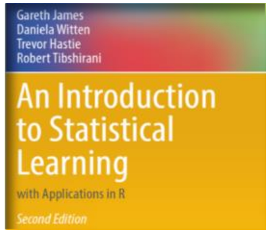
p 241

$$\hat{\boldsymbol{\beta}} = \arg\min_{\boldsymbol{\beta}} \left[\sum_{i=1}^{N} L(y_i, \hat{y}(\mathbf{x}_i, \boldsymbol{\beta})) + \lambda \sum_{j=1}^{p} \beta_j^2 \right]$$

- How can λ be selected? **Cross-validation** (or AIC, BIC, etc., but CV is preferred)
- Cross-validation is built into the MATLAB function "`lasso`"
- Can we use other penalty functions?

Shrinkage methods (regularization)





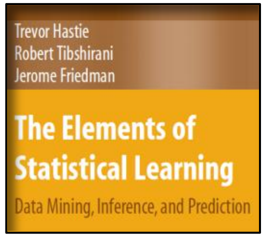
p 68 p 241

$$\widehat{\boldsymbol{\beta}} = \arg\min_{\boldsymbol{\beta}} \left[\sum_{i=1}^{N} L(y_i, \hat{y}(\mathbf{x}_i, \boldsymbol{\beta}) + \lambda \sum_{j=1}^{p} \beta_j^2 \right]$$

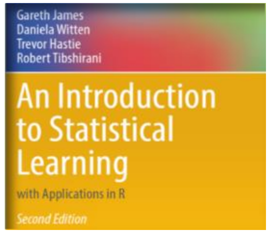
- How can λ be selected? **Cross-validation** (or AIC, BIC, etc., but CV is preferred)
- Cross-validation is built into the MATLAB function "`lasso`"
- Can we use other penalty functions? **Yes**, the L_1 penalty is the most popular

$$\widehat{\boldsymbol{\beta}} = \arg\min_{\boldsymbol{\beta}} \left[\sum_{i=1}^{N} L(y_i, \hat{y}(\mathbf{x}_i, \boldsymbol{\beta}) + \lambda \sum_{j=1}^{p} |\beta_j| \right]$$

Shrinkage methods (regularization)



