

# Models Selection and Feature Selection

Evgeny Burnaev

Skoltech, Moscow, Russia



Skolkovo Institute of Science and Technology

- 1 Overfitting. Bias-variance decomposition
- 2 Model Quality Criteria
- 3 Feature Selection
- 4 Regularization
- 5 Linear Regression Model Selection
- 6 Appendix. Materials for Additional Study: Sensitivity Analysis

## 1 Overfitting. Bias-variance decomposition

## 2 Model Quality Criteria

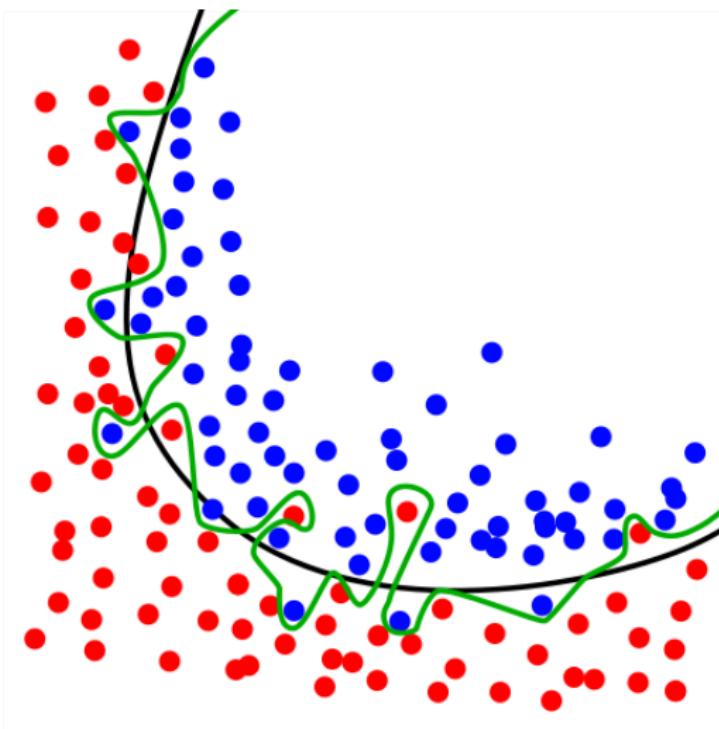
## 3 Feature Selection

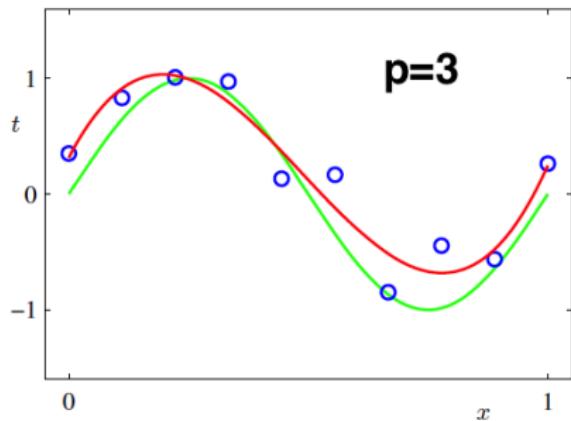
## 4 Regularization

## 5 Linear Regression Model Selection

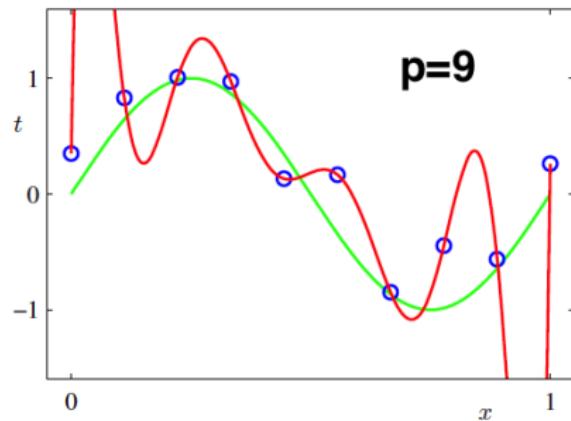
## 6 Appendix. Materials for Additional Study: Sensitivity Analysis

# Overfitting: KNN

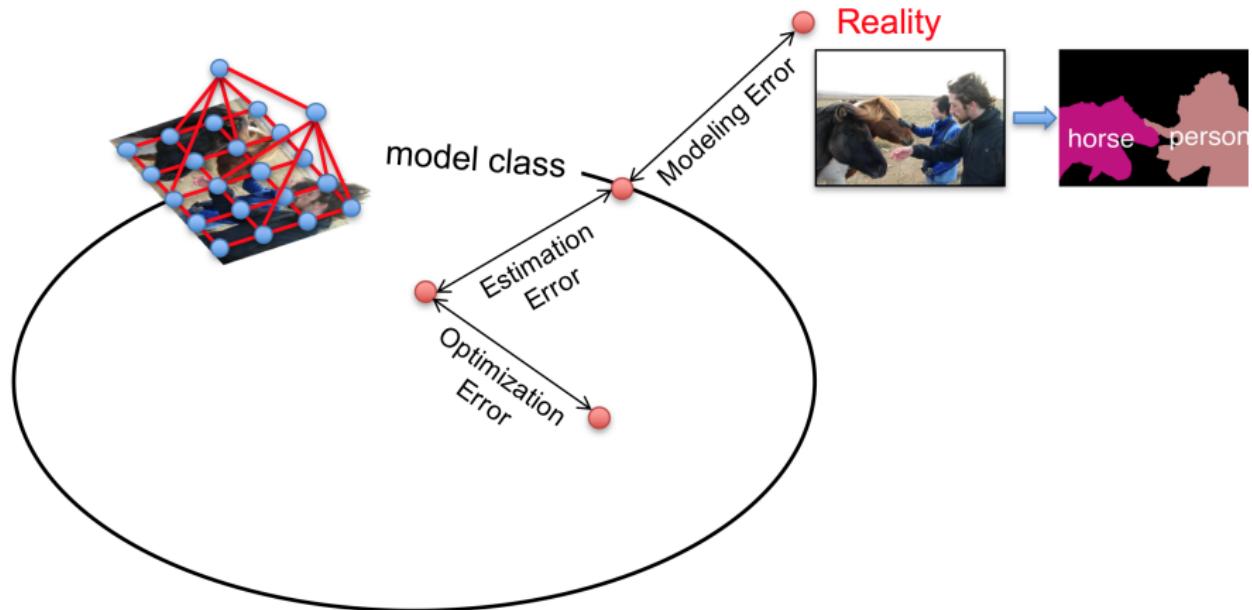




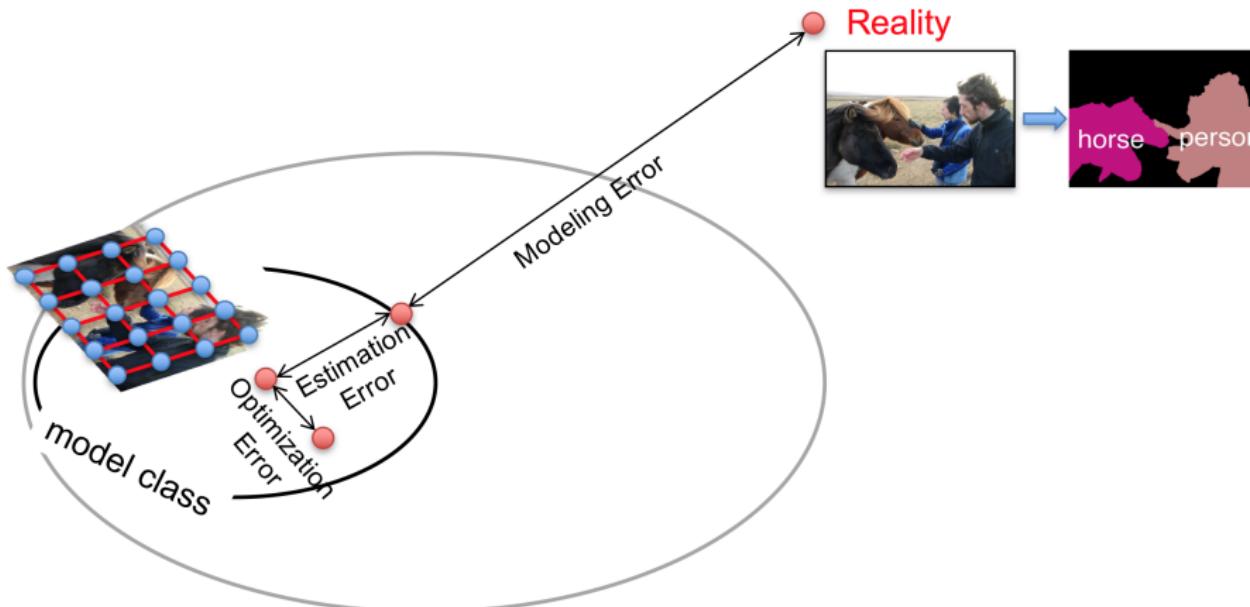
**$p=3$**



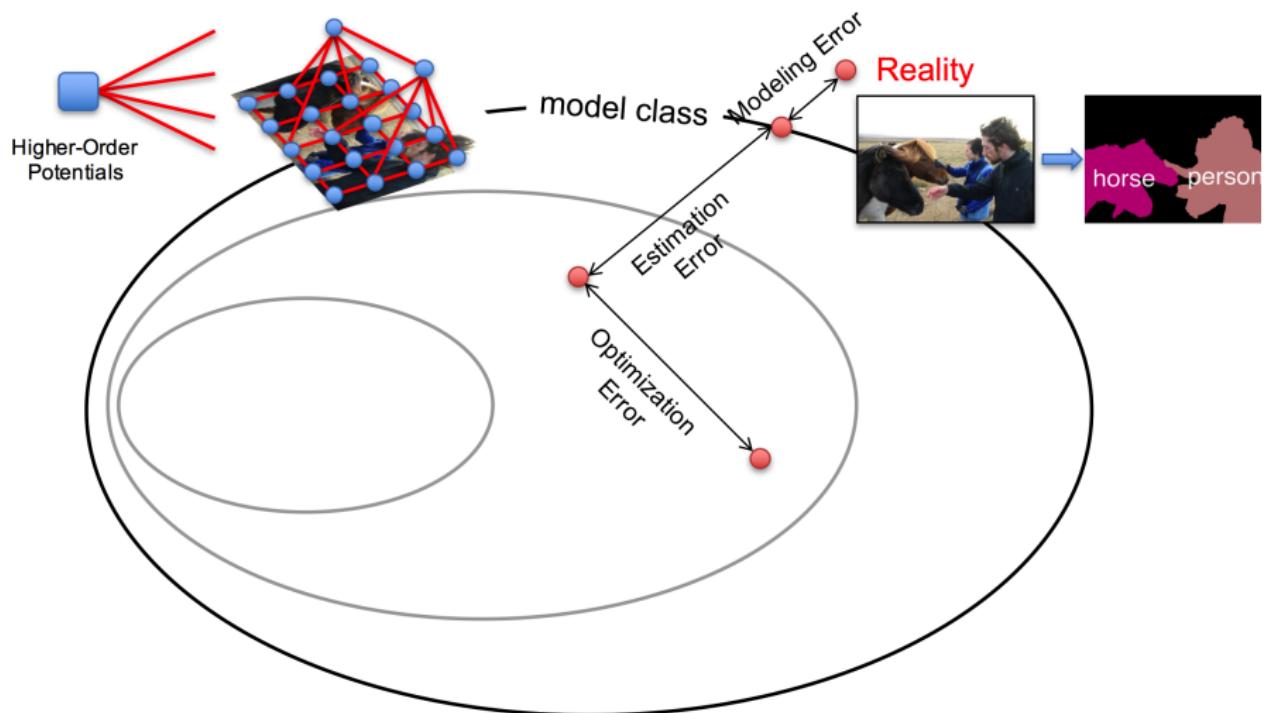
**$p=9$**



# Error Decomposition



# Error Decomposition



- Approximation/Modeling Error
  - You approximated reality with a model
- Estimation Error
  - You tried to learn model with finite data
- Optimization Error
  - You were lazy and couldn't/didn't optimize to completion
- Bayes Error
  - Reality just sucks (i.e. there is a lower bound on error for all models, usually non-zero)

- Approximation/Modeling Error
  - You approximated reality with a model
- Estimation Error
  - You tried to learn model with finite data
- Optimization Error
  - You were lazy and couldn't/didn't optimize to completion
- Bayes Error
  - Reality just sucks (i.e. there is a lower bound on error for all models, usually non-zero)

- Approximation/Modeling Error
  - You approximated reality with a model
- Estimation Error
  - You tried to learn model with finite data
- Optimization Error
  - You were lazy and couldn't/didn't optimize to completion
- Bayes Error
  - Reality just sucks (i.e. there is a lower bound on error for all models, usually non-zero)

- Approximation/Modeling Error
  - You approximated reality with a model
- Estimation Error
  - You tried to learn model with finite data
- Optimization Error
  - You were lazy and couldn't/didn't optimize to completion
- Bayes Error
  - Reality just sucks (i.e. there is a lower bound on error for all models, usually non-zero)

- Using regression as model, assume that

$$y = f(\mathbf{x}) + \varepsilon,$$

where  $\mathbb{E}\varepsilon = 0$ ,  $\text{Var}(\varepsilon) = \sigma_\varepsilon^2$

- Then at an input point  $\mathbf{x} = \mathbf{x}_0$

$$\begin{aligned} Err(\mathbf{x}_0) &= \mathbb{E} [(y - \hat{f}(\mathbf{x}_0))^2 | \mathbf{x} = \mathbf{x}_0] \\ &= \sigma_\varepsilon^2 + [\mathbb{E}\hat{f}(\mathbf{x}_0) - f(\mathbf{x}_0)]^2 + \mathbb{E} [\hat{f}(\mathbf{x}_0) - \mathbb{E}\hat{f}(\mathbf{x}_0)]^2 \\ &= \sigma_\varepsilon^2 + \text{Bias}^2(\hat{f}(\mathbf{x}_0)) + \text{V}(\hat{f}(\mathbf{x}_0)) \\ &= \text{Irreducible Error} + \text{Bias}^2 + \text{Variance} \end{aligned}$$

- Using regression as model, assume that

$$y = f(\mathbf{x}) + \varepsilon,$$

where  $\mathbb{E}\varepsilon = 0$ ,  $\text{Var}(\varepsilon) = \sigma_\varepsilon^2$

- Then at an input point  $\mathbf{x} = \mathbf{x}_0$

$$\begin{aligned} Err(\mathbf{x}_0) &= \mathbb{E} \left[ (y - \hat{f}(\mathbf{x}_0))^2 | \mathbf{x} = \mathbf{x}_0 \right] \\ &= \sigma_\varepsilon^2 + \left[ \mathbb{E} \hat{f}(\mathbf{x}_0) - f(\mathbf{x}_0) \right]^2 + \mathbb{E} \left[ \hat{f}(\mathbf{x}_0) - \mathbb{E} \hat{f}(\mathbf{x}_0) \right]^2 \\ &= \sigma_\varepsilon^2 + \text{Bias}^2 \left( \hat{f}(\mathbf{x}_0) \right) + \text{V} \left( \hat{f}(\mathbf{x}_0) \right) \\ &= \text{Irreducible Error} + \text{Bias}^2 + \text{Variance} \end{aligned}$$

- Using regression as model, assume that

$$y = f(\mathbf{x}) + \varepsilon,$$

where  $\mathbb{E}\varepsilon = 0$ ,  $\text{Var}(\varepsilon) = \sigma_\varepsilon^2$

- Then at an input point  $\mathbf{x} = \mathbf{x}_0$

$$\begin{aligned} Err(\mathbf{x}_0) &= \mathbb{E} \left[ (y - \hat{f}(\mathbf{x}_0))^2 | \mathbf{x} = \mathbf{x}_0 \right] \\ &= \sigma_\varepsilon^2 + \left[ \mathbb{E} \hat{f}(\mathbf{x}_0) - f(\mathbf{x}_0) \right]^2 + \mathbb{E} \left[ \hat{f}(\mathbf{x}_0) - \mathbb{E} \hat{f}(\mathbf{x}_0) \right]^2 \\ &= \sigma_\varepsilon^2 + \text{Bias}^2 \left( \hat{f}(\mathbf{x}_0) \right) + \text{V} \left( \hat{f}(\mathbf{x}_0) \right) \\ &= \text{Irreducible Error} + \text{Bias}^2 + \text{Variance} \end{aligned}$$

- Using regression as model, assume that

$$y = f(\mathbf{x}) + \varepsilon,$$

where  $\mathbb{E}\varepsilon = 0$ ,  $\text{Var}(\varepsilon) = \sigma_\varepsilon^2$

- Then at an input point  $\mathbf{x} = \mathbf{x}_0$

$$\begin{aligned} Err(\mathbf{x}_0) &= \mathbb{E} \left[ (y - \hat{f}(\mathbf{x}_0))^2 | \mathbf{x} = \mathbf{x}_0 \right] \\ &= \sigma_\varepsilon^2 + \left[ \mathbb{E} \hat{f}(\mathbf{x}_0) - f(\mathbf{x}_0) \right]^2 + \mathbb{E} \left[ \hat{f}(\mathbf{x}_0) - \mathbb{E} \hat{f}(\mathbf{x}_0) \right]^2 \\ &= \sigma_\varepsilon^2 + \text{Bias}^2 \left( \hat{f}(\mathbf{x}_0) \right) + \text{V} \left( \hat{f}(\mathbf{x}_0) \right) \\ &= \text{Irreducible Error} + \text{Bias}^2 + \text{Variance} \end{aligned}$$

- Using regression as model, assume that

$$y = f(\mathbf{x}) + \varepsilon,$$

where  $\mathbb{E}\varepsilon = 0$ ,  $\text{Var}(\varepsilon) = \sigma_\varepsilon^2$

- Then at an input point  $\mathbf{x} = \mathbf{x}_0$

$$\begin{aligned} Err(\mathbf{x}_0) &= \mathbb{E} \left[ (y - \hat{f}(\mathbf{x}_0))^2 | \mathbf{x} = \mathbf{x}_0 \right] \\ &= \sigma_\varepsilon^2 + \left[ \mathbb{E} \hat{f}(\mathbf{x}_0) - f(\mathbf{x}_0) \right]^2 + \mathbb{E} \left[ \hat{f}(\mathbf{x}_0) - \mathbb{E} \hat{f}(\mathbf{x}_0) \right]^2 \\ &= \sigma_\varepsilon^2 + \text{Bias}^2 \left( \hat{f}(\mathbf{x}_0) \right) + \text{V} \left( \hat{f}(\mathbf{x}_0) \right) \\ &= \text{Irreducible Error} + \text{Bias}^2 + \text{Variance} \end{aligned}$$

- Using regression as model, assume that

$$y = f(\mathbf{x}) + \varepsilon,$$

where  $\mathbb{E}\varepsilon = 0$ ,  $\text{Var}(\varepsilon) = \sigma_\varepsilon^2$

