Improving model performance:

Model Selection, Parameter Tuning and Feature Engineering

B. Michel

Ecole Centrale de Nantes

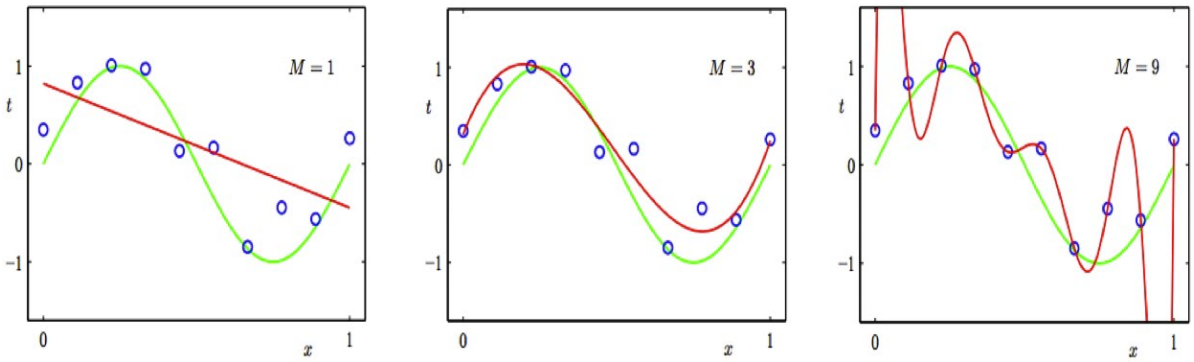
Statistical Learning

Outline

Overfitting

- Statistics is not just interpolating between observation points !
- Overfitting occurs when the learning rule too closely corresponds to a particular set of data, and may therefore fail to generalize on future observations.

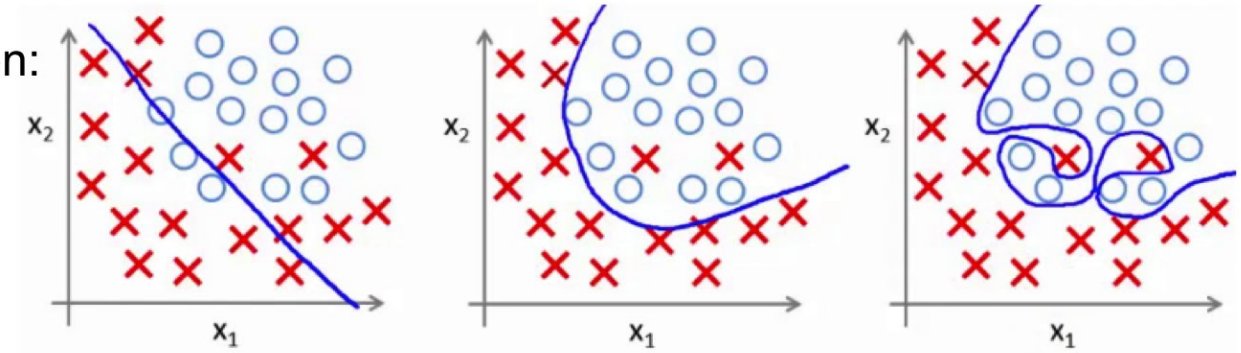
Regression:



predictor too inflexible:
cannot capture pattern

predictor too flexible:
fits noise in the data

Classification:



Overfitting

- Statistics is not just interpolating between observation points !
- Overfitting occurs when the learning rule too closely corresponds to a particular set of data, and may therefore fail to generalize on future observations.
- An overfitted model is a statistical model that contains more parameters than can be learned (estimated) by the data.
- Examples :
  - In regression : e.g. a linear model with too many variables (features).
  - In classification : e.g. a k-NN with k too small.
  - In density estimation : an histogram with too many bins.
  - Kernel methods : bandwidth too small
  - ...
- High dimensional statistics : n is of the order (or smaller than) the number of features p.