p 68

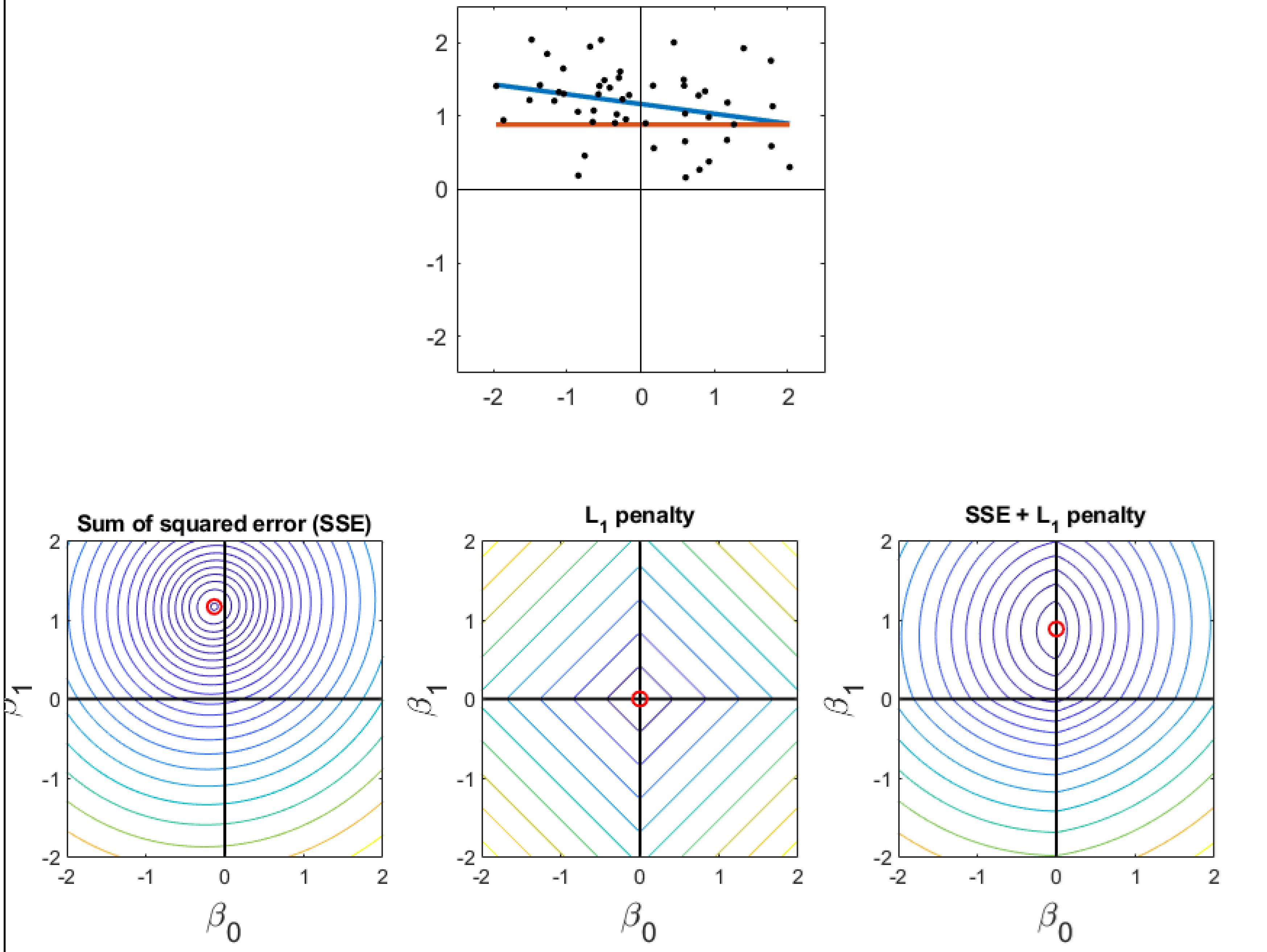


p 241

$$\widehat{\boldsymbol{\beta}} = \arg\min_{\boldsymbol{\beta}} \left[\sum_{i=1}^{N} L(y_i, \hat{y}(\mathbf{x}_i, \boldsymbol{\beta})) + \lambda \sum_{j=1}^{p} |\beta_j| \right]$$

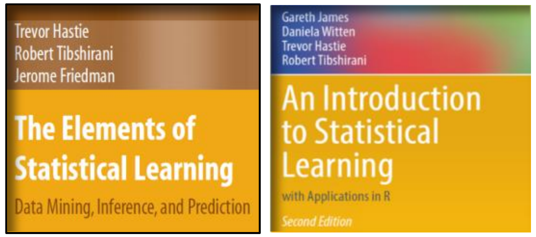
- The L_1 penalty has the ability to set coefficients to exactly zero, i.e. $\beta_j = 0$

- Thus, L_1 penalty, also called "the lasso" inherently performs feature selection on)