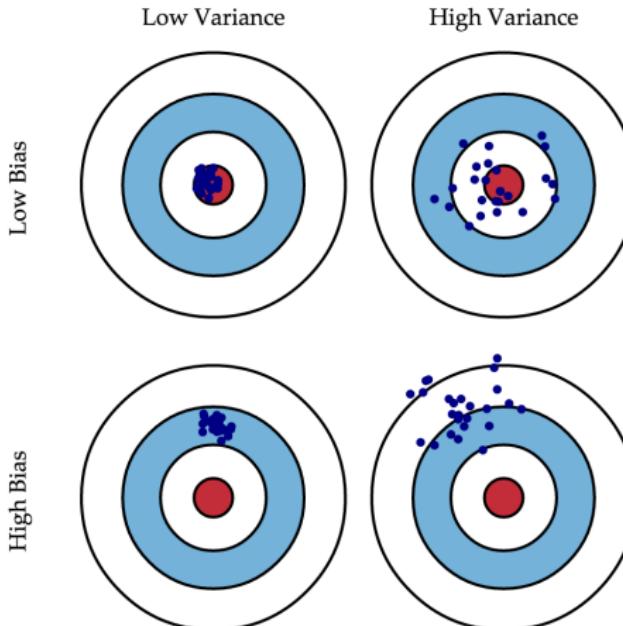
- Then at an input point  $\mathbf{x} = \mathbf{x}_0$

$$\begin{aligned} Err(\mathbf{x}_0) &= \mathbb{E} \left[ (y - \hat{f}(\mathbf{x}_0))^2 | \mathbf{x} = \mathbf{x}_0 \right] \\ &= \sigma_\varepsilon^2 + \left[ \mathbb{E} \hat{f}(\mathbf{x}_0) - f(\mathbf{x}_0) \right]^2 + \mathbb{E} \left[ \hat{f}(\mathbf{x}_0) - \mathbb{E} \hat{f}(\mathbf{x}_0) \right]^2 \\ &= \sigma_\varepsilon^2 + \text{Bias}^2 \left( \hat{f}(\mathbf{x}_0) \right) + \text{V} \left( \hat{f}(\mathbf{x}_0) \right) \\ &= \text{Irreducible Error} + \text{Bias}^2 + \text{Variance} \end{aligned}$$

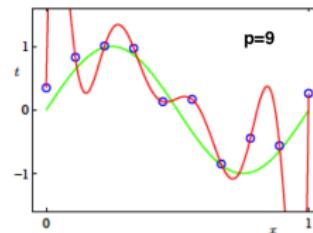
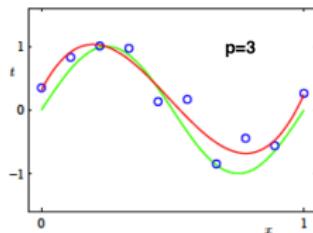
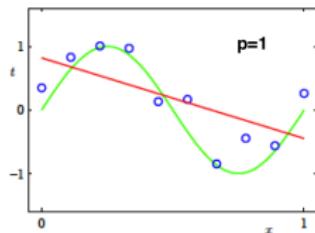
- **Bias:** difference between what you expect to learn and the truth
  - Measures how well you expect to represent true solution
  - Decreases with more complex model
- **Variance:** difference between what you expect to learn and what you learn from a from a particular dataset
  - Measures how sensitive learner is to specific dataset
  - Increases with more complex model

- **Bias:** difference between what you expect to learn and the truth
  - Measures how well you expect to represent true solution
  - Decreases with more complex model
- **Variance:** difference between what you expect to learn and what you learn from a from a particular dataset
  - Measures how sensitive learner is to specific dataset
  - Increases with more complex model

# Bias-variance decomposition



- Example: polynomial regression  $h(x) = \sum_{k=0}^p w_k x^k$



- Value of the optimal (ML) regression coefficients

	$p=0$	$p=1$	$p=3$	$p=9$
$w_0^*$	0.19	0.82	0.31	0.35
$w_1^*$		-1.27	7.99	232.37
$w_2^*$			-25.43	-5321.83
$w_3^*$			17.37	48568.31
$w_4^*$				-231639.30
$w_5^*$				640042.26
$w_6^*$				-1061800.52
$w_7^*$				1042400.18
$w_8^*$				-557682.99
$w_9^*$				125201.43

- Assume average of  $k$  nearest neighbors

$$\begin{aligned} Err(\mathbf{x}_0) &= \mathbb{E} \left[ (y - \hat{f}(\mathbf{x}_0))^2 | \mathbf{x} = \mathbf{x}_0 \right] \\ &= \sigma_\varepsilon^2 + \left[ f(\mathbf{x}_0) - \frac{1}{k} \sum_{\mathbf{x}_i \in k\text{NN}(\mathbf{x}_0)} y_i \right]^2 + \frac{\sigma_\varepsilon^2}{k} \end{aligned}$$

- For small  $k$ , good fit (small bias), larger variance. For big  $k$ , more bias, less variance
- This is a model selection problem

- Given a set of models  $F = \{F_1, \dots, F_K\}$ , choose the model **expected to do the best on the test data**
- $F$  may consist of
  - Same learning model with different complexities or hyperparameters
    - Nonlinear regression: polynomials with different degrees
    - $k$ -Nearest Neighbors: Different choices of  $k$
    - Decision Trees: different choices of the number of levels/leaves
    - SVM: different choices of the misclassification penalty hyperparameter  $C$
    - Regularized Models: different choices of the regularization parameter
    - Kernel based Methods: different choices of kernels, etc.
    - Different subsets of features  $(x_j, j \in J)$ ,  $J \subseteq \{1, 2, \dots, d\}$
  - Different learning models (e.g., SVM, KNN, DT, etc.)
- Note:** usually considered in supervised learning contexts but unsupervised learning also faces this issue (e.g., "how many clusters" when doing clustering)

- Given a set of models  $F = \{F_1, \dots, F_K\}$ , choose the model **expected to do the best on the test data**
- $F$  may consist of
  - Same learning model with different complexities or hyperparameters
    - Nonlinear regression: polynomials with different degrees
    - $k$ -Nearest Neighbors: Different choices of  $k$
    - Decision Trees: different choices of the number of levels/leaves
    - SVM: different choices of the misclassification penalty hyperparameter  $C$
    - Regularized Models: different choices of the regularization parameter
    - Kernel based Methods: different choices of kernels, etc.
    - Different subsets of features  $(x_j, j \in J)$ ,  $J \subseteq \{1, 2, \dots, d\}$
  - Different learning models (e.g., SVM, KNN, DT, etc.)
- Note:** usually considered in supervised learning contexts but unsupervised learning also faces this issue (e.g., "how many clusters" when doing clustering)

- Given a set of models  $F = \{F_1, \dots, F_K\}$ , choose the model **expected to do the best on the test data**
- $F$  may consist of
  - Same learning model with different complexities or hyperparameters
    - Nonlinear regression: polynomials with different degrees
    - $k$ -Nearest Neighbors: Different choices of  $k$
    - Decision Trees: different choices of the number of levels/leaves
    - SVM: different choices of the misclassification penalty hyperparameter  $C$
    - Regularized Models: different choices of the regularization parameter
    - Kernel based Methods: different choices of kernels, etc.
    - Different subsets of features  $(x_j, j \in J)$ ,  $J \subseteq \{1, 2, \dots, d\}$
  - Different learning models (e.g., SVM, KNN, DT, etc.)
- Note:** usually considered in supervised learning contexts but unsupervised learning also faces this issue (e.g., "how many clusters" when doing clustering)

- Given a set of models  $F = \{F_1, \dots, F_K\}$ , choose the model **expected to do the best on the test data**
- $F$  may consist of
  - Same learning model with different complexities or hyperparameters
    - Nonlinear regression: polynomials with different degrees
    - $k$ -Nearest Neighbors: Different choices of  $k$
    - Decision Trees: different choices of the number of levels/leaves
    - SVM: different choices of the misclassification penalty hyperparameter  $C$
    - Regularized Models: different choices of the regularization parameter
    - Kernel based Methods: different choices of kernels, etc.
    - Different subsets of features  $(x_j, j \in J)$ ,  $J \subseteq \{1, 2, \dots, d\}$
  - Different learning models (e.g., SVM, KNN, DT, etc.)
- Note:** usually considered in supervised learning contexts but unsupervised learning also faces this issue (e.g., "how many clusters" when doing clustering)

- Occam's razor: among competing hypotheses, the one with the fewest assumptions should be selected
- Too much variables/parameters  $\Rightarrow$  significant prediction variance and small bias on the training sample, and vice versa
- We have two interrelated problems
  - Problem 1. To estimate value of a target function, characterizing generalization ability of the considered model
  - Problem 2. To computationally efficiently select an optimal model w.r.t. to the constructed accuracy criterion

- Occam's razor: among competing hypotheses, the one with the fewest assumptions should be selected
- Too much variables/parameters  $\Rightarrow$  significant prediction variance and small bias on the training sample, and vice versa
- We have two interrelated problems
  - Problem 1. To estimate value of a target function, characterizing generalization ability of the considered model
  - Problem 2. To computationally efficiently select an optimal model w.r.t. to the constructed accuracy criterion

- Occam's razor: among competing hypotheses, the one with the fewest assumptions should be selected
- Too much variables/parameters  $\Rightarrow$  significant prediction variance and small bias on the training sample, and vice versa
- We have **two interrelated problems**
  - **Problem 1.** To estimate value of a target function, characterizing generalization ability of the considered model
  - **Problem 2.** To computationally efficiently select an optimal model w.r.t. to the constructed accuracy criterion

- Occam's razor: among competing hypotheses, the one with the fewest assumptions should be selected
- Too much variables/parameters  $\Rightarrow$  significant prediction variance and small bias on the training sample, and vice versa
- We have **two interrelated problems**
  - **Problem 1.** To estimate value of a target function, characterizing generalization ability of the considered model
  - **Problem 2.** To computationally efficiently select an optimal model w.r.t. to the constructed accuracy criterion

1 Overfitting. Bias-variance decomposition

2 Model Quality Criteria

3 Feature Selection

4 Regularization

5 Linear Regression Model Selection

6 Appendix. Materials for Additional Study: Sensitivity Analysis

- Input:
  - Training sample  $S_m = \{(\mathbf{x}_i, y_i)\}_{i=1}^m$ ,  $\mathbf{x}_i \in X$ ,  $y_i \in Y$
  - $F_n = \{f : X \rightarrow Y\}$  is a hypothesis set,  $n = 1, 2, \dots$
  - $\mathcal{A}_n : (X \times Y)^m \rightarrow F_n$  is a learner,  $n = 1, 2, \dots$
- Output: obtain a learner with the best generalization ability
- Examples:
  - Select the best model  $F_n$  (e.g., SVM vs. DT)
  - Select the best learner  $\mathcal{A}_n$  for a given model  $F_n$  (e.g. tuning of  $C$  for SVM)
  - Select subset of features  $\mathbf{x}_J = (x_j, j \in J)$  from the available features  $\mathbf{x} = (x_1, \dots, x_d)$ , i.e. learner  $\mathcal{A}_n$  uses only features  $\mathbf{x}_J$

- Input:
  - Training sample  $S_m = \{(\mathbf{x}_i, y_i)\}_{i=1}^m$ ,  $\mathbf{x}_i \in X$ ,  $y_i \in Y$
  - $F_n = \{f : X \rightarrow Y\}$  is a hypothesis set,  $n = 1, 2, \dots$
  - $\mathcal{A}_n : (X \times Y)^m \rightarrow F_n$  is a learner,  $n = 1, 2, \dots$
- Output: obtain a learner with the best generalization ability
- Examples:
  - Select the best model  $F_n$  (e.g., SVM vs. DT)
  - Select the best learner  $\mathcal{A}_n$  for a given model  $F_n$  (e.g. tuning of  $C$  for SVM)
  - Select subset of features  $\mathbf{x}_J = (x_j, j \in J)$  from the available features  $\mathbf{x} = (x_1, \dots, x_d)$ , i.e. learner  $\mathcal{A}_n$  uses only features  $\mathbf{x}_J$

- Input:
  - Training sample  $S_m = \{(\mathbf{x}_i, y_i)\}_{i=1}^m$ ,  $\mathbf{x}_i \in X$ ,  $y_i \in Y$
  - $F_n = \{f : X \rightarrow Y\}$  is a hypothesis set,  $n = 1, 2, \dots$
  - $\mathcal{A}_n : (X \times Y)^m \rightarrow F_n$  is a learner,  $n = 1, 2, \dots$
- Output: obtain a learner with the best generalization ability
- Examples:
  - Select the best model  $F_n$  (e.g., SVM vs. DT)
  - Select the best learner  $\mathcal{A}_n$  for a given model  $F_n$  (e.g. tuning of  $C$  for SVM)
  - Select subset of features  $\mathbf{x}_J = (x_j, j \in J)$  from the available features  $\mathbf{x} = (x_1, \dots, x_d)$ , i.e. learner  $\mathcal{A}_n$  uses only features  $\mathbf{x}_J$

- Input:
  - Training sample  $S_m = \{(\mathbf{x}_i, y_i)\}_{i=1}^m$ ,  $\mathbf{x}_i \in X$ ,  $y_i \in Y$
  - $F_n = \{f : X \rightarrow Y\}$  is a hypothesis set,  $n = 1, 2, \dots$
  - $\mathcal{A}_n : (X \times Y)^m \rightarrow F_n$  is a learner,  $n = 1, 2, \dots$
- Output: obtain a learner with the best generalization ability
- Examples:

- Select the best model  $F_n$  (e.g., SVM vs. DT)
- Select the best learner  $\mathcal{A}_n$  for a given model  $F_n$  (e.g. tuning of  $C$  for SVM)
- Select subset of features  $\mathbf{x}_J = (x_j, j \in J)$  from the available features  $\mathbf{x} = (x_1, \dots, x_d)$ , i.e. learner  $\mathcal{A}_n$  uses only features  $\mathbf{x}_J$

- Input:
  - Training sample  $S_m = \{(\mathbf{x}_i, y_i)\}_{i=1}^m$ ,  $\mathbf{x}_i \in X$ ,  $y_i \in Y$
  - $F_n = \{f : X \rightarrow Y\}$  is a hypothesis set,  $n = 1, 2, \dots$
  - $\mathcal{A}_n : (X \times Y)^m \rightarrow F_n$  is a learner,  $n = 1, 2, \dots$
- Output: obtain a learner with the best generalization ability
- Examples:
  - Select the best model  $F_n$  (e.g., SVM vs. DT)
  - Select the best learner  $\mathcal{A}_n$  for a given model  $F_n$  (e.g. tuning of  $C$  for SVM)
  - Select subset of features  $\mathbf{x}_J = (x_j, j \in J)$  from the available features  $\mathbf{x} = (x_1, \dots, x_d)$ , i.e. learner  $\mathcal{A}_n$  uses only features  $\mathbf{x}_J$

- Input:
  - Training sample  $S_m = \{(\mathbf{x}_i, y_i)\}_{i=1}^m$ ,  $\mathbf{x}_i \in X$ ,  $y_i \in Y$
  - $F_n = \{f : X \rightarrow Y\}$  is a hypothesis set,  $n = 1, 2, \dots$
  - $\mathcal{A}_n : (X \times Y)^m \rightarrow F_n$  is a learner,  $n = 1, 2, \dots$
- Output: obtain a learner with the best generalization ability
- Examples:
  - Select the best model  $F_n$  (e.g., SVM vs. DT)
  - Select the best learner  $\mathcal{A}_n$  for a given model  $F_n$  (e.g. tuning of  $C$  for SVM)
  - Select subset of features  $\mathbf{x}_J = (x_j, j \in J)$  from the available features  $\mathbf{x} = (x_1, \dots, x_d)$ , i.e. learner  $\mathcal{A}_n$  uses only features  $\mathbf{x}_J$

- Input:
  - Training sample  $S_m = \{(\mathbf{x}_i, y_i)\}_{i=1}^m$ ,  $\mathbf{x}_i \in X$ ,  $y_i \in Y$
  - $F_n = \{f : X \rightarrow Y\}$  is a hypothesis set,  $n = 1, 2, \dots$
  - $\mathcal{A}_n : (X \times Y)^m \rightarrow F_n$  is a learner,  $n = 1, 2, \dots$
- Output: obtain a learner with the best generalization ability
- Examples:
  - Select the best model  $F_n$  (e.g., SVM vs. DT)
  - Select the best learner  $\mathcal{A}_n$  for a given model  $F_n$  (e.g. tuning of  $C$  for SVM)
  - Select subset of features  $\mathbf{x}_J = (x_j, j \in J)$  from the available features  $\mathbf{x} = (x_1, \dots, x_d)$ , i.e. learner  $\mathcal{A}_n$  uses only features  $\mathbf{x}_J$

- $L(f(\mathbf{x}), y)$  is a loss for a pair  $(\mathbf{x}, y)$  and a model  $f$
- $\widehat{R}(f; S_m) = \frac{1}{m} \sum_{i=1}^m L(f(\mathbf{x}_i), y_i)$  is a loss of  $f$  on  $S_m$
- Empirical Training Error

$$\widehat{R}(f; S_m), f(\cdot) = \mathcal{A}(S_m)$$

This error is a biased estimate of the generalization risk

- Empirical Test Error is estimated using a hold-out test sample  $S'$

$$\widehat{R}(f; S'), f(\cdot) = \mathcal{A}(S_m)$$

- either we need an additional test set  $S'$
- or we have to divide  $S_m$  into a train set and a validation set  
(results depend on this division)

- $L(f(\mathbf{x}), y)$  is a loss for a pair  $(\mathbf{x}, y)$  and a model  $f$
- $\widehat{R}(f; S_m) = \frac{1}{m} \sum_{i=1}^m L(f(\mathbf{x}_i), y_i)$  is a loss of  $f$  on  $S_m$
- Empirical Training Error

$$\widehat{R}(f; S_m), f(\cdot) = \mathcal{A}(S_m)$$

This error is a biased estimate of the generalization risk

- Empirical Test Error is estimated using a hold-out test sample  $S'$

$$\widehat{R}(f; S'), f(\cdot) = \mathcal{A}(S_m)$$

- either we need an additional test set  $S'$
- or we have to divide  $S_m$  into a train set and a validation set  
(results depend on this division)

- $L(f(\mathbf{x}), y)$  is a loss for a pair  $(\mathbf{x}, y)$  and a model  $f$
- $\widehat{R}(f; S_m) = \frac{1}{m} \sum_{i=1}^m L(f(\mathbf{x}_i), y_i)$  is a loss of  $f$  on  $S_m$
- Empirical Training Error

$$\widehat{R}(f; S_m), f(\cdot) = \mathcal{A}(S_m)$$

This error is a biased estimate of the generalization risk

- Empirical Test Error is estimated using a hold-out test sample  $S'$

$$\widehat{R}(f; S'), f(\cdot) = \mathcal{A}(S_m)$$

- either we need an additional test set  $S'$
- or we have to divide  $S_m$  into a train set and a validation set  
(results depend on this division)