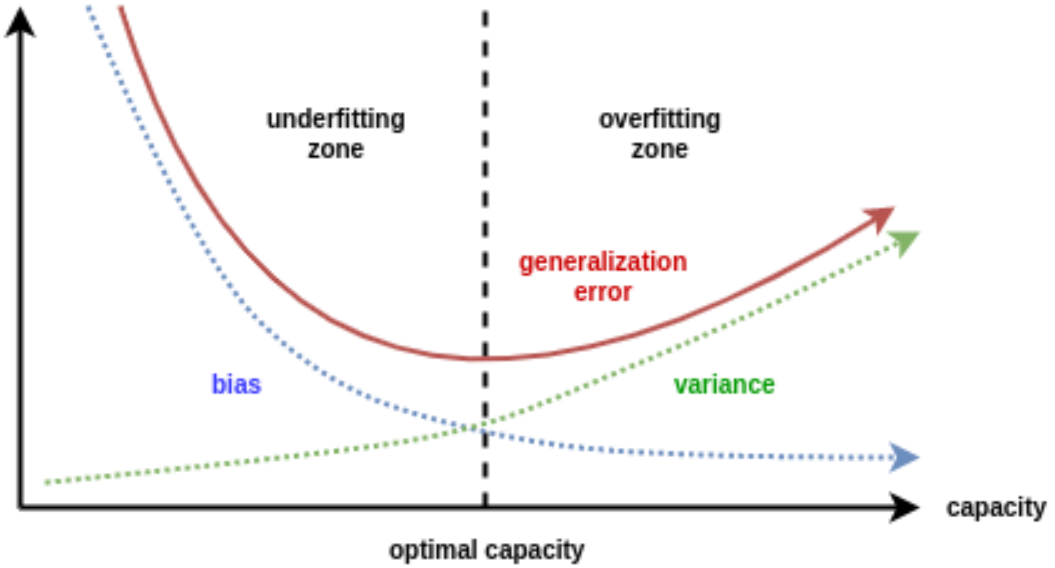
Bias Variance Tradeoff

- loss ℓ : quadratic, logistic, hinge loss ....
- Risk or Generalization error of a learning rule $\hat{f}$: $\mathbb{E}\ell(Y, \hat{f}(X))$
- Typical decomposition of the generalization error

Generalization error = (or $\leq$) Approximation error + Estimation error

See for instance this page for more details for regression with the quadratic loss.

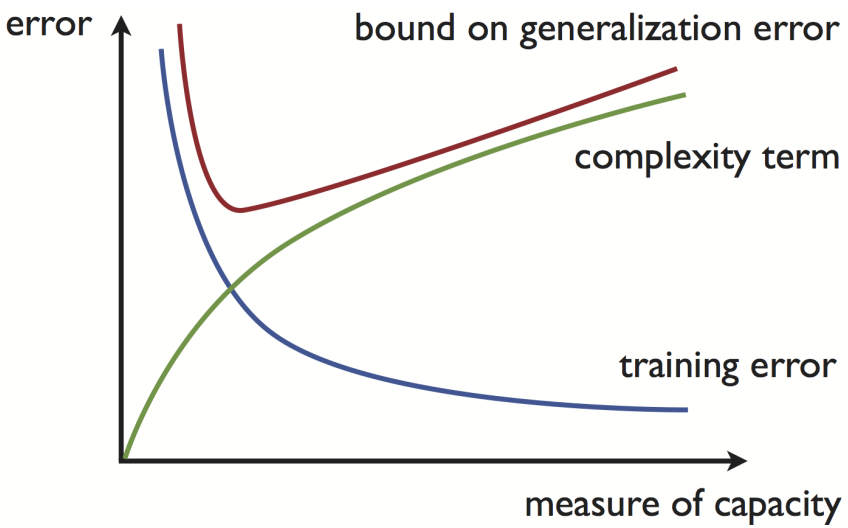
- Estimation error : random term $\approx$ variance term in statistics.
- Approximation error : deterministic term $\approx$ bias term in statistics.



[Daniel Saunders]

Training error versus test error

- How can we **estimate** the generalization error from the data ?
- The **learning error** is the average error that results from estimating the performance of a learning rule using the sample **already used** to fit the learning rule.
- The **test error** is the average error that results from estimating the performance of a learning rule using a **different** sample.
- The **learning error** can dramatically **underestimate** the generalization error :



Question: Why does the learning error may fail to estimate the generalization error ?

Outline

Splitting

- Sample $(X_i, Y_i)_{i=\dots n}$, loss function ℓ.
- Split the data at random :
  - I_{train} : observations in the train set ;
  - I_{test} : observations in the test set.
- Learn a learning rule $\hat{f}$ on I_{train}.
  E.g Empirical Risk Minimization (ERM) on S with loss ℓ:

$$\hat{f} = \underset{f \in S}{\operatorname{argmin}} \frac{1}{|I_{\text{train}}|} \sum_{i \in I_{\text{train}}} \ell(Y_i, f(X_i))$$