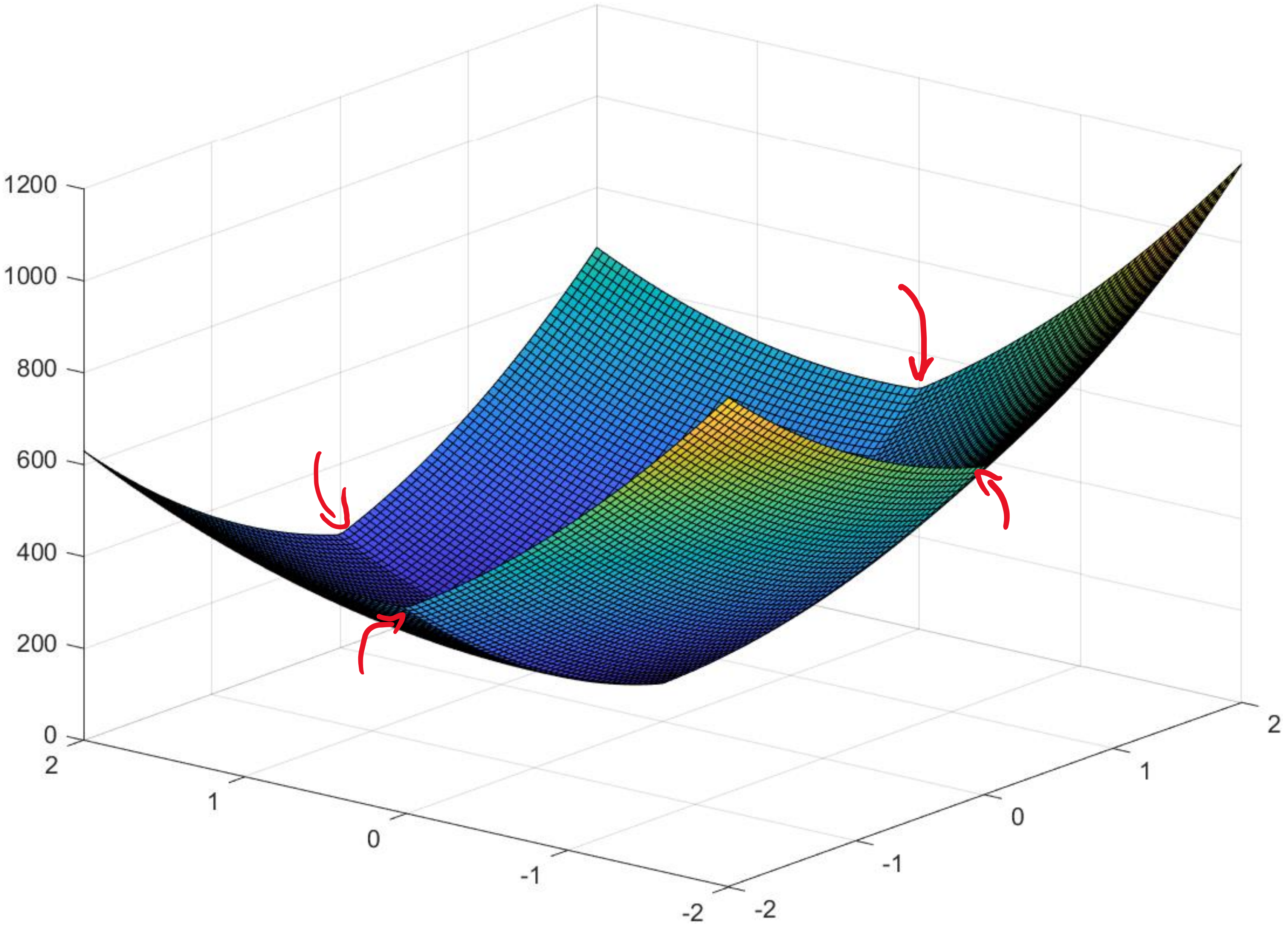


The L_1 penalty has sharp vertices whenever $\beta_j = 0$

Optimal solution lies along these vertices, i.e. where some $\beta_j = 0$

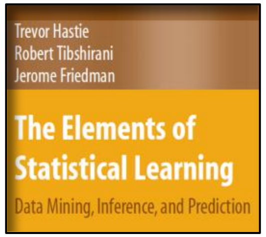


p 68 p 241

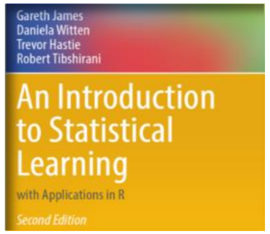


The L_1 penalty has sharp vertices whenever $\beta_j = 0$

Optimal solution lies along these vertices, i.e. where some $\beta_j = 0$

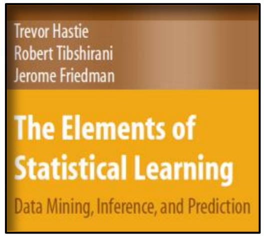


p 68



p 241

Shrinkage methods (regularization)



p 72

- "Elastic net" regularization is a hybrid between ridge regression and the lasso

$$\widehat{\boldsymbol{\beta}} = \arg\min_{\boldsymbol{\beta}} \left[\sum_{i=1}^{N} L(y_i, \hat{y}(\mathbf{x}_i, \boldsymbol{\beta}) + \lambda \left((1-\alpha) \sum_{j=1}^{p} \beta_j^2 + 2\alpha \sum_{j=1}^{p} |\beta_j| \right) \right]$$