- $L(f(\mathbf{x}), y)$  is a loss for a pair  $(\mathbf{x}, y)$  and a model  $f$
- $\widehat{R}(f; S_m) = \frac{1}{m} \sum_{i=1}^m L(f(\mathbf{x}_i), y_i)$  is a loss of  $f$  on  $S_m$
- Empirical Training Error

$$\widehat{R}(f; S_m), f(\cdot) = \mathcal{A}(S_m)$$

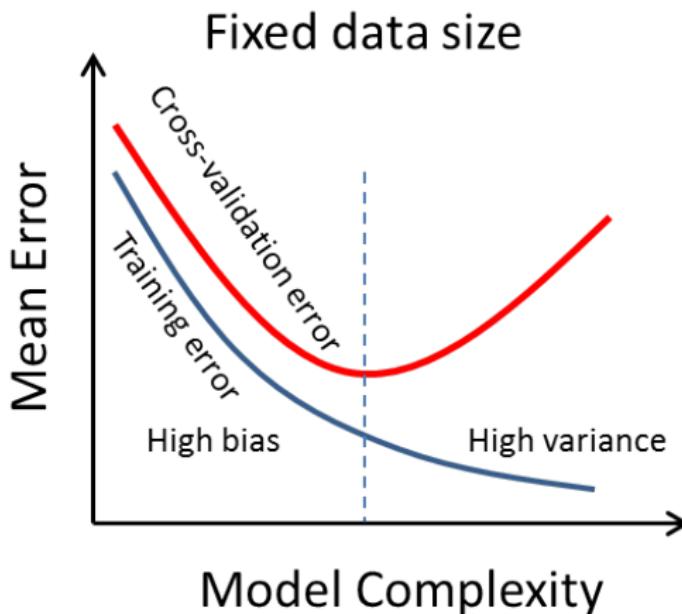
This error is a biased estimate of the generalization risk

- Empirical Test Error is estimated using a hold-out test sample  $S'$

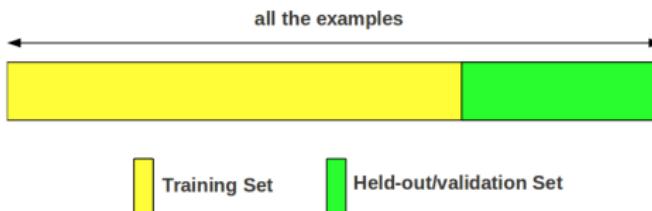
$$\widehat{R}(f; S'), f(\cdot) = \mathcal{A}(S_m)$$

- either we need an additional test set  $S'$
- or we have to divide  $S_m$  into a train set and a validation set  
(results depend on this division)

- Train Error decreases w.r.t. increasing model complexity
- Test Error increases w.r.t. increasing model complexity

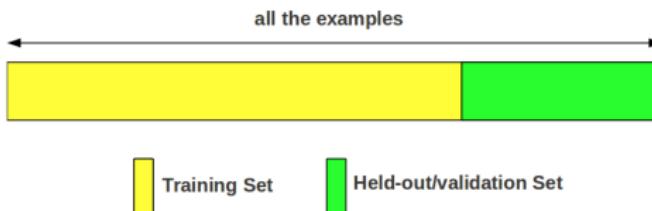


- Set aside a fraction (say 10% – 20%) of the training data
- This part becomes our hold-out data (validation or development data)



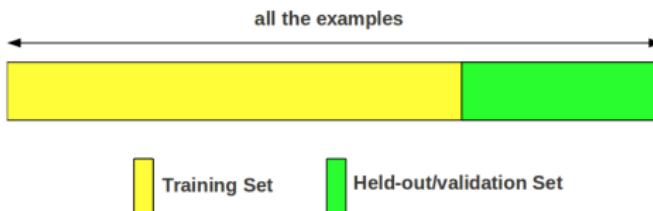
- Remember: hold-out data is NOT the test data
  - Train each model using the remaining training data
  - Evaluate error on the hold-out data
  - Choose the model with the smallest hold-out error
  - Problems:
    - Wastes training data, so typically used when we have plenty of training data
    - Hold-out data may not be good if there was an unfortunate split (use random splitting!)

- Set aside a fraction (say 10% – 20%) of the training data
- This part becomes our hold-out data (validation or development data)



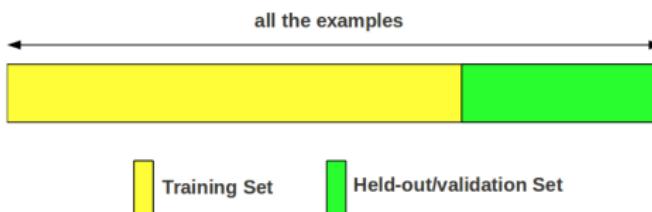
- Remember: hold-out data is NOT the test data
- Train each model using the remaining training data
- Evaluate error on the hold-out data
- Choose the model with the smallest hold-out error
- Problems:
  - Wastes training data, so typically used when we have plenty of training data
  - Hold-out data may not be good if there was an unfortunate split (use random splitting!)

- Set aside a fraction (say 10% – 20%) of the training data
- This part becomes our hold-out data (validation or development data)



- Remember: hold-out data is NOT the test data
- Train each model using the remaining training data
- Evaluate error on the hold-out data
- Choose the model with the smallest hold-out error
- Problems:
  - Wastes training data, so typically used when we have plenty of training data
  - Hold-out data may not be good if there was an unfortunate split (use random splitting!)

- Set aside a fraction (say 10% – 20%) of the training data
- This part becomes our hold-out data (validation or development data)



- Remember: hold-out data is NOT the test data
- Train each model using the remaining training data
- Evaluate error on the hold-out data
- Choose the model with the smallest hold-out error
- Problems:
  - Wastes training data, so typically used when we have plenty of training data
  - Hold-out data may not be good if there was an unfortunate split (use random splitting!)

## $M$ -fold Cross-Validation

- Create  $M$  equal sized partitions of the training data
- Each partition has  $m/M$  examples
- Train using  $M - 1$  partitions, validate on the remaining partitions
- Repeat the same  $M$  times, each with a different validation partition



- Finally, choose the model with smallest average validation error
- Usually  $M$  is chosen as  $4 - 10$

## $M$ -fold Cross-Validation

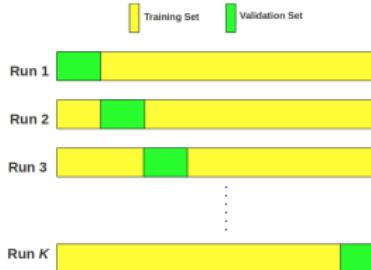
- Create  $M$  equal sized partitions of the training data
- Each partition has  $m/M$  examples
- Train using  $M - 1$  partitions, validate on the remaining partitions
- Repeat the same  $M$  times, each with a different validation partition



- Finally, choose the model with smallest average validation error
- Usually  $M$  is chosen as  $4 - 10$

## $M$ -fold Cross-Validation

- Create  $M$  equal sized partitions of the training data
- Each partition has  $m/M$  examples
- Train using  $M - 1$  partitions, validate on the remaining partitions
- Repeat the same  $M$  times, each with a different validation partition



- Finally, choose the model with smallest average validation error
- Usually  $M$  is chosen as  $4 - 10$

## $N \times M$ -fold Cross-Validation

- We divide the training sample  $N$  times into  $M$  equal sized partitions
- Results of all  $N$   $M$ -fold cross-validations are aggregated (e.g. averaged)
- By increasing  $N$  we can improve accuracy
- Each object is used in a test set  $N$  times
- We can construct confidence intervals using results of  $N$  repetitions

## $N \times M$ -fold Cross-Validation

- We divide the training sample  $N$  times into  $M$  equal sized partitions
- Results of all  $N$   $M$ -fold cross-validations are aggregated (e.g. averaged)
- By increasing  $N$  we can improve accuracy
- Each object is used in a test set  $N$  times
- We can construct confidence intervals using results of  $N$  repetitions

## $N \times M$ -fold Cross-Validation

- We divide the training sample  $N$  times into  $M$  equal sized partitions
- Results of all  $N$   $M$ -fold cross-validations are aggregated (e.g. averaged)
- By increasing  $N$  we can improve accuracy
- Each object is used in a test set  $N$  times
- We can construct confidence intervals using results of  $N$  repetitions

- If a hypothesis set  $F$  is appropriate, then models  $f = \mathcal{A}(S)$ , constructed for different subsets  $S \subset S_m$  should be “similar”
- E.g. we can divide  $S_m$   $N$ -times into two parts  $\{S_1^i \cup S_2^i\}_{i=1}^N$ , and calculate

$$\Delta_N(\mathcal{A}; S_m) = \frac{1}{N} \sum_{i=1}^N \frac{1}{m} \sum_{j=1}^m |\mathcal{A}(S_1^i)(\mathbf{x}_j) - \mathcal{A}(S_2^i)(\mathbf{x}_j)|$$

- Problems:
  - Sample size is two times smaller
  - Computational complexity is two times bigger

- If a hypothesis set  $F$  is appropriate, then models  $f = \mathcal{A}(S)$ , constructed for different subsets  $S \subset S_m$  should be “similar”
- E.g. we can divide  $S_m$   $N$ -times into two parts  $\{S_1^i \cup S_2^i\}_{i=1}^N$ , and calculate

$$\Delta_N(\mathcal{A}; S_m) = \frac{1}{N} \sum_{i=1}^N \frac{1}{m} \sum_{j=1}^m |\mathcal{A}(S_1^i)(\mathbf{x}_j) - \mathcal{A}(S_2^i)(\mathbf{x}_j)|$$

- Problems:
  - Sample size is two times smaller
  - Computational complexity is two times bigger

- If a hypothesis set  $F$  is appropriate, then models  $f = \mathcal{A}(S)$ , constructed for different subsets  $S \subset S_m$  should be “similar”
- E.g. we can divide  $S_m$   $N$ -times into two parts  $\{S_1^i \cup S_2^i\}_{i=1}^N$ , and calculate

$$\Delta_N(\mathcal{A}; S_m) = \frac{1}{N} \sum_{i=1}^N \frac{1}{m} \sum_{j=1}^m |\mathcal{A}(S_1^i)(\mathbf{x}_j) - \mathcal{A}(S_2^i)(\mathbf{x}_j)|$$

- Problems:
  - Sample size is two times smaller
  - Computational complexity is two times bigger

1 Overfitting. Bias-variance decomposition

2 Model Quality Criteria

3 Feature Selection

4 Regularization

5 Linear Regression Model Selection

6 Appendix. Materials for Additional Study: Sensitivity Analysis

## Why Feature Selection?

- Some algorithms scale (computationally) poorly with increased dimension
- Irrelevant features can confuse some algorithms
- Redundant features adversely affect regularization
- Removal of features can increase (relative) margin (and generalization)
- Reduces data set and resulting model size
- **Note:** Feature Selection is different from Feature Extraction
  - The latter transforms original features to get a small set of new features
  - Dimensionality Reduction is a type of Feature Extraction

## Why Feature Selection?

- Some algorithms scale (computationally) poorly with increased dimension
- Irrelevant features can confuse some algorithms
- Redundant features adversely affect regularization
- Removal of features can increase (relative) margin (and generalization)
- Reduces data set and resulting model size
- **Note:** Feature Selection is different from Feature Extraction
  - The latter transforms original features to get a small set of new features
  - Dimensionality Reduction is a type of Feature Extraction

## Why Feature Selection?

- Some algorithms scale (computationally) poorly with increased dimension
- Irrelevant features can confuse some algorithms
- Redundant features adversely affect regularization
- Removal of features can increase (relative) margin (and generalization)
- Reduces data set and resulting model size
- **Note:** Feature Selection is different from Feature Extraction
  - The latter transforms original features to get a small set of new features
  - Dimensionality Reduction is a type of Feature Extraction

## Why Feature Selection?

- Some algorithms scale (computationally) poorly with increased dimension
- Irrelevant features can confuse some algorithms
- Redundant features adversely affect regularization
- Removal of features can increase (relative) margin (and generalization)
- Reduces data set and resulting model size
- Note: Feature Selection is different from Feature Extraction
  - The latter transforms original features to get a small set of new features
  - Dimensionality Reduction is a type of Feature Extraction

## Why Feature Selection?

- Some algorithms scale (computationally) poorly with increased dimension
- Irrelevant features can confuse some algorithms
- Redundant features adversely affect regularization
- Removal of features can increase (relative) margin (and generalization)
- Reduces data set and resulting model size
- Note: Feature Selection is different from Feature Extraction
  - The latter transforms original features to get a small set of new features
  - Dimensionality Reduction is a type of Feature Extraction

## Why Feature Selection?

- Some algorithms scale (computationally) poorly with increased dimension
- Irrelevant features can confuse some algorithms
- Redundant features adversely affect regularization
- Removal of features can increase (relative) margin (and generalization)
- Reduces data set and resulting model size
- **Note:** Feature Selection is different from Feature Extraction
  - The latter transforms original features to get a small set of new features
  - Dimensionality Reduction is a type of Feature Extraction

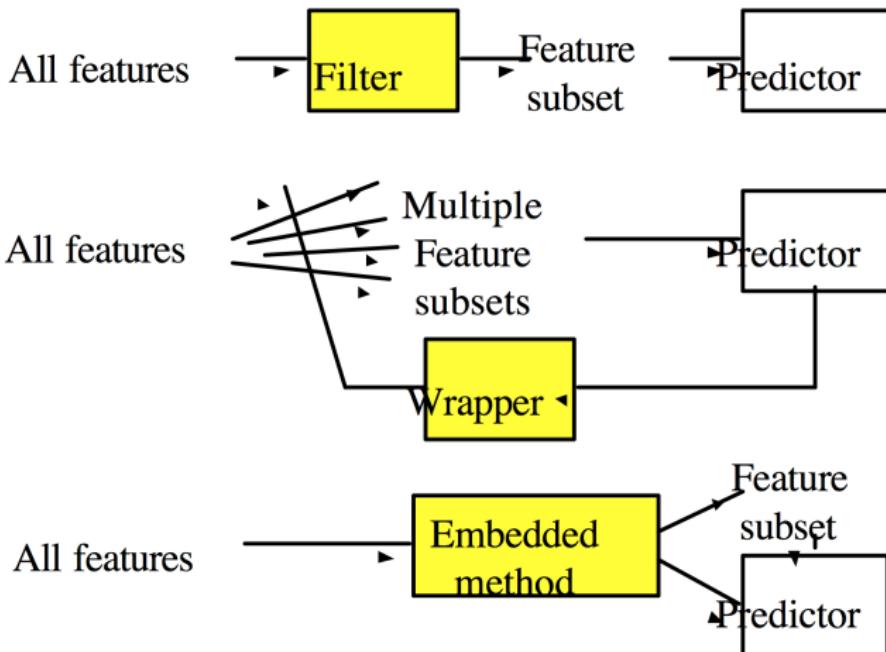
- Methods agnostic to the learning algorithm
  - Preprocessing based methods
    - a) E.g., remove a binary feature if it's ON in very few or most examples
  - Filter Feature Selection methods
    - a) Use some ranking criteria to rank features
    - b) Select the top ranking features
  - Wrapper Methods (keep the learning algorithm in the loop)
    - a) Requires repeated runs of the learning algorithm with different set of features
    - b) Can be computationally expensive

- Methods agnostic to the learning algorithm
  - Preprocessing based methods
    - a) E.g., remove a binary feature if it's ON in very few or most examples
  - Filter Feature Selection methods
    - a) Use some ranking criteria to rank features
    - b) Select the top ranking features
  - Wrapper Methods (keep the learning algorithm in the loop)
    - a) Requires repeated runs of the learning algorithm with different set of features
    - b) Can be computationally expensive

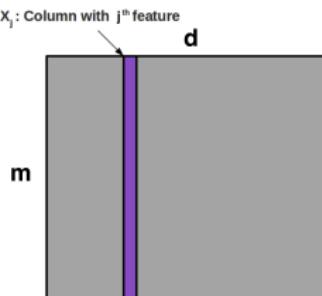
- Methods agnostic to the learning algorithm
  - Preprocessing based methods
    - a) E.g., remove a binary feature if it's ON in very few or most examples
  - Filter Feature Selection methods
    - a) Use some ranking criteria to rank features
    - b) Select the top ranking features
  - Wrapper Methods (keep the learning algorithm in the loop)
    - a) Requires repeated runs of the learning algorithm with different set of features
    - b) Can be computationally expensive

# Scheme of Feature Selection Methods

Filters, Wrappers, and Embedded methods



- Uses heuristics but is much faster than wrapper methods



- Correlation Criteria: rank features in order of their correlation with the labels

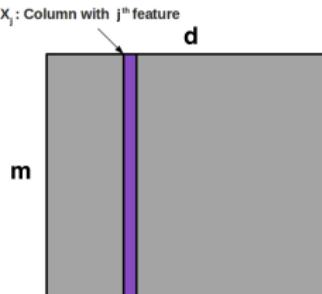
$$r(x_j, y) = \frac{\text{cov}(x_j, y)}{\sqrt{\text{var}(x_j)\text{var}(y)}}$$

- Mutual Information Criteria:

$$MI(x_j, y) = \sum_{x_j} \sum_y p(x_j, y) \log \frac{p(x_j, y)}{p(x_j)p(y)}$$

- High mutual information mean high relevance of that feature
- Note: these probabilities can be easily estimated from the data

- Uses heuristics but is much faster than wrapper methods



- Correlation Criteria: rank features in order of their correlation with the labels

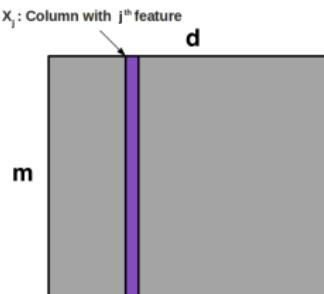
$$r(x_j, y) = \frac{\text{cov}(x_j, y)}{\sqrt{\text{var}(x_j)\text{var}(y)}}$$

- Mutual Information Criteria:

$$MI(x_j, y) = \sum_{x_j} \sum_y p(x_j, y) \log \frac{p(x_j, y)}{p(x_j)p(y)}$$

- High mutual information mean high relevance of that feature
- Note: these probabilities can be easily estimated from the data

- Uses heuristics but is much faster than wrapper methods



- Correlation Criteria: rank features in order of their correlation with the labels

$$r(x_j, y) = \frac{\text{cov}(x_j, y)}{\sqrt{\text{var}(x_j)\text{var}(y)}}$$

- Mutual Information Criteria:

$$MI(x_j, y) = \sum_{x_j} \sum_y p(x_j, y) \log \frac{p(x_j, y)}{p(x_j)p(y)}$$

- High mutual information mean high relevance of that feature
- Note: these probabilities can be easily estimated from the data

Two types: Forward Search and Backward Search

- Forward Search
  - Start with no features
  - Greedily include the most relevant feature and estimate model's error on a renewed feature set
  - Stop when selected the desired number of features
- Backward Search
  - Start with all the features
  - Greedily remove the least relevant and estimate model's error on a renewed feature set
  - Stop when selected the desired number of features
- Inclusion/Removal criteria uses cross-validation

Two types: Forward Search and Backward Search

- Forward Search
  - Start with no features
  - Greedily include the most relevant feature and estimate model's error on a renewed feature set
  - Stop when selected the desired number of features
- Backward Search
  - Start with all the features
  - Greedily remove the least relevant and estimate model's error on a renewed feature set
  - Stop when selected the desired number of features
- Inclusion/Removal criteria uses cross-validation

Two types: Forward Search and Backward Search

- Forward Search
  - Start with no features
  - Greedily include the most relevant feature and estimate model's error on a renewed feature set
  - Stop when selected the desired number of features
- Backward Search
  - Start with all the features
  - Greedily remove the least relevant and estimate model's error on a renewed feature set
  - Stop when selected the desired number of features
- Inclusion/Removal criteria uses cross-validation

- **Optimization problem:** “Least Absolute Shrinkage and Selection Operator” (see lecture 2 on “Intro to Regression”)

$$F(\mathbf{w}, b) = \lambda \|\mathbf{w}\|_1 + \sum_{i=1}^m (\mathbf{w} \cdot \mathbf{x}_i^\top + b - y_i)^2 \rightarrow \min_{\mathbf{w}, b}$$

where  $\lambda \geq 0$  is a regularization parameter

- **Solution:** equivalent convex quadratic programming (QP)
  - general: standard QP solvers
  - specific algorithms: LARS (least angle regression procedure), entire path of solution
- Generalization: Elastic Net