- Compute the test error on I_{test}:

$$\text{Err}_{\text{test}} = \frac{1}{|I_{\text{test}}|} \sum_{i \in I_{\text{test}}} \ell(Y_i, \hat{f}(X_i))$$

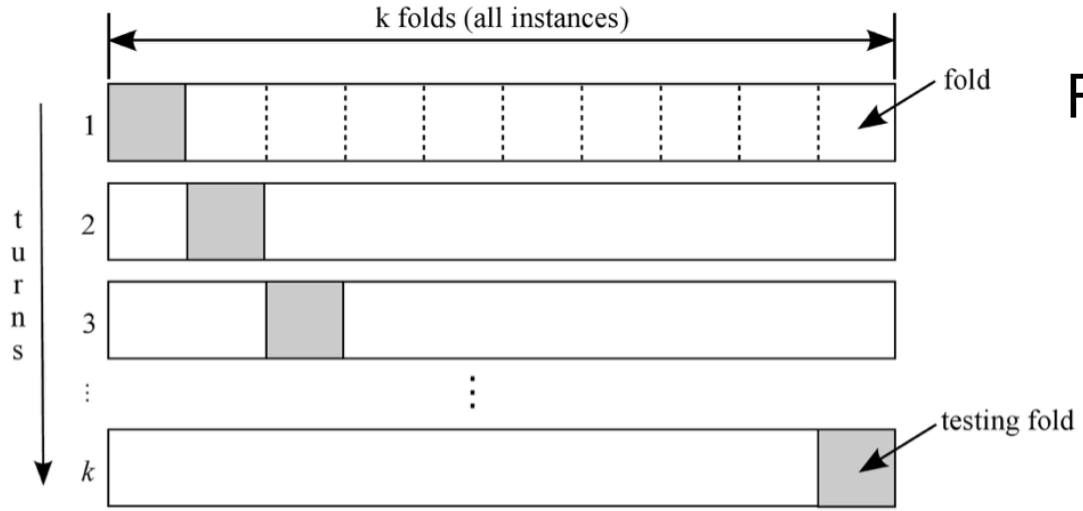
- Split at random N times and average the N test errors.

Splitting

- For each splitting we compute a learning rule : high computational cost.
- The train sample is smaller than the complete sample : it overestimates the generalization error.
- Alternative : Cross validation methods.

Cross validation : k-fold cross validation

- k-fold validation: pick a random partition $I_1, \ldots, I_v, \ldots I_k$ of $\{1, \ldots, n\}$ where $|I_v| \sim n/k$.



[Gaiffas]

For each fold v :

- $I_{\text{test},v} = I_v$,
- $I_{\text{train},v} = \{1, \ldots, n\} \setminus I_v$,
- $\hat{f}_v$: learning rule fitted (trained) on $I_{\text{train},v}$.