In MATLAB and Python: Example 9

- Go to the next cell, and set the value of "p" to your choice

```
%% Example 9:  Regularize the G-RBF model …with cross-validation
```

- The MATLAB/Python code is already setup and ready to run, no need to add additional code.
- Inputs to "`lasso`" function:

  α in elastic net: `alpha = 1;`
  K in K-fold cross-validation: `K = 10;`
  Vector of λ values to consider: `lambda_vec = logspace(-3,0);`

```
[beta, FitInfo] = lasso(X_train, Data.y, 'Alpha', alpha, …
                                          'CV', K, …
                                      'Lambda',
lambda_vec);
```

- Look closely at the "`FitInfo`" object

In MATLAB and Python: Example 9

- Go to the next cell, and set the value of "p" to your choice

```
%% Example 9:  Regularize the G-RBF model …with cross-validation
```

- The MATLAB/Python code is already setup and ready to run, no need to add additional code.
- Inputs to "`linear_model.ElasticNetCV`" function:

  α in elastic net: `l1_ratio = 1;`

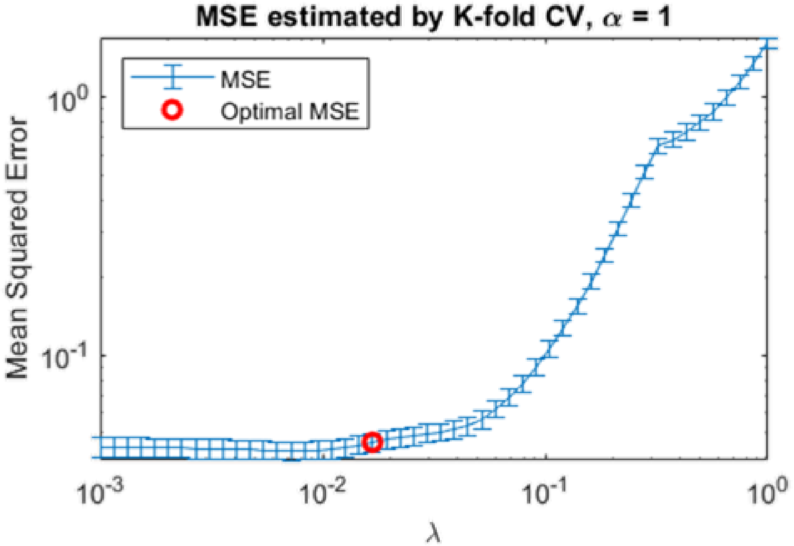
  K in K-fold cross-validation: `k = 10;`