- **Optimization problem:** “Least Absolute Shrinkage and Selection Operator” (see lecture 2 on “Intro to Regression”)

$$F(\mathbf{w}, b) = \lambda \|\mathbf{w}\|_1 + \sum_{i=1}^m (\mathbf{w} \cdot \mathbf{x}_i^\top + b - y_i)^2 \rightarrow \min_{\mathbf{w}, b}$$

where  $\lambda \geq 0$  is a regularization parameter

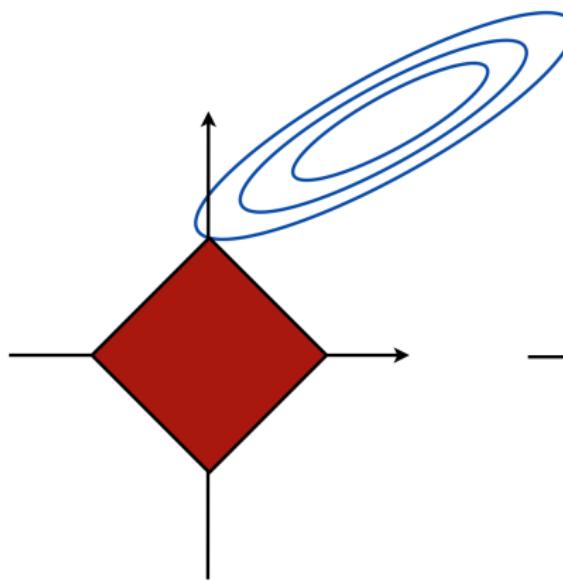
- **Solution:** equivalent convex quadratic programming (QP)
  - general: standard QP solvers
  - specific algorithms: LARS (least angle regression procedure), entire path of solution
- Generalization: Elastic Net

- **Optimization problem:** “Least Absolute Shrinkage and Selection Operator” (see lecture 2 on “Intro to Regression”)

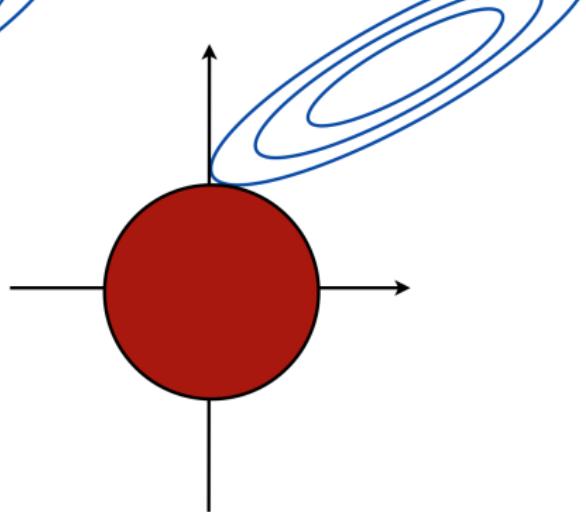
$$F(\mathbf{w}, b) = \lambda \|\mathbf{w}\|_1 + \sum_{i=1}^m (\mathbf{w} \cdot \mathbf{x}_i^\top + b - y_i)^2 \rightarrow \min_{\mathbf{w}, b}$$

where  $\lambda \geq 0$  is a regularization parameter

- **Solution:** equivalent convex quadratic programming (QP)
  - general: standard QP solvers
  - specific algorithms: LARS (least angle regression procedure), entire path of solution
- Generalization: Elastic Net



$L_1$  regularization



$L_2$  regularization

1 Overfitting. Bias-variance decomposition

2 Model Quality Criteria

3 Feature Selection

4 Regularization

5 Linear Regression Model Selection

6 Appendix. Materials for Additional Study: Sensitivity Analysis

- Upper bound on a probability of over-training, valid for any sample  $S_m$ , rather general hypothesis class  $F$  and learner  $\mathcal{A}$ :

$$f(\cdot) = \mathcal{A}(S_m)$$

$$\mathbb{P}\left(\widehat{R}(f; S') - \widehat{R}(f; S_m) \geq \varepsilon\right) \leq \delta(\varepsilon, F)$$

- Then for any  $S_m$ ,  $F$  and  $\mathcal{A}$ , and  $\delta \in (0, 1)$  with a probability not less than  $(1 - \delta)$  we get that

$$\widehat{R}(f; S') \leq \widehat{R}(f; S_m) + \varepsilon(\delta, F)$$

- Corrected Empirical Risk

$$\widehat{R}(f; S_m) + \varepsilon(\delta, F) \rightarrow \min_{\{f \in F\}, F}$$

- Upper bound on a probability of over-training, valid for any sample  $S_m$ , rather general hypothesis class  $F$  and learner  $\mathcal{A}$ :

$$f(\cdot) = \mathcal{A}(S_m)$$

$$\mathbb{P}\left(\widehat{R}(f; S') - \widehat{R}(f; S_m) \geq \varepsilon\right) \leq \delta(\varepsilon, F)$$

- Then for any  $S_m$ ,  $F$  and  $\mathcal{A}$ , and  $\delta \in (0, 1)$  with a probability not less than  $(1 - \delta)$  we get that

$$\widehat{R}(f; S') \leq \widehat{R}(f; S_m) + \varepsilon(\delta, F)$$

- Corrected Empirical Risk

$$\widehat{R}(f; S_m) + \varepsilon(\delta, F) \rightarrow \min_{\{f \in F\}, F}$$

- Upper bound on a probability of over-training, valid for any sample  $S_m$ , rather general hypothesis class  $F$  and learner  $\mathcal{A}$ :

$$f(\cdot) = \mathcal{A}(S_m)$$

$$\mathbb{P} \left( \widehat{R}(f; S') - \widehat{R}(f; S_m) \geq \varepsilon \right) \leq \delta(\varepsilon, F)$$

- Then for any  $S_m$ ,  $F$  and  $\mathcal{A}$ , and  $\delta \in (0, 1)$  with a probability not less than  $(1 - \delta)$  we get that

$$\widehat{R}(f; S') \leq \widehat{R}(f; S_m) + \varepsilon(\delta, F)$$

- Corrected Empirical Risk

$$\widehat{R}(f; S_m) + \varepsilon(\delta, F) \rightarrow \min_{\{f \in F\}, F}$$

- Upper bound on a probability of over-training, valid for any sample  $S_m$ , rather general hypothesis class  $F$  and learner  $\mathcal{A}$ :

$$f(\cdot) = \mathcal{A}(S_m)$$

$$\mathbb{P} \left( \widehat{R}(f; S') - \widehat{R}(f; S_m) \geq \varepsilon \right) \leq \delta(\varepsilon, F)$$

- Then for any  $S_m$ ,  $F$  and  $\mathcal{A}$ , and  $\delta \in (0, 1)$  with a probability not less than  $(1 - \delta)$  we get that

$$\widehat{R}(f; S') \leq \widehat{R}(f; S_m) + \varepsilon(\delta, F)$$

- Corrected Empirical Risk

$$\widehat{R}(f; S_m) + \varepsilon(\delta, F) \rightarrow \min_{\{f \in F\}, F}$$

- Upper bound on a probability of over-training, valid for any sample  $S_m$ , rather general hypothesis class  $F$  and learner  $\mathcal{A}$ :

$$f(\cdot) = \mathcal{A}(S_m)$$

$$\mathbb{P} \left( \widehat{R}(f; S') - \widehat{R}(f; S_m) \geq \varepsilon \right) \leq \delta(\varepsilon, F)$$

- Then for any  $S_m$ ,  $F$  and  $\mathcal{A}$ , and  $\delta \in (0, 1)$  with a probability not less than  $(1 - \delta)$  we get that

$$\widehat{R}(f; S') \leq \widehat{R}(f; S_m) + \varepsilon(\delta, F)$$

- Corrected Empirical Risk

$$\widehat{R}(f; S_m) + \varepsilon(\delta, F) \rightarrow \min_{\{f \in F\}, F}$$

- Regularization penalizes complex models  $F$

$$\widehat{R}_{\text{pen}}(f; S_m) = \widehat{R}(f; S_m) + \text{pen}(F)$$

- Let us consider linear models
  - $F = \{f(\mathbf{x}) = \text{sign}(\mathbf{w} \cdot \mathbf{x}^\top)\}$  (classification) or
  - $F = \{f(\mathbf{x}) = (\mathbf{w} \cdot \mathbf{x}^\top)\}$  (regression)
- Then
  - a)  $L_2$ -regularization  $\text{pen}(F) = \lambda \sum_{j=1}^d w_j^2$
  - b)  $L_1$ -regularization  $\text{pen}(F) = \lambda \sum_{j=1}^d |w_j|$
  - c)  $L_0$ -regularization  $\text{pen}(F) = \lambda \sum_{j=1}^d 1_{w_j \neq 0}$

- Regularization penalizes complex models  $F$

$$\widehat{R}_{\text{pen}}(f; S_m) = \widehat{R}(f; S_m) + \text{pen}(F)$$

- Let us consider linear models
  - $F = \{f(\mathbf{x}) = \text{sign}(\mathbf{w} \cdot \mathbf{x}^\top)\}$  (classification) or
  - $F = \{f(\mathbf{x}) = (\mathbf{w} \cdot \mathbf{x}^\top)\}$  (regression)
- Then
  - a)  $L_2$ -regularization  $\text{pen}(F) = \lambda \sum_{j=1}^d w_j^2$
  - b)  $L_1$ -regularization  $\text{pen}(F) = \lambda \sum_{j=1}^d |w_j|$
  - c)  $L_0$ -regularization  $\text{pen}(F) = \lambda \sum_{j=1}^d 1_{w_j \neq 0}$

- Regularization penalizes complex models  $F$

$$\widehat{R}_{\text{pen}}(f; S_m) = \widehat{R}(f; S_m) + \text{pen}(F)$$

- Let us consider linear models
  - $F = \{f(\mathbf{x}) = \text{sign}(\mathbf{w} \cdot \mathbf{x}^\top)\}$  (classification) or
  - $F = \{f(\mathbf{x}) = (\mathbf{w} \cdot \mathbf{x}^\top)\}$  (regression)
- Then
  - a)  $L_2$ -regularization  $\text{pen}(F) = \lambda \sum_{j=1}^d w_j^2$
  - b)  $L_1$ -regularization  $\text{pen}(F) = \lambda \sum_{j=1}^d |w_j|$
  - c)  $L_0$ -regularization  $\text{pen}(F) = \lambda \sum_{j=1}^d 1_{w_j \neq 0}$

- We consider linear regression model with a Gaussian i.i.d. noise

$$y_i = \mathbf{w} \cdot \mathbf{x}_i^\top + \varepsilon_i, \varepsilon_i \sim \mathcal{N}(0, \sigma^2)$$

- Thus the distribution of a data point

$$y_i \sim \mathcal{N}(\mathbf{w} \cdot \mathbf{x}_i^\top, \sigma^2)$$

- Data sample  $S_m$  likelihood

$$p(S_m | \mathbf{w}) = \prod_{i=1}^m \frac{1}{\sqrt{2\pi}\sigma} \exp \left\{ -\frac{1}{2\sigma^2} (y_i - \mathbf{w} \cdot \mathbf{x}_i^\top)^2 \right\}$$

- Log-likelihood of  $S_m$  has the form

$$\mathcal{L}(\mathbf{w}) = \log p(S_m | \mathbf{w}) = m \log \frac{1}{\sqrt{2\pi}\sigma} - \frac{1}{2\sigma^2} \sum_{i=1}^m (y_i - \mathbf{w} \cdot \mathbf{x}_i^\top)^2 \rightarrow \max_{\mathbf{w}}$$

- Maximum Likelihood Estimate  $\Leftrightarrow$  Least Squares Estimate

- We consider linear regression model with a Gaussian i.i.d. noise

$$y_i = \mathbf{w} \cdot \mathbf{x}_i^\top + \varepsilon_i, \varepsilon_i \sim \mathcal{N}(0, \sigma^2)$$

- Thus the distribution of a data point

$$y_i \sim \mathcal{N}(\mathbf{w} \cdot \mathbf{x}_i^\top, \sigma^2)$$

- Data sample  $S_m$  likelihood

$$p(S_m | \mathbf{w}) = \prod_{i=1}^m \frac{1}{\sqrt{2\pi}\sigma} \exp \left\{ -\frac{1}{2\sigma^2} (y_i - \mathbf{w} \cdot \mathbf{x}_i^\top)^2 \right\}$$

- Log-likelihood of  $S_m$  has the form

$$\mathcal{L}(\mathbf{w}) = \log p(S_m | \mathbf{w}) = m \log \frac{1}{\sqrt{2\pi}\sigma} - \frac{1}{2\sigma^2} \sum_{i=1}^m (y_i - \mathbf{w} \cdot \mathbf{x}_i^\top)^2 \rightarrow \max_{\mathbf{w}}$$

- Maximum Likelihood Estimate  $\Leftrightarrow$  Least Squares Estimate

- We consider linear regression model with a Gaussian i.i.d. noise

$$y_i = \mathbf{w} \cdot \mathbf{x}_i^\top + \varepsilon_i, \varepsilon_i \sim \mathcal{N}(0, \sigma^2)$$

- Thus the distribution of a data point

$$y_i \sim \mathcal{N}(\mathbf{w} \cdot \mathbf{x}_i^\top, \sigma^2)$$

- Data sample  $S_m$  likelihood

$$p(S_m | \mathbf{w}) = \prod_{i=1}^m \frac{1}{\sqrt{2\pi}\sigma} \exp \left\{ -\frac{1}{2\sigma^2} (y_i - \mathbf{w} \cdot \mathbf{x}_i^\top)^2 \right\}$$

- Log-likelihood of  $S_m$  has the form

$$\mathcal{L}(\mathbf{w}) = \log p(S_m | \mathbf{w}) = m \log \frac{1}{\sqrt{2\pi}\sigma} - \frac{1}{2\sigma^2} \sum_{i=1}^m (y_i - \mathbf{w} \cdot \mathbf{x}_i^\top)^2 \rightarrow \max_{\mathbf{w}}$$

- Maximum Likelihood Estimate  $\Leftrightarrow$  Least Squares Estimate

- We consider linear regression model with a Gaussian i.i.d. noise

$$y_i = \mathbf{w} \cdot \mathbf{x}_i^\top + \varepsilon_i, \varepsilon_i \sim \mathcal{N}(0, \sigma^2)$$

- Thus the distribution of a data point

$$y_i \sim \mathcal{N}(\mathbf{w} \cdot \mathbf{x}_i^\top, \sigma^2)$$

- Data sample  $S_m$  likelihood

$$p(S_m | \mathbf{w}) = \prod_{i=1}^m \frac{1}{\sqrt{2\pi}\sigma} \exp \left\{ -\frac{1}{2\sigma^2} (y_i - \mathbf{w} \cdot \mathbf{x}_i^\top)^2 \right\}$$

- Log-likelihood of  $S_m$  has the form

$$\mathcal{L}(\mathbf{w}) = \log p(S_m | \mathbf{w}) = m \log \frac{1}{\sqrt{2\pi}\sigma} - \frac{1}{2\sigma^2} \sum_{i=1}^m (y_i - \mathbf{w} \cdot \mathbf{x}_i^\top)^2 \rightarrow \max_{\mathbf{w}}$$

- Maximum Likelihood Estimate  $\Leftrightarrow$  Least Squares Estimate

- We consider linear regression model with a Gaussian i.i.d. noise

$$y_i = \mathbf{w} \cdot \mathbf{x}_i^\top + \varepsilon_i, \varepsilon_i \sim \mathcal{N}(0, \sigma^2)$$

- Thus the distribution of a data point

$$y_i \sim \mathcal{N}(\mathbf{w} \cdot \mathbf{x}_i^\top, \sigma^2)$$

- Data sample  $S_m$  likelihood

$$p(S_m | \mathbf{w}) = \prod_{i=1}^m \frac{1}{\sqrt{2\pi}\sigma} \exp \left\{ -\frac{1}{2\sigma^2} (y_i - \mathbf{w} \cdot \mathbf{x}_i^\top)^2 \right\}$$

- Log-likelihood of  $S_m$  has the form

$$\mathcal{L}(\mathbf{w}) = \log p(S_m | \mathbf{w}) = m \log \frac{1}{\sqrt{2\pi}\sigma} - \frac{1}{2\sigma^2} \sum_{i=1}^m (y_i - \mathbf{w} \cdot \mathbf{x}_i^\top)^2 \rightarrow \max_{\mathbf{w}}$$

- Maximum Likelihood Estimate  $\Leftrightarrow$  Least Squares Estimate

- Let us assume that (Bayesian assumption)

$$\mathbf{w} \sim \mathcal{N}(0, \tau^2 \mathbb{I})$$

- Posterior distribution of  $\mathbf{w}$  has the form

$$p(\mathbf{w}|S_m) \propto p(S_m|\mathbf{w})p(\mathbf{w})$$

$$= C \cdot \exp \left\{ -\frac{1}{2\sigma^2} \sum_{i=1}^m (y_i - \mathbf{w} \cdot \mathbf{x}_i^\top)^2 \right\} \exp \left\{ -\frac{\mathbf{w} \cdot \mathbf{w}^\top}{2\tau^2} \right\}$$

- Maximum A Posteriori (MAP) estimate

$$p(\mathbf{w}|S_m) \rightarrow \max_{\mathbf{w}}$$

- Let us assume that (Bayesian assumption)

$$\mathbf{w} \sim \mathcal{N}(0, \tau^2 \mathbb{I})$$

- Posterior distribution of  $\mathbf{w}$  has the form

$$p(\mathbf{w}|S_m) \propto p(S_m|\mathbf{w})p(\mathbf{w})$$

$$= C \cdot \exp \left\{ -\frac{1}{2\sigma^2} \sum_{i=1}^m (y_i - \mathbf{w} \cdot \mathbf{x}_i^\top)^2 \right\} \exp \left\{ -\frac{\mathbf{w} \cdot \mathbf{w}^\top}{2\tau^2} \right\}$$

- Maximum A Posteriori (MAP) estimate

$$p(\mathbf{w}|S_m) \rightarrow \max_{\mathbf{w}}$$

- Let us assume that (Bayesian assumption)

$$\mathbf{w} \sim \mathcal{N}(0, \tau^2 \mathbb{I})$$

- Posterior distribution of  $\mathbf{w}$  has the form

$$p(\mathbf{w}|S_m) \propto p(S_m|\mathbf{w})p(\mathbf{w})$$

$$= C \cdot \exp \left\{ -\frac{1}{2\sigma^2} \sum_{i=1}^m (y_i - \mathbf{w} \cdot \mathbf{x}_i^\top)^2 \right\} \exp \left\{ -\frac{\mathbf{w} \cdot \mathbf{w}^\top}{2\tau^2} \right\}$$

- Maximum A Posteriori (MAP) estimate

$$p(\mathbf{w}|S_m) \rightarrow \max_{\mathbf{w}}$$

- Posterior distribution of  $\mathbf{w}$  has the form

$$p(\mathbf{w}|S_m) \propto C \cdot \exp \left\{ -\frac{1}{2\sigma^2} \sum_{i=1}^m (y_i - \mathbf{w} \cdot \mathbf{x}_i^\top)^2 \right\} \exp \left\{ -\frac{\mathbf{w} \cdot \mathbf{w}^\top}{2\tau^2} \right\}$$