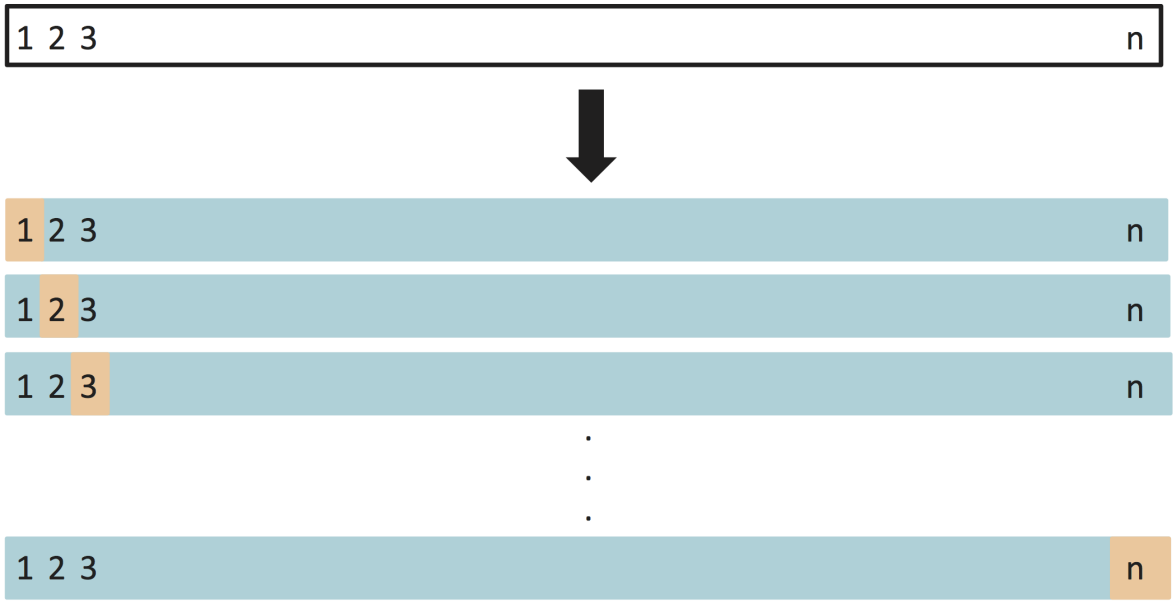
- Compute

$$\text{Err} = \frac{1}{k} \sum_{v=1 \ldots k} \frac{1}{|I_v|} \sum_{i \in I_v} \ell(Y_i, \hat{f}_v(X_i)).$$

- Only k learning rules are fitted with this approach.

Cross validation : Leave one out

- Leave one out (Loo) : $k = n$



[From : The Elements of Stat Learning]

- $I_{\text{test},v} = \{v\}$,
- $I_{\text{train},v} = \{1, \ldots, n\} \setminus \{v\}$,

Cross validation

- Cross validation aims at estimating the generalization error :
  - Cross validation methods are estimation procedures, they have a bias and a variance.
  - In practice : take $k \approx 5 - 10$.
  - Cross validation with sklearn :

- Cross validation can be used for **hyperparameter optimization** :
  - tuning a parameter, e.g. for regularization.
  - for selecting models in a given collection.

Hyperparameter optimization

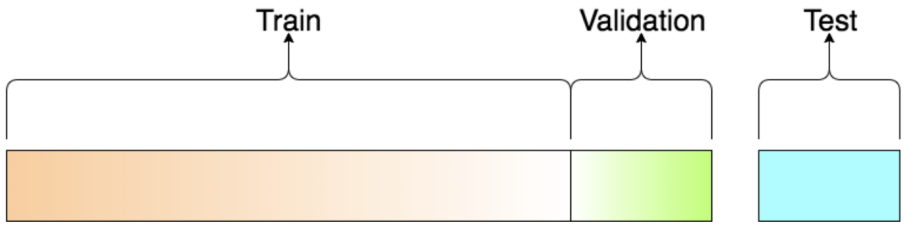
- **Hyperparameter optimization** or **tuning** is choosing a set of optimal hyperparameters for a ML algorithm. Ex. number of classes for nearest neighbor classification.

- **Cross-validation** is the standard approach to estimate the generalization performance of the algorithm for a given value of the hyperparameter.

- But how to **explore** the hyperparameter space ?
  - Gradient descent is generally not possible for hyperparam. optim.
  - **Grid search**: exhaustive search (brute-force) through a manually specified subset of the hyperparameter space, see sckit-learn. Simple but computationally expensive thus awfully energivorous process.
  - **Random search** : random search in the hyperparameter space, see sckit-learn.
  - **Bayesian optimization** see see this paper and this library

Train, Validation, Test and complete datasets

During Hyperparameter optimization, cross-validation repeatedly splits a dataset into several training and a validation datasets.



[Toward Data Sciences, Tarang Shah's blog.]

- A **training** data set is a dataset used during the learning process and is used to fit the parameters:
- A **validation** data set is a dataset used to tune the hyperparameters of predictor.
- A **test** set is an additional dataset used only to assess the performance (i.e. generalization) of a fully specified predictor.

In all cases, **after tuning the parameters and selecting a model, you should fit the chosen model on the complete dataset (not only on the train dataset) to define the final predictor.**

Outline

Linear regression in high dimension

- Regression, design matrix X of size $n \times p$, target Y.
- Linear predictors of Y : $f_\beta(x) = x'\beta$.
- Quadratic loss : $\hat{\beta} = (X'X)^{-1}X'Y$.
- High dim: $n \sim p$ or $n \lesssim p$: $X'X$ is not inversible.
- Ridge regularization :

$$\hat{\beta}_\lambda^{\text{ridge}} = \underset{\beta \in \mathbb{R}^p}{\text{argmin}} \, \|Y - X\beta\|^2 + \lambda \sum_{j=1}^{p} \beta_j^2.$$

  and the solution is

$$\hat{\beta}_\lambda^{\text{ridge}} = (X'X + \lambda \text{Id}_p)^{-1}X'Y.$$

- It can be proved that ridge penalization leads to a better generalization error.