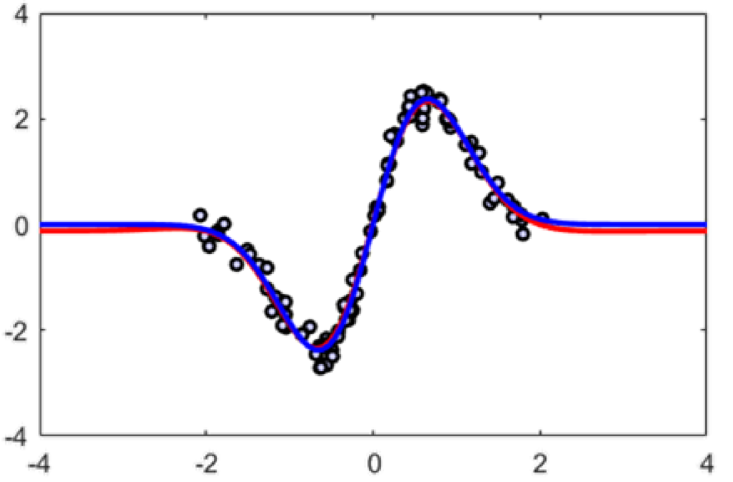
  Vector of λ values to consider: `alphas = logspace(-3,0);`

```
mdl = linear_model.ElasticNetCV(  l1_ratio = l1_ratio,
                                  alphas = alpha_vec,
                                  cv = k)
mdl.fit(X_train, Data.y)
```

```
%% Example 9:  Regularize the G-RBF model …
⋮
alpha = 1e-0;
K = 10;
lambda_vec = logspace(-3,0);
[beta, FitInfo] = lasso(X_train, Data.y, 'Alpha', alpha, …
                        'CV', K, 'Lambda', lambda_vec);

% Find the beta values that correspond to the "smallest"
model
beta_best = [FitInfo.Intercept(FitInfo.Index1SE);…
             beta(:,FitInfo.Index1SE)]
MSE_best = FitInfo.MSE(FitInfo.Index1SE);

⋮
% Evaluate the best fit model
subplot(2,1,2)
X_train = CreateGaussDesignMatrix(Fit.t, c);
Fit.RBF_lasso = beta_best(1) + X_train*beta_best(2:end);
plot(Fit.t, Fit.RBF_lasso, 'r', Fit.t, Fit.f, ‘b’,…
     'LineWidth', 2);
```

```
%% Example 9:  Regularize the G-RBF model …
:
alpha = 1e-0;
K = 10;
lambda_vec = logspace(-3,0);
[beta, FitInfo] = lasso(X_train, Data.y, 'Alpha', alpha, …
                        'CV', K, 'Lambda', lambda_vec);

% Find the beta values that correspond to the "smallest"
model
beta_best = [FitInfo.Intercept(FitInfo.Index1SE);…
             beta(:,FitInfo.Index1SE)]
MSE_best = FitInfo.MSE(FitInfo.Index1SE);

:
% Evaluate the best fit model
subplot(2,1,2)
X_train = CreateGaussDesignMatrix(Fit.t, c);
Fit.RBF_lasso = beta_best(1) + X_train*beta_best(2:end);
plot(Fit.t, Fit.RBF_lasso, 'r', Fit.t, Fit.f, 'b',…
     'LineWidth', 2);
```