- Log-posterior

$$\mathcal{L}_{MAP}(\mathbf{w}|S_m) = \log p(\mathbf{w}|S_m)$$

$$= -\frac{1}{2\sigma^2} \sum_{i=1}^m (y_i - \mathbf{w} \cdot \mathbf{x}_i^\top)^2 - \frac{1}{2\tau^2} \sum_{k=1}^d w_k^2 + \text{const}$$

$$= -\frac{m}{2\sigma^2} \left( \frac{1}{m} \sum_{i=1}^m (y_i - \mathbf{w} \cdot \mathbf{x}_i^\top)^2 + \frac{\sigma^2}{m\tau^2} \sum_{k=1}^d w_k^2 \right) + \text{const}$$

$$= -\frac{m}{2\sigma^2} \left( \hat{R}(f; S_m) + \lambda \|\mathbf{w}\|^2 \right) + \text{const} \rightarrow \max_{\mathbf{w}}$$

$$\text{where } \lambda = \frac{\sigma^2}{m\tau^2}$$

- Thus MAP estimate  $\Leftrightarrow L_2$ -regularized Least Squares Estimate

- Posterior distribution of  $\mathbf{w}$  has the form

$$p(\mathbf{w}|S_m) \propto C \cdot \exp \left\{ -\frac{1}{2\sigma^2} \sum_{i=1}^m (y_i - \mathbf{w} \cdot \mathbf{x}_i^\top)^2 \right\} \exp \left\{ -\frac{\mathbf{w} \cdot \mathbf{w}^\top}{2\tau^2} \right\}$$

- Log-posterior

$$\mathcal{L}_{MAP}(\mathbf{w}|S_m) = \log p(\mathbf{w}|S_m)$$

$$= -\frac{1}{2\sigma^2} \sum_{i=1}^m (y_i - \mathbf{w} \cdot \mathbf{x}_i^\top)^2 - \frac{1}{2\tau^2} \sum_{k=1}^d w_k^2 + \text{const}$$

$$= -\frac{m}{2\sigma^2} \left( \frac{1}{m} \sum_{i=1}^m (y_i - \mathbf{w} \cdot \mathbf{x}_i^\top)^2 + \frac{\sigma^2}{m\tau^2} \sum_{k=1}^d w_k^2 \right) + \text{const}$$

$$= -\frac{m}{2\sigma^2} \left( \hat{R}(f; S_m) + \lambda \|\mathbf{w}\|^2 \right) + \text{const} \rightarrow \max_{\mathbf{w}}$$

$$\text{where } \lambda = \frac{\sigma^2}{m\tau^2}$$

- Thus MAP estimate  $\Leftrightarrow L_2$ -regularized Least Squares Estimate

- Posterior distribution of  $\mathbf{w}$  has the form

$$p(\mathbf{w}|S_m) \propto C \cdot \exp \left\{ -\frac{1}{2\sigma^2} \sum_{i=1}^m (y_i - \mathbf{w} \cdot \mathbf{x}_i^\top)^2 \right\} \exp \left\{ -\frac{\mathbf{w} \cdot \mathbf{w}^\top}{2\tau^2} \right\}$$

- Log-posterior

$$\mathcal{L}_{MAP}(\mathbf{w}|S_m) = \log p(\mathbf{w}|S_m)$$

$$= -\frac{1}{2\sigma^2} \sum_{i=1}^m (y_i - \mathbf{w} \cdot \mathbf{x}_i^\top)^2 - \frac{1}{2\tau^2} \sum_{k=1}^d w_k^2 + \text{const}$$

$$= -\frac{m}{2\sigma^2} \left( \frac{1}{m} \sum_{i=1}^m (y_i - \mathbf{w} \cdot \mathbf{x}_i^\top)^2 + \frac{\sigma^2}{m\tau^2} \sum_{k=1}^d w_k^2 \right) + \text{const}$$

$$= -\frac{m}{2\sigma^2} \left( \hat{R}(f; S_m) + \lambda \|\mathbf{w}\|^2 \right) + \text{const} \rightarrow \max_{\mathbf{w}}$$

$$\text{where } \lambda = \frac{\sigma^2}{m\tau^2}$$

- Thus MAP estimate  $\Leftrightarrow L_2$ -regularized Least Squares Estimate

- Posterior distribution of  $\mathbf{w}$  has the form

$$p(\mathbf{w}|S_m) \propto C \cdot \exp \left\{ -\frac{1}{2\sigma^2} \sum_{i=1}^m (y_i - \mathbf{w} \cdot \mathbf{x}_i^\top)^2 \right\} \exp \left\{ -\frac{\mathbf{w} \cdot \mathbf{w}^\top}{2\tau^2} \right\}$$

- Log-posterior

$$\mathcal{L}_{MAP}(\mathbf{w}|S_m) = \log p(\mathbf{w}|S_m)$$

$$= -\frac{1}{2\sigma^2} \sum_{i=1}^m (y_i - \mathbf{w} \cdot \mathbf{x}_i^\top)^2 - \frac{1}{2\tau^2} \sum_{k=1}^d w_k^2 + \text{const}$$

$$= -\frac{m}{2\sigma^2} \left( \frac{1}{m} \sum_{i=1}^m (y_i - \mathbf{w} \cdot \mathbf{x}_i^\top)^2 + \frac{\sigma^2}{m\tau^2} \sum_{k=1}^d w_k^2 \right) + \text{const}$$

$$= -\frac{m}{2\sigma^2} \left( \hat{R}(f; S_m) + \lambda \|\mathbf{w}\|^2 \right) + \text{const} \rightarrow \max_{\mathbf{w}}$$

$$\text{where } \lambda = \frac{\sigma^2}{m\tau^2}$$

- Thus MAP estimate  $\Leftrightarrow L_2$ -regularized Least Squares Estimate

- Posterior distribution of  $\mathbf{w}$  has the form

$$p(\mathbf{w}|S_m) \propto C \cdot \exp \left\{ -\frac{1}{2\sigma^2} \sum_{i=1}^m (y_i - \mathbf{w} \cdot \mathbf{x}_i^\top)^2 \right\} \exp \left\{ -\frac{\mathbf{w} \cdot \mathbf{w}^\top}{2\tau^2} \right\}$$

- Log-posterior

$$\begin{aligned}\mathcal{L}_{MAP}(\mathbf{w}|S_m) &= \log p(\mathbf{w}|S_m) \\ &= -\frac{1}{2\sigma^2} \sum_{i=1}^m (y_i - \mathbf{w} \cdot \mathbf{x}_i^\top)^2 - \frac{1}{2\tau^2} \sum_{k=1}^d w_k^2 + \text{const} \\ &= -\frac{m}{2\sigma^2} \left( \frac{1}{m} \sum_{i=1}^m (y_i - \mathbf{w} \cdot \mathbf{x}_i^\top)^2 + \frac{\sigma^2}{m\tau^2} \sum_{k=1}^d w_k^2 \right) + \text{const} \\ &= -\frac{m}{2\sigma^2} \left( \hat{R}(f; S_m) + \lambda \|\mathbf{w}\|^2 \right) + \text{const} \rightarrow \max_{\mathbf{w}}, \\ \text{where } \lambda &= \frac{\sigma^2}{m\tau^2}\end{aligned}$$

- Thus MAP estimate  $\Leftrightarrow L_2$ -regularized Least Squares Estimate

- Posterior distribution of  $\mathbf{w}$  has the form

$$p(\mathbf{w}|S_m) \propto C \cdot \exp \left\{ -\frac{1}{2\sigma^2} \sum_{i=1}^m (y_i - \mathbf{w} \cdot \mathbf{x}_i^\top)^2 \right\} \exp \left\{ -\frac{\mathbf{w} \cdot \mathbf{w}^\top}{2\tau^2} \right\}$$

- Log-posterior

$$\begin{aligned}\mathcal{L}_{MAP}(\mathbf{w}|S_m) &= \log p(\mathbf{w}|S_m) \\ &= -\frac{1}{2\sigma^2} \sum_{i=1}^m (y_i - \mathbf{w} \cdot \mathbf{x}_i^\top)^2 - \frac{1}{2\tau^2} \sum_{k=1}^d w_k^2 + \text{const} \\ &= -\frac{m}{2\sigma^2} \left( \frac{1}{m} \sum_{i=1}^m (y_i - \mathbf{w} \cdot \mathbf{x}_i^\top)^2 + \frac{\sigma^2}{m\tau^2} \sum_{k=1}^d w_k^2 \right) + \text{const} \\ &= -\frac{m}{2\sigma^2} \left( \hat{R}(f; S_m) + \lambda \|\mathbf{w}\|^2 \right) + \text{const} \rightarrow \max_{\mathbf{w}},\end{aligned}$$

$$\text{where } \lambda = \frac{\sigma^2}{m\tau^2}$$

- Thus MAP estimate  $\Leftrightarrow L_2$ -regularized Least Squares Estimate

- Posterior distribution of  $\mathbf{w}$  has the form

$$p(\mathbf{w}|S_m) \propto C \cdot \exp \left\{ -\frac{1}{2\sigma^2} \sum_{i=1}^m (y_i - \mathbf{w} \cdot \mathbf{x}_i^\top)^2 \right\} \exp \left\{ -\frac{\mathbf{w} \cdot \mathbf{w}^\top}{2\tau^2} \right\}$$

- Log-posterior

$$\begin{aligned}\mathcal{L}_{MAP}(\mathbf{w}|S_m) &= \log p(\mathbf{w}|S_m) \\ &= -\frac{1}{2\sigma^2} \sum_{i=1}^m (y_i - \mathbf{w} \cdot \mathbf{x}_i^\top)^2 - \frac{1}{2\tau^2} \sum_{k=1}^d w_k^2 + \text{const} \\ &= -\frac{m}{2\sigma^2} \left( \frac{1}{m} \sum_{i=1}^m (y_i - \mathbf{w} \cdot \mathbf{x}_i^\top)^2 + \frac{\sigma^2}{m\tau^2} \sum_{k=1}^d w_k^2 \right) + \text{const} \\ &= -\frac{m}{2\sigma^2} \left( \hat{R}(f; S_m) + \lambda \|\mathbf{w}\|^2 \right) + \text{const} \rightarrow \max_{\mathbf{w}},\end{aligned}$$

$$\text{where } \lambda = \frac{\sigma^2}{m\tau^2}$$

- Thus MAP estimate  $\Leftrightarrow L_2$ -regularized Least Squares Estimate

- Posterior distribution of  $\mathbf{w}$  has the form

$$p(\mathbf{w}|S_m) \propto C \cdot \exp \left\{ -\frac{1}{2\sigma^2} \sum_{i=1}^m (y_i - \mathbf{w} \cdot \mathbf{x}_i^\top)^2 \right\} \exp \left\{ -\frac{\mathbf{w} \cdot \mathbf{w}^\top}{2\tau^2} \right\}$$

- Log-posterior

$$\begin{aligned}\mathcal{L}_{MAP}(\mathbf{w}|S_m) &= \log p(\mathbf{w}|S_m) \\ &= -\frac{1}{2\sigma^2} \sum_{i=1}^m (y_i - \mathbf{w} \cdot \mathbf{x}_i^\top)^2 - \frac{1}{2\tau^2} \sum_{k=1}^d w_k^2 + \text{const} \\ &= -\frac{m}{2\sigma^2} \left( \frac{1}{m} \sum_{i=1}^m (y_i - \mathbf{w} \cdot \mathbf{x}_i^\top)^2 + \frac{\sigma^2}{m\tau^2} \sum_{k=1}^d w_k^2 \right) + \text{const} \\ &= -\frac{m}{2\sigma^2} \left( \hat{R}(f; S_m) + \lambda \|\mathbf{w}\|^2 \right) + \text{const} \rightarrow \max_{\mathbf{w}},\end{aligned}$$

$$\text{where } \lambda = \frac{\sigma^2}{m\tau^2}$$

- Thus MAP estimate  $\Leftrightarrow L_2$ -regularized Least Squares Estimate



- Training sample  $S_m = \{(\mathbf{x}_i, y_i)\}_{i=1}^m$ ,  $\mathbf{x}_i \in \mathbb{R}^d$ ,  $y_i \in \mathbb{R}^1$
- $\mathbf{X} = \{\mathbf{x}_i, i = 1, \dots, m\} \in \mathbb{R}^{m \times d}$  is a design matrix
- We consider a linear model with a white Gaussian noise

$$y = f(\mathbf{x}) + \varepsilon = \mathbf{w} \cdot \mathbf{x}^\top + \varepsilon, \varepsilon \sim \mathcal{N}(0, \sigma^2)$$

- Let  $J \subseteq \{1, \dots, d\}$  be a subset of features from  $\mathbf{x}$  we use to construct a linear model
- We denote by
  - $\mathbf{X}_J$  a submatrix of  $\mathbf{X}$  with features from  $J$
  - $\mathbf{w}_J$  coefficients, corresponding to  $\mathbf{X}_J$ ,  $\hat{\mathbf{w}}_J$  are their estimates by the least squares method
  - $\hat{f}_J(\mathbf{x}) = \hat{\mathbf{w}}_J \cdot \mathbf{x}^\top$  — regression function,  $\hat{y}_i(J) = \hat{f}_J(\mathbf{x}_i)$  — prediction

- Training sample  $S_m = \{(\mathbf{x}_i, y_i)\}_{i=1}^m$ ,  $\mathbf{x}_i \in \mathbb{R}^d$ ,  $y_i \in \mathbb{R}^1$
- $\mathbf{X} = \{\mathbf{x}_i, i = 1, \dots, m\} \in \mathbb{R}^{m \times d}$  is a design matrix
- We consider a linear model with a white Gaussian noise

$$y = f(\mathbf{x}) + \varepsilon = \mathbf{w} \cdot \mathbf{x}^\top + \varepsilon, \varepsilon \sim \mathcal{N}(0, \sigma^2)$$

- Let  $J \subseteq \{1, \dots, d\}$  be a subset of features from  $\mathbf{x}$  we use to construct a linear model
- We denote by
  - $\mathbf{X}_J$  a submatrix of  $\mathbf{X}$  with features from  $J$
  - $w_J$  coefficients, corresponding to  $\mathbf{X}_J$ ,  $\hat{w}_J$  are their estimates by the least squares method
  - $\hat{f}_J(\mathbf{x}) = \hat{w}_J \cdot \mathbf{x}^\top$  — regression function,  $\hat{y}_i(J) = \hat{f}_J(\mathbf{x}_i)$  — prediction

- Training sample  $S_m = \{(\mathbf{x}_i, y_i)\}_{i=1}^m$ ,  $\mathbf{x}_i \in \mathbb{R}^d$ ,  $y_i \in \mathbb{R}^1$
- $\mathbf{X} = \{\mathbf{x}_i, i = 1, \dots, m\} \in \mathbb{R}^{m \times d}$  is a design matrix
- We consider a linear model with a white Gaussian noise

$$y = f(\mathbf{x}) + \varepsilon = \mathbf{w} \cdot \mathbf{x}^\top + \varepsilon, \varepsilon \sim \mathcal{N}(0, \sigma^2)$$

- Let  $J \subseteq \{1, \dots, d\}$  be a subset of features from  $\mathbf{x}$  we use to construct a linear model
- We denote by
  - $\mathbf{X}_J$  a submatrix of  $\mathbf{X}$  with features from  $J$
  - $w_J$  coefficients, corresponding to  $\mathbf{X}_J$ ,  $\hat{w}_J$  are their estimates by the least squares method
  - $\hat{f}_J(\mathbf{x}) = \hat{w}_J \cdot \mathbf{x}^\top$  — regression function,  $\hat{y}_i(J) = \hat{f}_J(\mathbf{x}_i)$  — prediction

- Training sample  $S_m = \{(\mathbf{x}_i, y_i)\}_{i=1}^m$ ,  $\mathbf{x}_i \in \mathbb{R}^d$ ,  $y_i \in \mathbb{R}^1$
- $\mathbf{X} = \{\mathbf{x}_i, i = 1, \dots, m\} \in \mathbb{R}^{m \times d}$  is a design matrix
- We consider a linear model with a white Gaussian noise

$$y = f(\mathbf{x}) + \varepsilon = \mathbf{w} \cdot \mathbf{x}^\top + \varepsilon, \varepsilon \sim \mathcal{N}(0, \sigma^2)$$

- Let  $J \subseteq \{1, \dots, d\}$  be a subset of features from  $\mathbf{x}$  we use to construct a linear model
- We denote by
  - $\mathbf{X}_J$  a submatrix of  $\mathbf{X}$  with features from  $J$
  - $\mathbf{w}_J$  coefficients, corresponding to  $\mathbf{X}_J$ ,  $\hat{\mathbf{w}}_J$  are their estimates by the least squares method
  - $\hat{f}_J(\mathbf{x}) = \hat{\mathbf{w}}_J \cdot \mathbf{x}^\top$  — regression function,  $\hat{y}_i(J) = \hat{f}_J(\mathbf{x}_i)$  — prediction

- Training sample  $S_m = \{(\mathbf{x}_i, y_i)\}_{i=1}^m$ ,  $\mathbf{x}_i \in \mathbb{R}^d$ ,  $y_i \in \mathbb{R}^1$
- $\mathbf{X} = \{\mathbf{x}_i, i = 1, \dots, m\} \in \mathbb{R}^{m \times d}$  is a design matrix
- We consider a linear model with a white Gaussian noise

$$y = f(\mathbf{x}) + \varepsilon = \mathbf{w} \cdot \mathbf{x}^\top + \varepsilon, \varepsilon \sim \mathcal{N}(0, \sigma^2)$$

- Let  $J \subseteq \{1, \dots, d\}$  be a subset of features from  $\mathbf{x}$  we use to construct a linear model
- We denote by
  - $\mathbf{X}_J$  a submatrix of  $\mathbf{X}$  with features from  $J$
  - $\mathbf{w}_J$  coefficients, corresponding to  $\mathbf{X}_J$ ,  $\hat{\mathbf{w}}_J$  are their estimates by the least squares method
  - $\hat{f}_J(\mathbf{x}) = \hat{\mathbf{w}}_J \cdot \mathbf{x}_J^\top$  — regression function,  $\hat{y}_i(J) = \hat{f}_J(\mathbf{x}_i)$  — prediction

- Training sample  $S_m = \{(\mathbf{x}_i, y_i)\}_{i=1}^m$ ,  $\mathbf{x}_i \in \mathbb{R}^d$ ,  $y_i \in \mathbb{R}^1$
- $\mathbf{X} = \{\mathbf{x}_i, i = 1, \dots, m\} \in \mathbb{R}^{m \times d}$  is a design matrix
- We consider a linear model with a white Gaussian noise

$$y = f(\mathbf{x}) + \varepsilon = \mathbf{w} \cdot \mathbf{x}^\top + \varepsilon, \varepsilon \sim \mathcal{N}(0, \sigma^2)$$

- Let  $J \subseteq \{1, \dots, d\}$  be a subset of features from  $\mathbf{x}$  we use to construct a linear model
- We denote by
  - $\mathbf{X}_J$  a submatrix of  $\mathbf{X}$  with features from  $J$
  - $\mathbf{w}_J$  coefficients, corresponding to  $\mathbf{X}_J$ ,  $\hat{\mathbf{w}}_J$  are their estimates by the least squares method
  - $\hat{f}_J(\mathbf{x}) = \hat{\mathbf{w}}_J \cdot \mathbf{x}_J^\top$  — regression function,  $\hat{y}_i(J) = \hat{f}_J(\mathbf{x}_i)$  — prediction