Penalized Empirical Risk Minimization

- Penalized Empirical Risk Minimization is not restricted to linear models, it consists in finding a solution to

$$\hat{f}_{S,\lambda} := \underset{f \in S}{\operatorname{argmin}} \sum_{i=1\ldots n} \ell(Y, f(X_i)) + \lambda \operatorname{pen}(f)$$

  where :
  - S is a class of predictors
  - pen is a (positive) penalization function, that controls the complexity of the selected model.
  - λ is a tuning or smoothing parameter, that balances goodness-of-fit and penalization.

- λ can be chosen by cross validation.

- Other penalties : data driven penalty, radamacher, boostrap ...

Penalized linear predictors

$$\underset{f \in S}{\operatorname{argmin}} \sum_{i=1\ldots n} \ell(Y_i, f(X_i)) + \lambda \operatorname{pen}(f_\beta) \tag{1}$$

with

$$S: \quad f_\beta^{reg}(x) = \langle x, \beta \rangle \quad \text{or} \quad f_\beta^{cla}(x) = \operatorname{sign}(\langle x, \beta \rangle)$$

- When $\beta_j = 0$, then feature j has no impact on the prediction.
- To control overfitting but also to improve interpretation we would like β to be sparse : many 0's in β.

Penalized linear predictors

$$\underset{f \in S}{\operatorname{argmin}} \sum_{i=1\ldots n} \ell(Y_i, f(X_i)) + \lambda \operatorname{pen}(f_\beta) \tag{1}$$

with

$$S: \quad f_\beta^{reg}(x) = \langle x, \beta \rangle \quad \text{or} \quad f_\beta^{cla}(x) = \operatorname{sign}(\langle x, \beta \rangle)$$

- First option: the ℓ_0 penalty $\|\beta\|_0 = \left|\{j \in \{1, \ldots, p\} \ \beta_j \neq 0\}\right|$
  - corresponds to AIC, BIC and Mallows C_p and more recent ℓ_0 strategies.
  - strong mathematical guarantees.
  - ... but NP hard problem because we have to consider and find the solution for all the possible supports for β.

Penalized linear predictors

$$\underset{f \in S}{\operatorname{argmin}} \sum_{i=1\ldots n} \ell(Y_i, f(X_i)) + \lambda \operatorname{pen}(f_\beta) \tag{1}$$

with

$$S: \quad f_\beta^{reg}(x) = \langle x, \beta \rangle \quad \text{or} \quad f_\beta^{cla}(x) = \operatorname{sign}(\langle x, \beta \rangle)$$

- The ridge (ℓ_2) penalty $\|\beta\|_2 = \sqrt{\sum_{j=1}^{p} \beta_j^2}$
  - Lagrange duality:

    (1) $\Leftrightarrow$ minimize $\sum_{i=1\ldots n} \ell(Y, f(X_i))$ under $\|\beta\|_2 \leq C_\lambda$.

Question: Draw the ℓ_2 unit ball in $\mathbb{R}^2$. How does it intersect with the level sets of the empirical risk ?

  - No strong sparsity induced !

Penalized linear predictors

$$\underset{f \in S}{\operatorname{argmin}} \sum_{i=1\ldots n} \ell(Y_i, f(X_i)) + \lambda \operatorname{pen}(f_\beta) \tag{1}$$

with

$$S: \quad f_\beta^{reg}(x) = \langle x, \beta \rangle \quad \text{or} \quad f_\beta^{cla}(x) = \operatorname{sign}(\langle x, \beta \rangle)$$

- The Lasso (ℓ_1) penalty $\|\beta\|_1 = \sum_{j=1}^{p} |\beta_j|$
  - ℓ_1 appears as a convex proxy of the ℓ_0.
  - Lagrange duality :

    (1) $\Leftrightarrow$ minimize $\sum_{i=1\ldots n} \ell(Y, f(X_i))$ under $\|\beta\|_1 \leq C_\lambda$.

Question: Draw the ℓ_1 unit ball in $\mathbb{R}^2$. How does it intersect with the level sets of the empirical risk ?

  - Sparsity induced. Strong mathematical guarantees.
  - E.g standard Lasso (regression with quadratic loss), logistic Lasso (classification with logistic loss), SVM lasso (hinge loss) ...

Penalized linear predictors

$$\underset{f \in S}{\operatorname{argmin}} \sum_{i=1\ldots n} \ell(Y_i, f(X_i)) + \lambda \operatorname{pen}(f_\beta) \tag{1}$$

with

$$S: \quad f_\beta^{reg}(x) = \langle x, \beta \rangle \quad \text{or} \quad f_\beta^{cla}(x) = \operatorname{sign}(\langle x, \beta \rangle)$$

- Other options:
  - elastic-net : sum of lasso and ridge,
  - group lasso : select variables by blocks,
  - Fused lasso : penalizes the ℓ_1 norm of both the coefficients and their successive differences,
  - mixed strategies.