Recap

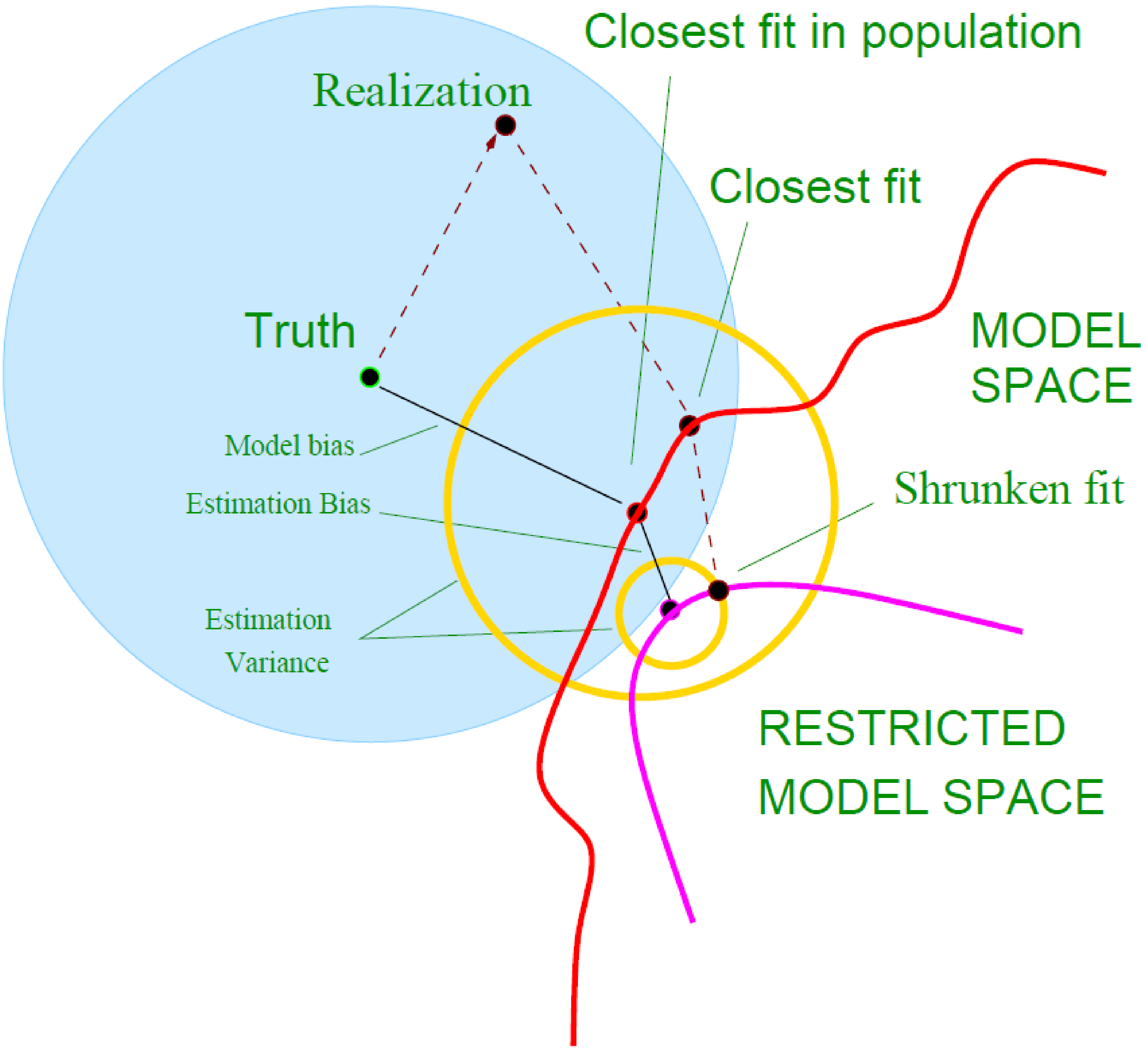
- Goal of machine learning is to train models that perform well on new data

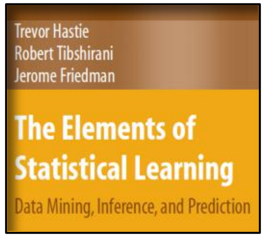
$$\widehat{\boldsymbol{\beta}} = \arg\min_{\boldsymbol{\beta}} E\big(\mathbf{y}_{new}; \hat{y}(\mathbf{X}_{new}; \boldsymbol{\beta})\big)$$

- Model performance depends on bias-variance trade-off
- By introducing a small amount of bias, a large reduction in variance can be obtained, with a net increase in model performance
- Regularization:

  provides the means of gradually introducing bias through λ,
  to optimize model performance as assessed using cross-validation

Recap


Closest fit in population
Realization
Closest fit
Truth
MODEL
SPACE
Model bias
Shrunken fit
Estimation Bias
Estimation
Variance
RESTRICTED
MODEL SPACE


Trevor Hastie
Robert Tibshirani
Jerome Friedman
The Elements of
Statistical Learning
Data Mining, Inference, and Prediction

p 225

Recap

Shrinkage methods (ridge regression, lasso) are but one example of many regularization approaches

- Feature selection (beginning of this session)
- Dimensionality reduction (next session)
- Random drop-out (neural networks)
- Weight-sharing (convolutional neural networks)
- Early stopping (any approach requiring iterative optimization)
- Kernel methods (Gaussian process regression)
- ...

Much of the success of machine learning can be associated with the ability to precisely manipulate the bias-variance trade-off to improve performance on new data