- Training sample  $S_m = \{(\mathbf{x}_i, y_i)\}_{i=1}^m$ ,  $\mathbf{x}_i \in \mathbb{R}^d$ ,  $y_i \in \mathbb{R}^1$
- $\mathbf{X} = \{\mathbf{x}_i, i = 1, \dots, m\} \in \mathbb{R}^{m \times d}$  is a design matrix
- We consider a linear model with a white Gaussian noise

$$y = f(\mathbf{x}) + \varepsilon = \mathbf{w} \cdot \mathbf{x}^\top + \varepsilon, \varepsilon \sim \mathcal{N}(0, \sigma^2)$$

- Let  $J \subseteq \{1, \dots, d\}$  be a subset of features from  $\mathbf{x}$  we use to construct a linear model
- We denote by
  - $\mathbf{X}_J$  a submatrix of  $\mathbf{X}$  with features from  $J$
  - $\mathbf{w}_J$  coefficients, corresponding to  $\mathbf{X}_J$ ,  $\hat{\mathbf{w}}_J$  are their estimates by the least squares method
  - $\hat{f}_J(\mathbf{x}) = \hat{\mathbf{w}}_J \cdot \mathbf{x}_J^\top$  — regression function,  $\hat{y}_i(J) = \hat{f}_J(\mathbf{x}_i)$  — prediction

- Training sample  $S_m = \{(\mathbf{x}_i, y_i)\}_{i=1}^m$ ,  $\mathbf{x}_i \in \mathbb{R}^d$ ,  $y_i \in \mathbb{R}^1$
- $\mathbf{X} = \{\mathbf{x}_i, i = 1, \dots, m\} \in \mathbb{R}^{m \times d}$  is a design matrix
- We consider a linear model with a white Gaussian noise

$$y = f(\mathbf{x}) + \varepsilon = \mathbf{w} \cdot \mathbf{x}^\top + \varepsilon, \varepsilon \sim \mathcal{N}(0, \sigma^2)$$

- Let  $J \subseteq \{1, \dots, d\}$  be a subset of features from  $\mathbf{x}$  we use to construct a linear model
- We denote by
  - $\mathbf{X}_J$  a submatrix of  $\mathbf{X}$  with features from  $J$
  - $\mathbf{w}_J$  coefficients, corresponding to  $\mathbf{X}_J$ ,  $\hat{\mathbf{w}}_J$  are their estimates by the least squares method
  - $\hat{f}_J(\mathbf{x}) = \hat{\mathbf{w}}_J \cdot \mathbf{x}_J^\top$  — regression function,  $\hat{y}_i(J) = \hat{f}_J(\mathbf{x}_i)$  — prediction

- Risk estimate on the training set is equal to

$$\hat{R}_{\text{tr}}(J) = \frac{1}{m} \sum_{i=1}^m (\hat{y}_i(J) - y_i)^2$$

- Select subset of features  $J$  by minimizing  $\hat{R}_{\text{tr}}(J)$  w.r.t.  $J$ ?
- **No!**  $\hat{R}_{\text{tr}}(J)$  is a biased risk estimate:  $\hat{y}_i(J)$  and  $y_i$  are **correlated**!
- In fact,  $\hat{y}_i(J)$  are correlated with  $y_i$  due to the same noise values  $\varepsilon_i$

$$y_i = \mathbf{w} \cdot \mathbf{x}_i^\top + \varepsilon_i$$

$$\hat{\mathbf{w}}_J = (\mathbf{X}_J^\top \mathbf{X}_J)^{-1} \mathbf{X}_J^\top \mathbf{Y}$$

$$\hat{y}_i(J) = \hat{\mathbf{w}}_J \cdot \mathbf{x}_{i,J}^\top$$

- Risk estimate on the training set is equal to

$$\hat{R}_{\text{tr}}(J) = \frac{1}{m} \sum_{i=1}^m (\hat{y}_i(J) - y_i)^2$$

- Select subset of features  $J$  by minimizing  $\hat{R}_{\text{tr}}(J)$  w.r.t.  $J$ ?
- No!  $\hat{R}_{\text{tr}}(J)$  is a biased risk estimate:  $\hat{y}_i(J)$  and  $y_i$  are correlated!
- In fact,  $\hat{y}_i(J)$  are correlated with  $y_i$  due to the same noise values  $\varepsilon_i$

$$\begin{aligned}y_i &= \mathbf{w} \cdot \mathbf{x}_i^\top + \varepsilon_i \\ \hat{\mathbf{w}}_J &= (\mathbf{X}_J^\top \mathbf{X}_J)^{-1} \mathbf{X}_J^\top \mathbf{Y} \\ \hat{y}_i(J) &= \hat{\mathbf{w}}_J \cdot \mathbf{x}_{i,J}^\top\end{aligned}$$

- Risk estimate on the training set is equal to

$$\hat{R}_{\text{tr}}(J) = \frac{1}{m} \sum_{i=1}^m (\hat{y}_i(J) - y_i)^2$$

- Select subset of features  $J$  by minimizing  $\hat{R}_{\text{tr}}(J)$  w.r.t.  $J$ ?
- **No!**  $\hat{R}_{\text{tr}}(J)$  is a biased risk estimate:  $\hat{y}_i(J)$  and  $y_i$  are **correlated**!
- In fact,  $\hat{y}_i(J)$  are correlated with  $y_i$  due to the same noise values  $\varepsilon_i$

$$y_i = \mathbf{w} \cdot \mathbf{x}_i^\top + \varepsilon_i$$

$$\hat{\mathbf{w}}_J = (\mathbf{X}_J^\top \mathbf{X}_J)^{-1} \mathbf{X}_J^\top \mathbf{Y}$$

$$\hat{y}_i(J) = \hat{\mathbf{w}}_J \cdot \mathbf{x}_{i,J}^\top$$

- Risk estimate on the training set is equal to

$$\hat{R}_{\text{tr}}(J) = \frac{1}{m} \sum_{i=1}^m (\hat{y}_i(J) - y_i)^2$$

- Select subset of features  $J$  by minimizing  $\hat{R}_{\text{tr}}(J)$  w.r.t.  $J$ ?
- **No!**  $\hat{R}_{\text{tr}}(J)$  is a biased risk estimate:  $\hat{y}_i(J)$  and  $y_i$  are **correlated**!
- In fact,  $\hat{y}_i(J)$  are correlated with  $y_i$  due to the same noise values  $\varepsilon_i$

$$y_i = \mathbf{w} \cdot \mathbf{x}_i^\top + \varepsilon_i$$

$$\hat{\mathbf{w}}_J = (\mathbf{X}_J^\top \mathbf{X}_J)^{-1} \mathbf{X}_J^\top \mathbf{Y}$$

$$\hat{y}_i(J) = \hat{\mathbf{w}}_J \cdot \mathbf{x}_{i,J}^\top$$

- Risk estimate on the training set is equal to

$$\hat{R}_{\text{tr}}(J) = \frac{1}{m} \sum_{i=1}^m (\hat{y}_i(J) - y_i)^2$$

- Select subset of features  $J$  by minimizing  $\hat{R}_{\text{tr}}(J)$  w.r.t.  $J$ ?
- **No!**  $\hat{R}_{\text{tr}}(J)$  is a biased risk estimate:  $\hat{y}_i(J)$  and  $y_i$  are **correlated**!
- In fact,  $\hat{y}_i(J)$  are correlated with  $y_i$  due to the same noise values  $\varepsilon_i$

$$y_i = \mathbf{w} \cdot \mathbf{x}_i^\top + \varepsilon_i$$

$$\hat{\mathbf{w}}_J = (\mathbf{X}_J^\top \mathbf{X}_J)^{-1} \mathbf{X}_J^\top \mathbf{Y}$$

$$\hat{y}_i(J) = \hat{\mathbf{w}}_J \cdot \mathbf{x}_{i,J}^\top$$

- Risk estimate on the training set is equal to

$$\hat{R}_{\text{tr}}(J) = \frac{1}{m} \sum_{i=1}^m (\hat{y}_i(J) - y_i)^2$$

- Select subset of features  $J$  by minimizing  $\hat{R}_{\text{tr}}(J)$  w.r.t.  $J$ ?
- **No!**  $\hat{R}_{\text{tr}}(J)$  is a biased risk estimate:  $\hat{y}_i(J)$  and  $y_i$  are **correlated**!
- In fact,  $\hat{y}_i(J)$  are correlated with  $y_i$  due to the same noise values  $\varepsilon_i$

$$y_i = \mathbf{w} \cdot \mathbf{x}_i^\top + \varepsilon_i$$

$$\hat{\mathbf{w}}_J = (\mathbf{X}_J^\top \mathbf{X}_J)^{-1} \mathbf{X}_J^\top \mathbf{Y}$$

$$\hat{y}_i(J) = \hat{\mathbf{w}}_J \cdot \mathbf{x}_{i,J}^\top$$

- Risk estimate on the training set is equal to

$$\hat{R}_{\text{tr}}(J) = \frac{1}{m} \sum_{i=1}^m (\hat{y}_i(J) - y_i)^2$$

- Select subset of features  $J$  by minimizing  $\hat{R}_{\text{tr}}(J)$  w.r.t.  $J$ ?
- **No!**  $\hat{R}_{\text{tr}}(J)$  is a biased risk estimate:  $\hat{y}_i(J)$  and  $y_i$  are **correlated**!
- In fact,  $\hat{y}_i(J)$  are correlated with  $y_i$  due to the same noise values  $\varepsilon_i$

$$y_i = \mathbf{w} \cdot \mathbf{x}_i^\top + \varepsilon_i$$

$$\hat{\mathbf{w}}_J = (\mathbf{X}_J^\top \mathbf{X}_J)^{-1} \mathbf{X}_J^\top \mathbf{Y}$$

$$\hat{y}_i(J) = \hat{\mathbf{w}}_J \cdot \mathbf{x}_{i,J}^\top$$

- Let us denote by  $y_i^*$  newly obtained output measurements (with newly generated noise values  $\varepsilon_i^*$ ) for the same  $\mathbf{x}_i$ , i.e.

$$y_i^* = f(\mathbf{x}_i) + \varepsilon_i^* = \mathbf{w} \cdot \mathbf{x}_i^\top + \varepsilon_i^*, \quad \varepsilon_i^* \sim \mathcal{N}(0, \sigma^2)$$

- Real generalization error of a prediction (in-sample error)

$$R(J) = \frac{1}{m} \sum_{i=1}^m \mathbb{E}(\hat{y}_i(J) - y_i^*)^2,$$

- The problem is to greedily select  $J$ , such that  $R(J)$  is small
- How to estimate  $R(J)$ ?

- Let us denote by  $y_i^*$  newly obtained output measurements (with newly generated noise values  $\varepsilon_i^*$ ) for the same  $\mathbf{x}_i$ , i.e.

$$y_i^* = f(\mathbf{x}_i) + \varepsilon_i^* = \mathbf{w} \cdot \mathbf{x}_i^\top + \varepsilon_i^*, \quad \varepsilon_i^* \sim \mathcal{N}(0, \sigma^2)$$

- Real generalization error of a prediction (in-sample error)

$$R(J) = \frac{1}{m} \sum_{i=1}^m \mathbb{E}(\hat{y}_i(J) - y_i^*)^2,$$

- The problem is to greedily select  $J$ , such that  $R(J)$  is small
- How to estimate  $R(J)$ ?

- Let us denote by  $y_i^*$  newly obtained output measurements (with newly generated noise values  $\varepsilon_i^*$ ) for the same  $\mathbf{x}_i$ , i.e.

$$y_i^* = f(\mathbf{x}_i) + \varepsilon_i^* = \mathbf{w} \cdot \mathbf{x}_i^\top + \varepsilon_i^*, \quad \varepsilon_i^* \sim \mathcal{N}(0, \sigma^2)$$

- Real generalization error of a prediction (in-sample error)

$$R(J) = \frac{1}{m} \sum_{i=1}^m \mathbb{E}(\hat{y}_i(J) - y_i^*)^2,$$

- The problem is to greedily select  $J$ , such that  $R(J)$  is small
- How to estimate  $R(J)$ ?

- Let us denote by  $y_i^*$  newly obtained output measurements (with newly generated noise values  $\varepsilon_i^*$ ) for the same  $\mathbf{x}_i$ , i.e.

$$y_i^* = f(\mathbf{x}_i) + \varepsilon_i^* = \mathbf{w} \cdot \mathbf{x}_i^\top + \varepsilon_i^*, \quad \varepsilon_i^* \sim \mathcal{N}(0, \sigma^2)$$

- Real generalization error of a prediction (in-sample error)

$$R(J) = \frac{1}{m} \sum_{i=1}^m \mathbb{E}(\hat{y}_i(J) - y_i^*)^2,$$

- The problem is to greedily select  $J$ , such that  $R(J)$  is small
- How to estimate  $R(J)$ ?

- Error on the training sample and in-sample generalization error

$$\widehat{R}_{\text{tr}}(J) = \frac{1}{m} \sum_{i=1}^m (\widehat{y}_i(J) - y_i)^2,$$

$$R(J) = \frac{1}{m} \sum_{i=1}^m \mathbb{E}(\widehat{y}_i(J) - y_i^*)^2$$

- **Theorem:**  $\mathbb{E}(\widehat{R}_{\text{tr}}(J)) < R(J)$  and

$$\text{bias}(\widehat{R}_{\text{tr}}(J)) = \mathbb{E}(\widehat{R}_{\text{tr}}(J)) - R(J) = -\frac{2}{m} \sum_{i=1}^m \text{Cov}(\widehat{y}_i, y_i)$$

- Corrected (unbiased) risk estimate using the training sample

$$R(J) \approx \widehat{R}_{\text{tr}}(J) + \frac{2}{m} \sum_{i=1}^m \text{Cov}(\widehat{y}_i, y_i)$$

- Error on the training sample and in-sample generalization error

$$\widehat{R}_{\text{tr}}(J) = \frac{1}{m} \sum_{i=1}^m (\widehat{y}_i(J) - y_i)^2,$$

$$R(J) = \frac{1}{m} \sum_{i=1}^m \mathbb{E}(\widehat{y}_i(J) - y_i^*)^2$$

- **Theorem:**  $\mathbb{E}(\widehat{R}_{\text{tr}}(J)) < R(J)$  and

$$\text{bias}(\widehat{R}_{\text{tr}}(J)) = \mathbb{E}(\widehat{R}_{\text{tr}}(J)) - R(J) = -\frac{2}{m} \sum_{i=1}^m \text{Cov}(\widehat{y}_i, y_i)$$

- Corrected (unbiased) risk estimate using the training sample

$$R(J) \approx \widehat{R}_{\text{tr}}(J) + \frac{2}{m} \sum_{i=1}^m \text{Cov}(\widehat{y}_i, y_i)$$

- Error on the training sample and in-sample generalization error

$$\widehat{R}_{\text{tr}}(J) = \frac{1}{m} \sum_{i=1}^m (\widehat{y}_i(J) - y_i)^2,$$

$$R(J) = \frac{1}{m} \sum_{i=1}^m \mathbb{E}(\widehat{y}_i(J) - y_i^*)^2$$

- **Theorem:**  $\mathbb{E}(\widehat{R}_{\text{tr}}(J)) < R(J)$  and

$$\text{bias}(\widehat{R}_{\text{tr}}(J)) = \mathbb{E}(\widehat{R}_{\text{tr}}(J)) - R(J) = -\frac{2}{m} \sum_{i=1}^m \text{Cov}(\widehat{y}_i, y_i)$$

- Corrected (unbiased) risk estimate using the training sample

$$R(J) \approx \widehat{R}_{\text{tr}}(J) + \frac{2}{m} \sum_{i=1}^m \text{Cov}(\widehat{y}_i, y_i)$$

- It can be proved, that in the linear case

$$2 \sum_{i=1}^m \text{Cov}(\hat{y}_i, y_i) \sim 2|J|\sigma^2$$

- Thus, we get  $C_p$  Mallow statistics, representing asymptotically unbiased estimate of the regression risk

$$\hat{R}(J) = \hat{R}_{\text{tr}}(J) + \frac{2\sigma^2}{m}|J|$$

The second term here penalizes complexity

- As output noise variance  $\sigma^2$  we can use some estimate  $\hat{\sigma}^2$ , based on residuals on the training set, obtained by fitting the model
- Select a subset of features  $J$ : minimize greedily  $\hat{R}(J)$  w.r.t.  $J$ , i.e.

$$\hat{R}(J) \rightarrow \min_{\mathbf{w}_J, J}$$

- It can be proved, that in the linear case

$$2 \sum_{i=1}^m \text{Cov}(\hat{y}_i, y_i) \sim 2|J|\sigma^2$$

- Thus, we get  $C_p$  Mallow statistics, representing asymptotically unbiased estimate of the regression risk

$$\hat{R}(J) = \hat{R}_{\text{tr}}(J) + \frac{2\sigma^2}{m}|J|$$

The second term here penalizes complexity

- As output noise variance  $\sigma^2$  we can use some estimate  $\hat{\sigma}^2$ , based on residuals on the training set, obtained by fitting the model
- Select a subset of features  $J$ : minimize greedily  $\hat{R}(J)$  w.r.t.  $J$ , i.e.

$$\hat{R}(J) \rightarrow \min_{\mathbf{w}_J, J}$$

- It can be proved, that in the linear case

$$2 \sum_{i=1}^m \text{Cov}(\hat{y}_i, y_i) \sim 2|J|\sigma^2$$

- Thus, we get  $C_p$  Mallow statistics, representing asymptotically unbiased estimate of the regression risk

$$\hat{R}(J) = \hat{R}_{\text{tr}}(J) + \frac{2\sigma^2}{m}|J|$$

The second term here penalizes complexity

- As output noise variance  $\sigma^2$  we can use some estimate  $\hat{\sigma}^2$ , based on residuals on the training set, obtained by fitting the model
- Select a subset of features  $J$ : minimize greedily  $\hat{R}(J)$  w.r.t.  $J$ , i.e.

$$\hat{R}(J) \rightarrow \min_{w_J, J}$$

- It can be proved, that in the linear case

$$2 \sum_{i=1}^m \text{Cov}(\hat{y}_i, y_i) \sim 2|J|\sigma^2$$

- Thus, we get  $C_p$  Mallow statistics, representing asymptotically unbiased estimate of the regression risk

$$\hat{R}(J) = \hat{R}_{\text{tr}}(J) + \frac{2\sigma^2}{m}|J|$$

The second term here penalizes complexity

- As output noise variance  $\sigma^2$  we can use some estimate  $\hat{\sigma}^2$ , based on residuals on the training set, obtained by fitting the model
- Select a subset of features  $J$ : minimize greedily  $\hat{R}(J)$  w.r.t.  $J$ , i.e.

$$\hat{R}(J) \rightarrow \min_{\mathbf{w}_J, J}$$

- AIC (Akaike Information Criterion) provides estimate of the risk in case of more general models. It has the form

$$\mathcal{L}_J - |J| \rightarrow \max_{\mathbf{w}_J, J}$$

where

- $\mathcal{L}_J$  is a model log-likelihood
- $|J|$  is a number of free model parameters

- BIC (Bayesian Information Criterion) is equal to

$$\mathcal{L}_J - |J| \log m \rightarrow \max_{\mathbf{w}_J, J}$$

- AIC (Akaike Information Criterion) provides estimate of the risk in case of more general models. It has the form

$$\mathcal{L}_J - |J| \rightarrow \max_{\mathbf{w}_J, J}$$

where

- $\mathcal{L}_J$  is a model log-likelihood
- $|J|$  is a number of free model parameters