Penalty shapes

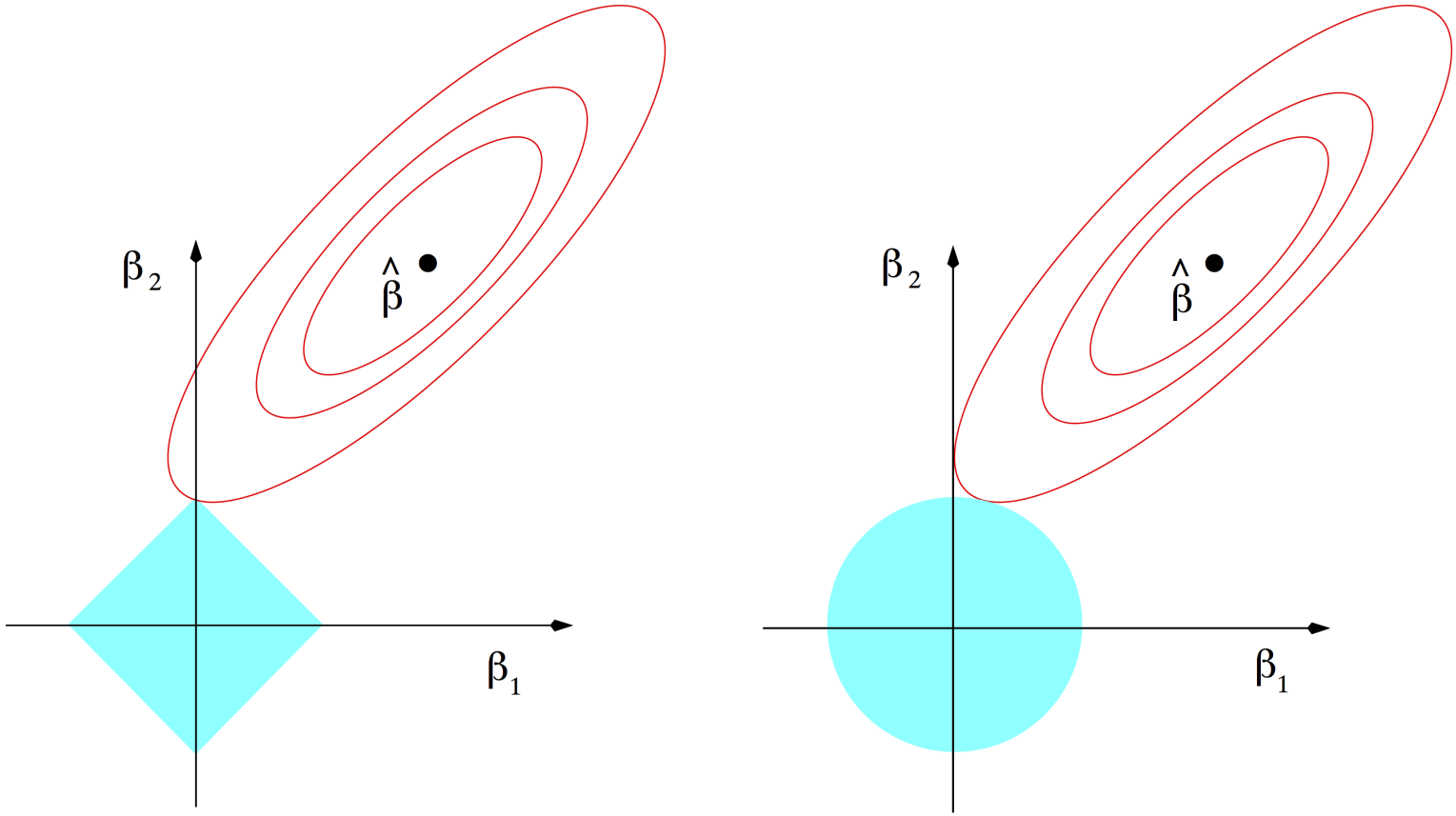
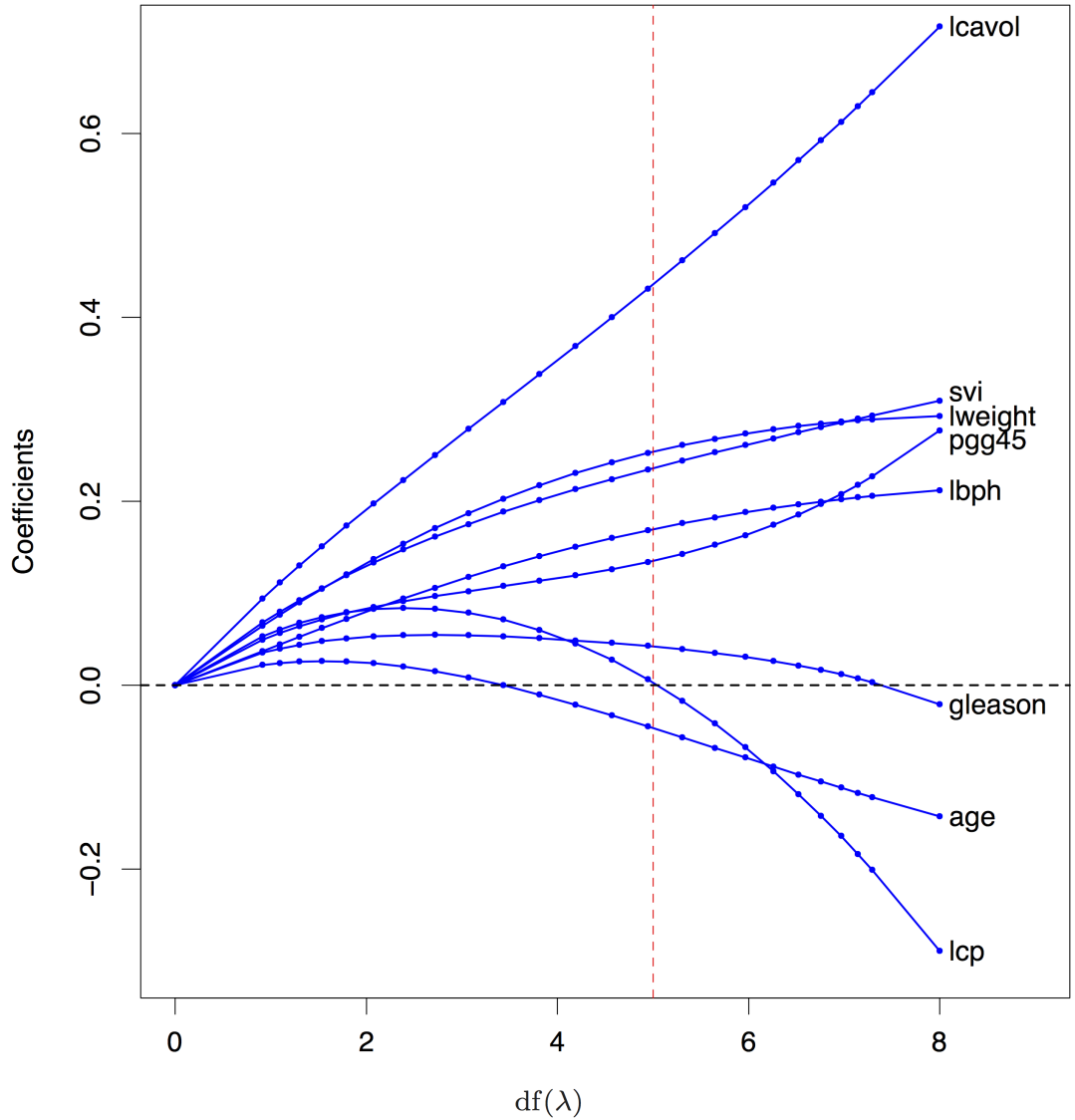