- BIC (Bayesian Information Criterion) is equal to

$$\mathcal{L}_J - |J| \log m \rightarrow \max_{\mathbf{w}_J, J}$$

- Log-likelihood of  $S_m$  under linear regression model with Gaussian noise has the form

$$\mathcal{L}_J(\mathbf{w}) = m \log \frac{1}{\sqrt{2\pi}\sigma} - \underbrace{\frac{1}{2\sigma^2} \sum_{i=1}^m (y_i - \mathbf{w}_J \cdot \mathbf{x}_{i,J}^\top)^2}_{-\frac{m}{2\sigma^2} \hat{R}_{\text{tr}}(J)}$$

- AIC has the form

$$\mathcal{L}_J(\mathbf{w}) - |J| = -\frac{1}{2\sigma^2} \underbrace{\sum_{i=1}^m (y_i - \mathbf{w}_J \cdot \mathbf{x}_{i,J}^\top)^2}_{-\frac{m}{2\sigma^2} \hat{R}_{\text{tr}}(J)} - |J| + \text{const} \rightarrow \max_{\mathbf{w}_J, J}$$

- Thus AIC is equivalent to Mallow  $C_p$  in case of linear regression model with a Gaussian noise, since

$$-\frac{2\sigma^2}{m} (\mathcal{L}_J - |J|) \sim \hat{R}_{\text{tr}}(J) + \frac{2\sigma^2}{m} |J|$$

$$\mathcal{L}_J - |J| \rightarrow \max_{\mathbf{w}_J, J} \Leftrightarrow \hat{R}(J) = \hat{R}_{\text{tr}}(J) + \frac{2\sigma^2}{m} |J| \rightarrow \min_{\mathbf{w}_J, J}$$

- Log-likelihood of  $S_m$  under linear regression model with Gaussian noise has the form

$$\mathcal{L}_J(\mathbf{w}) = m \log \frac{1}{\sqrt{2\pi}\sigma} - \underbrace{\frac{1}{2\sigma^2} \sum_{i=1}^m (y_i - \mathbf{w}_J \cdot \mathbf{x}_{i,J}^\top)^2}_{-\frac{m}{2\sigma^2} \hat{R}_{\text{tr}}(J)}$$

- AIC has the form

$$\mathcal{L}_J(\mathbf{w}) - |J| = -\underbrace{\frac{1}{2\sigma^2} \sum_{i=1}^m (y_i - \mathbf{w}_J \cdot \mathbf{x}_{i,J}^\top)^2}_{-\frac{m}{2\sigma^2} \hat{R}_{\text{tr}}(J)} - |J| + \text{const} \rightarrow \max_{\mathbf{w}_J, J}$$

- Thus AIC is equivalent to Mallow  $C_p$  in case of linear regression model with a Gaussian noise, since

$$-\frac{2\sigma^2}{m} (\mathcal{L}_J - |J|) \sim \hat{R}_{\text{tr}}(J) + \frac{2\sigma^2}{m} |J|$$

$$\mathcal{L}_J - |J| \rightarrow \max_{\mathbf{w}_J, J} \Leftrightarrow \hat{R}(J) = \hat{R}_{\text{tr}}(J) + \frac{2\sigma^2}{m} |J| \rightarrow \min_{\mathbf{w}_J, J}$$

- Log-likelihood of  $S_m$  under linear regression model with Gaussian noise has the form

$$\mathcal{L}_J(\mathbf{w}) = m \log \frac{1}{\sqrt{2\pi}\sigma} - \underbrace{\frac{1}{2\sigma^2} \sum_{i=1}^m (y_i - \mathbf{w}_J \cdot \mathbf{x}_{i,J}^\top)^2}_{-\frac{m}{2\sigma^2} \hat{R}_{\text{tr}}(J)}$$

- AIC has the form

$$\mathcal{L}_J(\mathbf{w}) - |J| = -\frac{1}{2\sigma^2} \underbrace{\sum_{i=1}^m (y_i - \mathbf{w}_J \cdot \mathbf{x}_{i,J}^\top)^2}_{-\frac{m}{2\sigma^2} \hat{R}_{\text{tr}}(J)} - |J| + \text{const} \rightarrow \max_{\mathbf{w}_J, J}$$

- Thus AIC is equivalent to Mallow  $C_p$  in case of linear regression model with a Gaussian noise, since

$$-\frac{2\sigma^2}{m} (\mathcal{L}_J - |J|) \sim \hat{R}_{\text{tr}}(J) + \frac{2\sigma^2}{m} |J|$$

$$\mathcal{L}_J - |J| \rightarrow \max_{\mathbf{w}_J, J} \Leftrightarrow \hat{R}(J) = \hat{R}_{\text{tr}}(J) + \frac{2\sigma^2}{m} |J| \rightarrow \min_{\mathbf{w}_J, J}$$

- Log-likelihood of  $S_m$  under linear regression model with Gaussian noise has the form

$$\mathcal{L}_J(\mathbf{w}) = m \log \frac{1}{\sqrt{2\pi}\sigma} - \underbrace{\frac{1}{2\sigma^2} \sum_{i=1}^m (y_i - \mathbf{w}_J \cdot \mathbf{x}_{i,J}^\top)^2}_{-\frac{m}{2\sigma^2} \hat{R}_{\text{tr}}(J)}$$

- AIC has the form

$$\mathcal{L}_J(\mathbf{w}) - |J| = -\frac{1}{2\sigma^2} \underbrace{\sum_{i=1}^m (y_i - \mathbf{w}_J \cdot \mathbf{x}_{i,J}^\top)^2}_{-\frac{m}{2\sigma^2} \hat{R}_{\text{tr}}(J)} - |J| + \text{const} \rightarrow \max_{\mathbf{w}_J, J}$$

- Thus AIC is equivalent to Mallow  $C_p$  in case of linear regression model with a Gaussian noise, since

$$-\frac{2\sigma^2}{m} (\mathcal{L}_J - |J|) \sim \hat{R}_{\text{tr}}(J) + \frac{2\sigma^2}{m} |J|$$

$$\mathcal{L}_J - |J| \rightarrow \max_{\mathbf{w}_J, J} \Leftrightarrow \hat{R}(J) = \hat{R}_{\text{tr}}(J) + \frac{2\sigma^2}{m} |J| \rightarrow \min_{\mathbf{w}_J, J}$$

- Log-likelihood of  $S_m$  under linear regression model with Gaussian noise has the form

$$\mathcal{L}_J(\mathbf{w}) = m \log \frac{1}{\sqrt{2\pi}\sigma} - \underbrace{\frac{1}{2\sigma^2} \sum_{i=1}^m (y_i - \mathbf{w}_J \cdot \mathbf{x}_{i,J}^\top)^2}_{-\frac{m}{2\sigma^2} \hat{R}_{\text{tr}}(J)}$$

- AIC has the form

$$\mathcal{L}_J(\mathbf{w}) - |J| = -\underbrace{\frac{1}{2\sigma^2} \sum_{i=1}^m (y_i - \mathbf{w}_J \cdot \mathbf{x}_{i,J}^\top)^2}_{-\frac{m}{2\sigma^2} \hat{R}_{\text{tr}}(J)} - |J| + \text{const} \rightarrow \max_{\mathbf{w}_J, J}$$

- Thus AIC is equivalent to Mallow  $C_p$  in case of linear regression model with a Gaussian noise, since

$$-\frac{2\sigma^2}{m} (\mathcal{L}_J - |J|) \sim \hat{R}_{\text{tr}}(J) + \frac{2\sigma^2}{m} |J|$$

$$\mathcal{L}_J - |J| \rightarrow \max_{\mathbf{w}_J, J} \Leftrightarrow \hat{R}(J) = \hat{R}_{\text{tr}}(J) + \frac{2\sigma^2}{m} |J| \rightarrow \min_{\mathbf{w}_J, J}$$

- Another possibility to estimate risk: leave-one-out cross-validation

$$\hat{R}_{CV}(J) = \frac{1}{m} \sum_{i=1}^m (y_i - \hat{y}_{(-i)})^2,$$

where  $\hat{y}_{(-i)}$  is a prediction, obtained by a model, constructed using a sample  $S_m \setminus \{(\mathbf{x}_i, y_i)\}$

- Increase computational efficiency using formula

$$\hat{R}_{CV}(J) = \frac{1}{m} \sum_{i=1}^m \left( \frac{y_i - \hat{y}_i(J)}{1 - U_{ii}(J)} \right)^2$$
$$U(J) = \mathbf{X}_J (\mathbf{X}_J^\top \mathbf{X}_J)^{-1} \mathbf{X}_J^\top$$

- Another possibility to estimate risk: leave-one-out cross-validation

$$\widehat{R}_{CV}(J) = \frac{1}{m} \sum_{i=1}^m (y_i - \widehat{y}_{(-i)})^2,$$

where  $\widehat{y}_{(-i)}$  is a prediction, obtained by a model, constructed using a sample  $S_m \setminus \{(\mathbf{x}_i, y_i)\}$

- Increase computational efficiency using formula

$$\widehat{R}_{CV}(J) = \frac{1}{m} \sum_{i=1}^m \left( \frac{y_i - \widehat{y}_i(J)}{1 - U_{ii}(J)} \right)^2$$

$$U(J) = \mathbf{X}_J (\mathbf{X}_J^\top \mathbf{X}_J)^{-1} \mathbf{X}_J^\top$$

- Another possibility to estimate risk: leave-one-out cross-validation

$$\widehat{R}_{CV}(J) = \frac{1}{m} \sum_{i=1}^m (y_i - \widehat{y}_{(-i)})^2,$$

where  $\widehat{y}_{(-i)}$  is a prediction, obtained by a model, constructed using a sample  $S_m \setminus \{(\mathbf{x}_i, y_i)\}$

- Increase computational efficiency using formula

$$\widehat{R}_{CV}(J) = \frac{1}{m} \sum_{i=1}^m \left( \frac{y_i - \widehat{y}_i(J)}{1 - U_{ii}(J)} \right)^2$$

$$U(J) = \mathbf{X}_J (\mathbf{X}_J^\top \mathbf{X}_J)^{-1} \mathbf{X}_J^\top$$

- Another possibility to estimate risk: leave-one-out cross-validation

$$\widehat{R}_{CV}(J) = \frac{1}{m} \sum_{i=1}^m (y_i - \widehat{y}_{(-i)})^2,$$

where  $\widehat{y}_{(-i)}$  is a prediction, obtained by a model, constructed using a sample  $S_m \setminus \{(\mathbf{x}_i, y_i)\}$

- Increase computational efficiency using formula

$$\widehat{R}_{CV}(J) = \frac{1}{m} \sum_{i=1}^m \left( \frac{y_i - \widehat{y}_i(J)}{1 - U_{ii}(J)} \right)^2$$

$$U(J) = \mathbf{X}_J (\mathbf{X}_J^\top \mathbf{X}_J)^{-1} \mathbf{X}_J^\top$$

- Use one of criteria ( $C_p$  Mallow, AIC, BIC or LOO CV) to estimate generalization error:
  - $C_p$  Mallow, AIC and BIC are more computationally efficient than LOO CV criterion
  - BIC more significantly penalizes models with big number of features and in fact is theoretically consistent opposed to AIC
- Use some greedy selection scheme (backward selection, forward selection, etc.) to select a subset of features, which delivers more efficient value of the considered criterion
- AIC & BIC are special cases of  $L_0$ -regularization

- Use one of criteria ( $C_p$  Mallow, AIC, BIC or LOO CV) to estimate generalization error:
  - $C_p$  Mallow, AIC and BIC are more computationally efficient than LOO CV criterion
  - BIC more significantly penalizes models with big number of features and in fact is theoretically consistent opposed to AIC
- Use some greedy selection scheme (backward selection, forward selection, etc.) to select a subset of features, which delivers more efficient value of the considered criterion
- AIC & BIC are special cases of  $L_0$ -regularization

- Use one of criteria ( $C_p$  Mallow, AIC, BIC or LOO CV) to estimate generalization error:
  - $C_p$  Mallow, AIC and BIC are more computationally efficient than LOO CV criterion
  - BIC more significantly penalizes models with big number of features and in fact is theoretically consistent opposed to AIC
- Use some greedy selection scheme (backward selection, forward selection, etc.) to select a subset of features, which delivers more efficient value of the considered criterion
- AIC & BIC are special cases of  $L_0$ -regularization

## 1 Overfitting. Bias-variance decomposition

## 2 Model Quality Criteria

## 3 Feature Selection

## 4 Regularization

## 5 Linear Regression Model Selection

## 6 Appendix. Materials for Additional Study: Sensitivity Analysis

Simple example: If the friction from brakes, no wind resistance and all other factors remain the same, which will stop first — a **heavier** car or a **lighter** car?



From physics we know that if we consider only first order effects breaking distance is

$$d = \frac{v^2}{2\mu g}.$$

Thus the breaking distance significantly depends on

- the speed of the car  $v$ ;
- the friction coefficient  $\mu$ ;
- the gravitational constant  $g$

and **does not depend on mass**

If we didn't know physics and only have made a number of experiments this answer could be obtained with **sensitivity analysis**

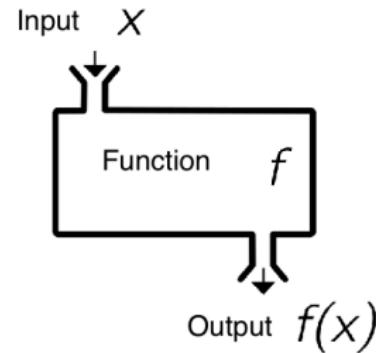
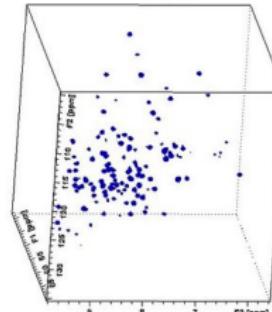
## Sample

It's not possible to get new sample points

## Black Box

*Expensive:* We can generate sample, but can't add points in real time

*Cheap:* We can add as many points as needed in real time



Two general settings are usually considered:

**Local sensitivity:** analyses how a small perturbation influences the value of the output in a *given input point*

**Global sensitivity:** how the input variability influences the variance output *on average over the whole input domain*

Let us consider global sensitivity

Let us consider the following function:

$$f(x_1, x_2) = x_1 + x_2,$$

$$x_1 \in [0, 1]; x_2 \in [0, 2]$$

**Local approach:**

$$\frac{\partial f}{\partial x_1}(x_1^0, x_2^0) = 1 = \frac{\partial f}{\partial x_2}(x_1^0, x_2^0).$$

The model is sensitive w.r.t. variables  $x_1$  and  $x_2$  to the same extent (starting from any input point, small change of any of input coordinates provides the same change in function values)

**Global approach:**

$$\max_{x_1} f(x_1, x_2^0) - \min_{x_1} f(x_1, x_2^0) = 1,$$

$$\max_{x_2} f(x_1^0, x_2) - \min_{x_2} f(x_1^0, x_2) = 2.$$

The model is two times more sensitive to variations of  $x_2$ , then to variations of  $x_1$ .

Let us consider the following function:

$$f(x_1, x_2) = x_1 + x_2,$$

$$x_1 \in [0, 1]; x_2 \in [0, 2]$$

**Local approach:**

$$\frac{\partial f}{\partial x_1}(x_1^0, x_2^0) = 1 = \frac{\partial f}{\partial x_2}(x_1^0, x_2^0).$$

The model is sensitive w.r.t. variables  $x_1$  and  $x_2$  to the same extent  
(starting from any input point, small change of any of input coordinates  
provides the same change in function values)

**Global approach:**

$$\max_{x_1} f(x_1, x_2^0) - \min_{x_1} f(x_1, x_2^0) = 1,$$

$$\max_{x_2} f(x_1^0, x_2) - \min_{x_2} f(x_1^0, x_2) = 2.$$

The model is two times more sensitive to variations of  $x_2$ , then to  
variations of  $x_1$ .

Sensitivity analysis techniques may provide answers to the following questions:

- **Factor prioritization (FP)**

What features affect output the most?

- **Factor fixing (FF)**

What feature values can be fixed without considerable impact on the output values?

- **Variance cutting (VC)**

What features should be fixed to reduce output variance up to the given boundaries?

- **Factor mapping (FM)**

What features are responsible for specific function behavior?

We consider FP and FF

Sensitivity analysis techniques may provide answers to the following questions:

- **Factor prioritization (FP)**

What features affect output the most?

- **Factor fixing (FF)**

What feature values can be fixed without considerable impact on the output values?

- **Variance cutting (VC)**

What features should be fixed to reduce output variance up to the given boundaries?

- **Factor mapping (FM)**

What features are responsible for specific function behavior?

We consider FP and FF

Sensitivity analysis techniques may provide answers to the following questions:

- **Factor prioritization (FP)**

What features affect output the most?

- **Factor fixing (FF)**

What feature values can be fixed without considerable impact on the output values?

- **Variance cutting (VC)**

What features should be fixed to reduce output variance up to the given boundaries?

- **Factor mapping (FM)**

What features are responsible for specific function behavior?

We consider FP and FF

Sensitivity analysis techniques may provide answers to the following questions:

- **Factor prioritization (FP)**

What features affect output the most?

- **Factor fixing (FF)**

What feature values can be fixed without considerable impact on the output values?

- **Variance cutting (VC)**

What features should be fixed to reduce output variance up to the given boundaries?

- **Factor mapping (FM)**

What features are responsible for specific function behavior?

We consider FP and FF

Sensitivity analysis techniques may provide answers to the following questions:

- **Factor prioritization (FP)**

What features affect output the most?

- **Factor fixing (FF)**

What feature values can be fixed without considerable impact on the output values?

- **Variance cutting (VC)**

What features should be fixed to reduce output variance up to the given boundaries?

- **Factor mapping (FM)**

What features are responsible for specific function behavior?