FIGURE 3.11. *Estimation picture for the lasso (left) and ridge regression (right). Shown are contours of the error and constraint functions. The solid blue areas are the constraint regions* $|\beta_1| + |\beta_2| \leq t$ *and* $\beta_1^2 + \beta_2^2 \leq t^2$*, respectively, while the red ellipses are the contours of the least squares error function.*

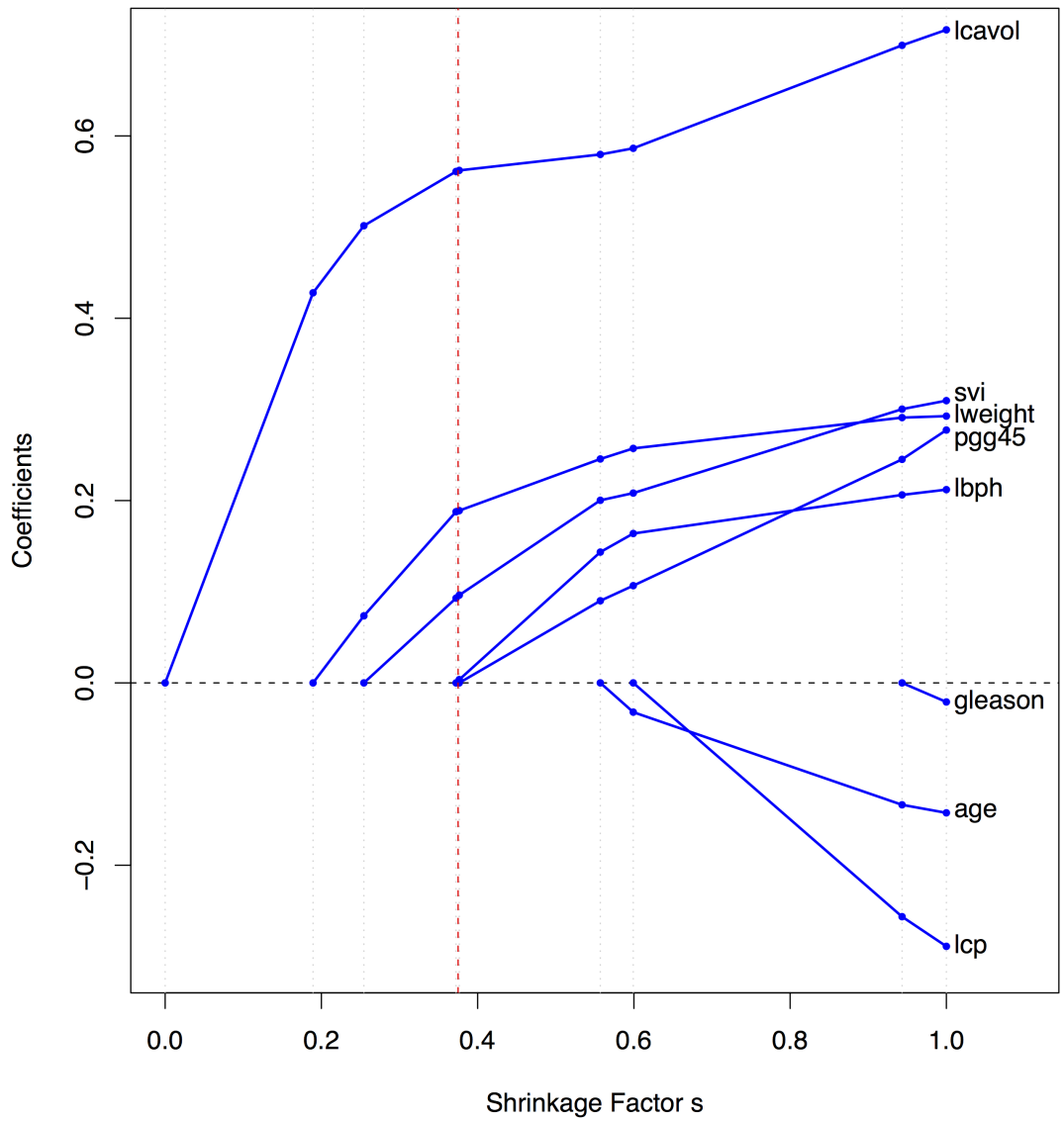
[From The Elements of Stat Learning]

Regularization path: Ridge



[From The Elements of Stat Learning]

Regularization path: Lasso



[From The Elements of Stat Learning]

Outline

Normalization

- Contrary to the standard least square estimator, ridge and lasso estimators are not scale invariant, even for the quadratic loss.
- We need to normalize the variables :
  - center, include an intercept, standardize
  - min-max scaling
  - binarize the variables
- More generally, these preliminary steps are recommended for most machine learning procedures.

Feature extraction and feature generation

- Create new variables to improve the prediction performances and use model selection / regularization methods to control the statistical complexity.
- Recipe / list of basic ideas :
  - Transform variables to obtain a better (simpler) relationship with the output : log, exponential, quadratic, sin, cos ..transformations
  - Introduce interaction variables
  - If there exists a model for Y (e.g. physical or biological model), including the corresponding variables will help the machine learning approach.
  - Times series : min / max / min-max / derivatives / spectral pattern
  - Decomposition and projection : PCA, splines, Fourrier, wavelets ...
  - Images : specific pattern
- The python library featuretools generates many classical features.
- Take home message: Feature engineering is generally a critical step for machine learning !