We consider FP and FF

- **Surrogate Model Construction**

- less features means more dense sample, dense sample means more accurate approximation
- approximation techniques work better in smaller dimensions

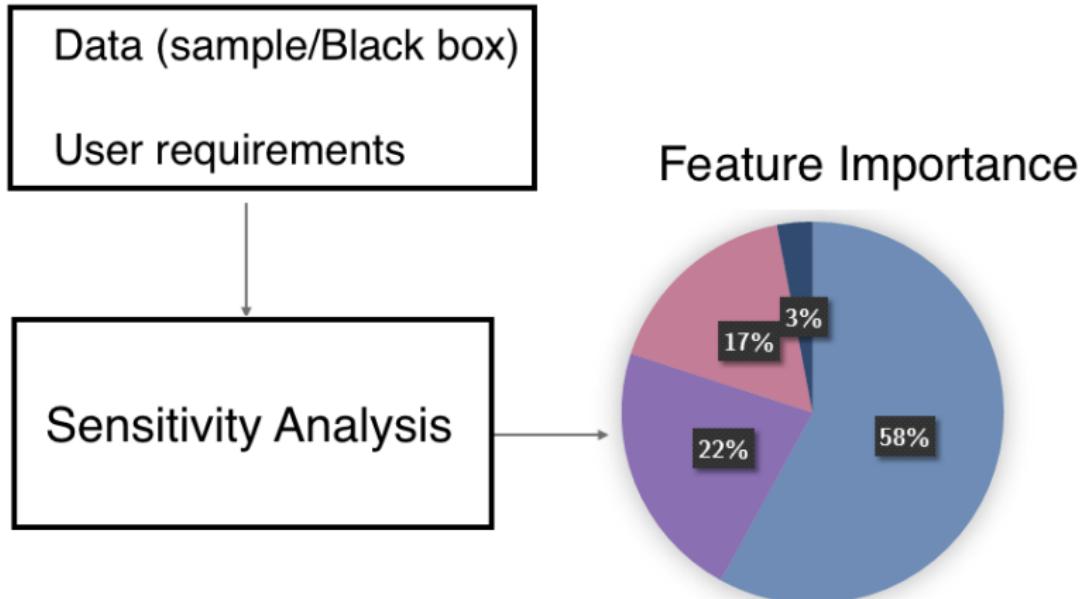
- **Optimization**

- less features allow to do more optimization iterations with the same budget

- **Data Analysis:** one may want to know what features should be:

- measured with the most precision
- dropped in the analysis
- represented with the most details in the DoE

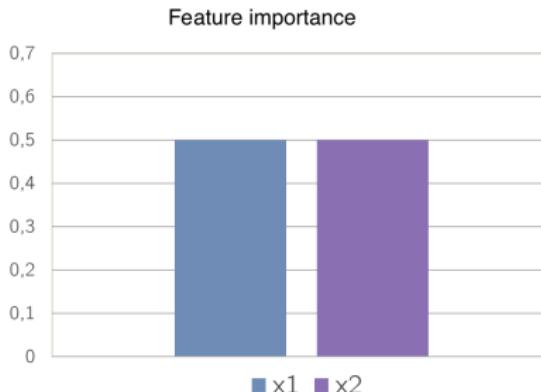
- $\mathbf{X}$  — input data (sample)
- $\mathbf{Y}$  — output data (result)
- $x_j$  — value of the  $j$ -th input feature
- **Budget** — the maximum possible number of calls of oracle to calculate output value



Let us consider the function:

$$f(\mathbf{x}) = x_1 + x_2,$$

$$x_1 \in [-1, 1]; x_2 \in [-1, 1].$$

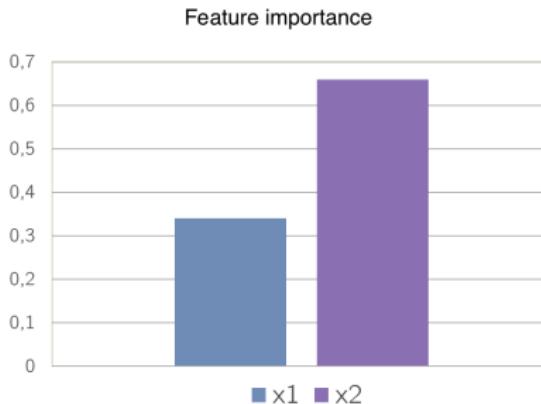


In this case both input variables are important and the result should be the same and equal to 0.5

Let us consider the function:

$$f(\mathbf{x}) = x_1 + 2x_2,$$

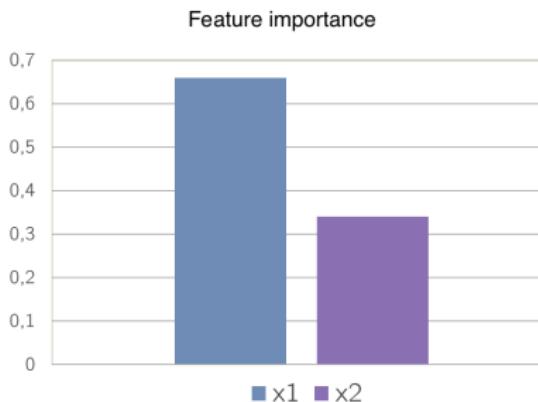
$$x_1 \in [-1, 1]; x_2 \in [-1, 1].$$



We can expect that the contribution of the second variable will increase

The same situation takes place if we change region of variation of one of variables:

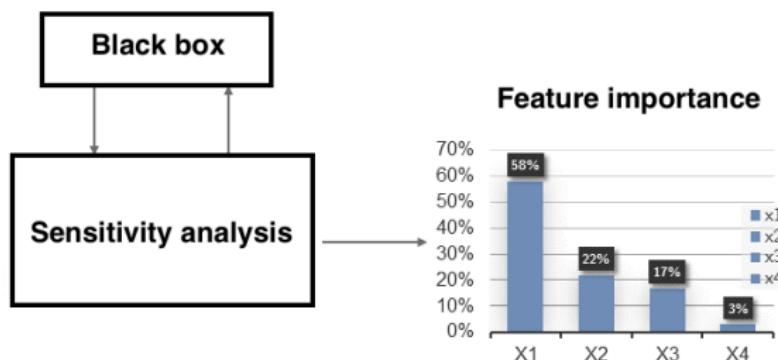
$$f(\mathbf{x}) = x_1 + x_2, \quad x_1 \in [-2, 2]; \quad x_2 \in [-1, 1].$$

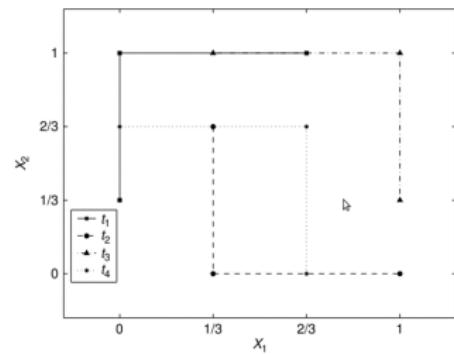
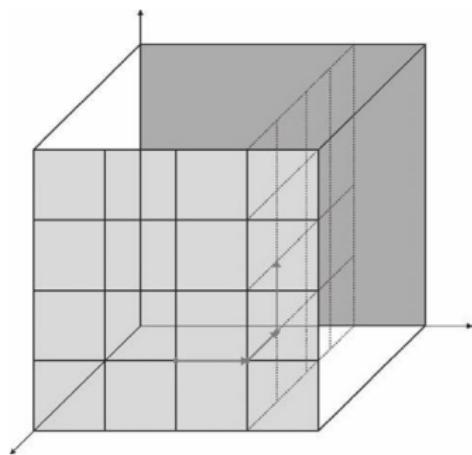


Therefore a global influence of the first variable is bigger (although for all input points considered features are equally important)

## Black-box model is available

- Elementary Effects (Morris Screening)
- Sobol indices
  - FAST
  - CSTA





Each step of the trajectory changes only one component  $x_j$  of input vector  $\mathbf{x}$  and at each step the function change on the trajectory is estimated:

$$d_j(\mathbf{x}) = \frac{y(x_1, \dots, x_{j-1}, x_j + \Delta, x_{j+1}, \dots, x_d) - y(x_1, \dots, x_{j-1}, x_j, x_{j+1}, \dots, x_d)}{\Delta},$$

where  $\Delta$  is a step size

Importance of the  $j$ -th feature is estimated as:

$$W_j = \frac{\mu_j}{\sum_{s=1}^d \mu_s} ,$$

where  $\mu_j = \frac{1}{r} \sum_{s=1}^r |d_j(\mathbf{x}_s)|$ .

Here  $r$  is the number of steps on all trajectories, on which  $x_j$  changes, and  $\mathbf{x}_s$  is input vector on each of these steps

## Pros

- + Can provide reliable estimates even for very small budgets.  
Minimal number of “black box” function calls equals few times number of features which is sufficient to get estimates for not very complex cases
- + Can be efficiently used to detect not-important features in case of not-noisy data

## Cons

- Generates trajectories randomly (can be time-consuming in case of a big number of trajectories)
- Not robust to outliers
- Not very efficient for big budgets

The main idea is to estimate which part of output variance can be explained due to variations in each input feature. We introduce the following indices:

- **The main index.** It estimates influence of each input feature on the output given all other input features are fixed to their average values:

$$S_j = \frac{Var[\mathbb{E}(y|x_j)]}{Var(y)}.$$

- **Total index.** It estimates interplay between features. The index shows which part of output variance will be lost, if a selected input feature is fixed to its average value, and other input features are varied:

$$T_j = 1 - \frac{Var[\mathbb{E}_{x_j}(y|x_1, \dots, x_{j-1}, x_{j+1}, \dots, x_d)]}{Var(y)}.$$

- **Mutual index.** It estimates strength of dependencies between input features in terms of their joint influence on the output:

$$I_j = T_j - S_j$$

The main idea is to estimate which part of output variance can be explained due to variations in each input feature. We introduce the following indices:

- **The main index.** It estimates influence of each input feature on the output given all other input features are fixed to their average values:

$$S_j = \frac{Var[\mathbb{E}(y|x_j)]}{Var(y)}.$$

- **Total index.** It estimates interplay between features. The index shows which part of output variance will be lost, if a selected input feature is fixed to its average value, and other input features are varied:

$$T_j = 1 - \frac{Var[\mathbb{E}_{x_j}(y|x_1, \dots, x_{j-1}, x_{j+1}, \dots, x_d)]}{Var(y)}.$$

- **Mutual index.** It estimates strength of dependencies between input features in terms of their joint influence on the output:

$$I_j = T_j - S_j$$

The main idea is to estimate which part of output variance can be explained due to variations in each input feature. We introduce the following indices:

- **The main index.** It estimates influence of each input feature on the output given all other input features are fixed to their average values:

$$S_j = \frac{Var[\mathbb{E}(y|x_j)]}{Var(y)}.$$

- **Total index.** It estimates interplay between features. The index shows which part of output variance will be lost, if a selected input feature is fixed to its average value, and other input features are varied:

$$T_j = 1 - \frac{Var[\mathbb{E}_{x_j}(y|x_1, \dots, x_{j-1}, x_{j+1}, \dots, x_d)]}{Var(y)}.$$

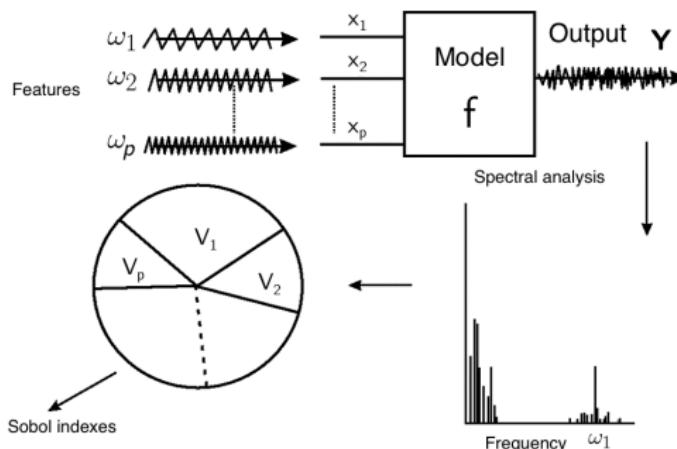
- **Mutual index.** It estimates strength of dependencies between input features in terms of their joint influence on the output:

$$I_j = T_j - S_j$$

Sobol indices can be estimated using the following formula:

$$S_j = \frac{Var\{\mathbb{E}(y|x_j)\}}{Var\{y\}}.$$

Variance estimates can be obtained from Fourier analysis



We sample  $j$ -th input feature w.r.t. one-dimensional space-filling curve:

$$x_j(s) = \frac{1}{2} + \frac{1}{\pi} \arcsin(\sin(\omega_j s + \varphi_j))$$

So we obtain one-dimensional Fourier decomposition of the considered function:

$$y = f(\mathbf{x}(s)) = \sum_{k=-\infty}^{\infty} (A_k \cos(ks) + B_k \sin(ks))$$

$$A_k = \frac{1}{2\pi} \int_{-\infty}^{\infty} f(\mathbf{x}(s)) \cos(ks) ds, \quad B_k = \frac{1}{2\pi} \int_{-\infty}^{\infty} f(\mathbf{x}(s)) \sin(ks) ds$$

Variance estimate is obtained using Fourier coefficients:

$$\text{Var}\{\mathbb{E}(y|x_j)\} = 2 \sum_{k=1}^{M/\omega_j} (A_{k\omega_k}^2 + B_{k\omega_k}^2)$$

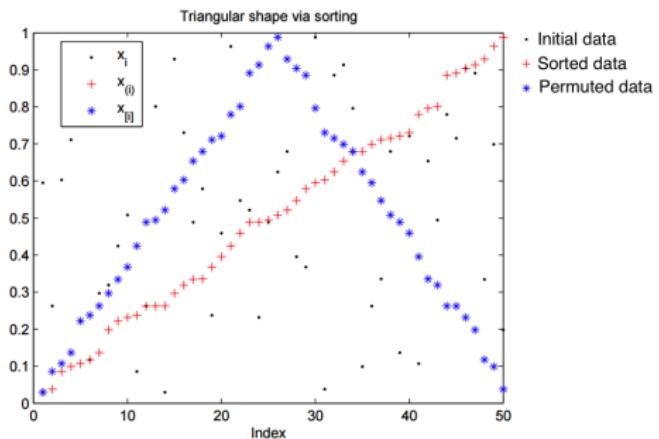
## Pros

- + We need significantly smaller sample size (65 – 100 data points for each feature), then estimate based on simple Monte-Carlo ( $\sim 1000$  data points for each feature).
- + Fourier-based estimate is more robust, than Monte-Carlo
- + Allows to measure to which extent the function is nonlinear

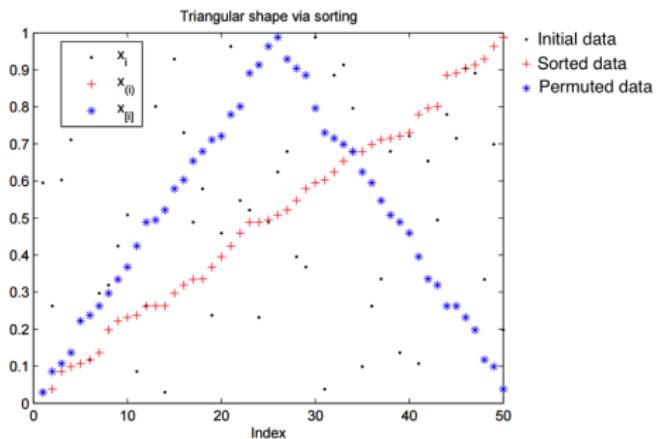
## Cons

- Still the required sample size is too big, i.e. high budgets are required
- Estimate can be biased due to difficulty to provide space-filling property in case of big input dimension

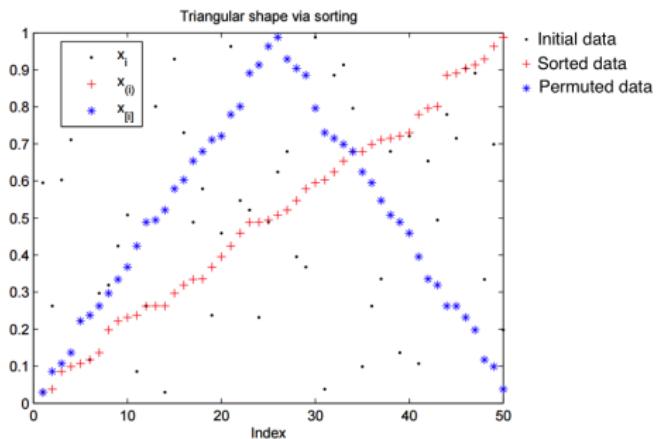
- Based on Fourier analysis (spectral decomposition)
- As opposed to FAST, which generates a sample in some specific way, EASI works with a given sample. It sorts and mixes the sample in order to get almost symmetric periodic signal with a frequency 1
- Output data is sorted according to sorted input data; after sorting spectral density of the output signal is analyzed



- Based on Fourier analysis (spectral decomposition)
- As opposed to FAST, which generates a sample in some specific way, EASI works with a given sample. It sorts and mixes the sample in order to get almost symmetric periodic signal with a frequency 1
- Output data is sorted according to sorted input data; after sorting spectral density of the output signal is analyzed



- Based on Fourier analysis (spectral decomposition)
- As opposed to FAST, which generates a sample in some specific way, EASI works with a given sample. It sorts and mixes the sample in order to get almost symmetric periodic signal with a frequency 1
- Output data is sorted according to sorted input data; after sorting spectral density of the output signal is analyzed



$$\widehat{S}_j = \frac{\sum_{m=1}^M |c_m|^2 + |c_{-m}|^2}{\sum_{m \neq 0} |c_m|^2} = 2 \cdot \frac{\sum_{m=1}^M |c_m|^2}{\sum_{m \neq 0} |c_m|^2},$$

where

$$c_m = \sum_{k=1}^p (\pi(y))_k \zeta_p^{(k-1)m}, \quad \zeta_p = e^{-\frac{2\pi i}{p}}, \quad m = 0, \pm 1, \pm 2, \dots, \pm [\frac{n}{2}].$$

Here  $\pi(y)$  is a permutations, obtained when sorting input data, and applied to the output data

**Pros****Cons**

- + Works fast even in case of big samples
- Only the main Sobol index can be calculated
- Currently can work only with a sample, but not with a black box function

The method is based on the following ideas

- ① For each variable  $x_j$  the main Sobol index  $S_j$  is equal to the mathematical expectation of the correlation coefficient between two samples of function values for inputs, such that values of the  $j$ -th feature are the same, but values for other variables are different
- ② For each input variable  $x_j$  the total Sobol index  $T_j$  is equal to the difference between 1 and the mathematical expectation of the correlation coefficient between two samples of function values for inputs, such that values of the  $j$ -th feature are different, but values for other variables are the same

- ❶ We generate 2 data sets (main and auxiliary), in each of which each input feature has  $K$  values
- ❷ We calculate output function value on  $2K + 2dK$  datasets  
( $2K$  values for the main and auxiliary data sets,  $2dK$  values for data sets, such that values of one variable is takes from one data set, and values for all other variables are taken from other data set)
- ❸ Normalize values:  $g = \frac{f - \mu}{\sqrt{\sigma}}$ .
- ❹ Calculate estimates of Sobol indices:

$$\widehat{S}_j = c_j, \quad \widehat{T}_j = 1 - c_{-j}$$

$$c_j = \frac{1}{2K} \sum_{s=1}^K (g'_{j,s} g_{0,s} + g'_{0,s} g_{j,s}), \quad c_{-j} = \frac{1}{2K} \sum_{s=1}^K (g_{j,s} g_{0,s} + g'_{0,s} g'_{j,s})$$

$g_{j,s}$  is a normalized function value from the  $s$ -th sample, in which value of the  $j$ -th variable is taken from the auxiliary data set, and values for other variables are taken from the main data set

$g'_{j,s}$  is a normalized function value from the  $s$ -th sample, for which the value of  $j$ -th variable is taken from the main data set, and values for other variables are taken from the auxiliary data set

## Pros

- + We can measure degree of nonlinearity
- + The main and the total Sobol indices are equal to zero for the  $i$ -th variable if the output  $\mathbf{Y}$  does not depend on  $x_i$
- + We can easily construct confidence regions

## Cons

- Less accurate than FAST
- In order to get significant result we need a bigger budget

## Example: extract parameters that most affect the level of radioactive isotopes in water

---

- In 1989, the Probabilistic Systems Assessment Group (PSAG) of the OECD Nuclear Energy Agency (NEA) published a report on the simple, artificial Level E model
- Goal is to model a nuclear waste disposal site
- The Level E model is often used as an example for sensitivity analysis methods, see 2004 Saltelli et al. *Sensitivity Analysis in Practice A Guide to Assessing Scientific Models*
- **Problem:** Analyze sensitivities of the outputs (what model parameters affect the dose of radioactive isotopes in water)

## Example: extract parameters that most affect the level of radioactive isotopes in water

---

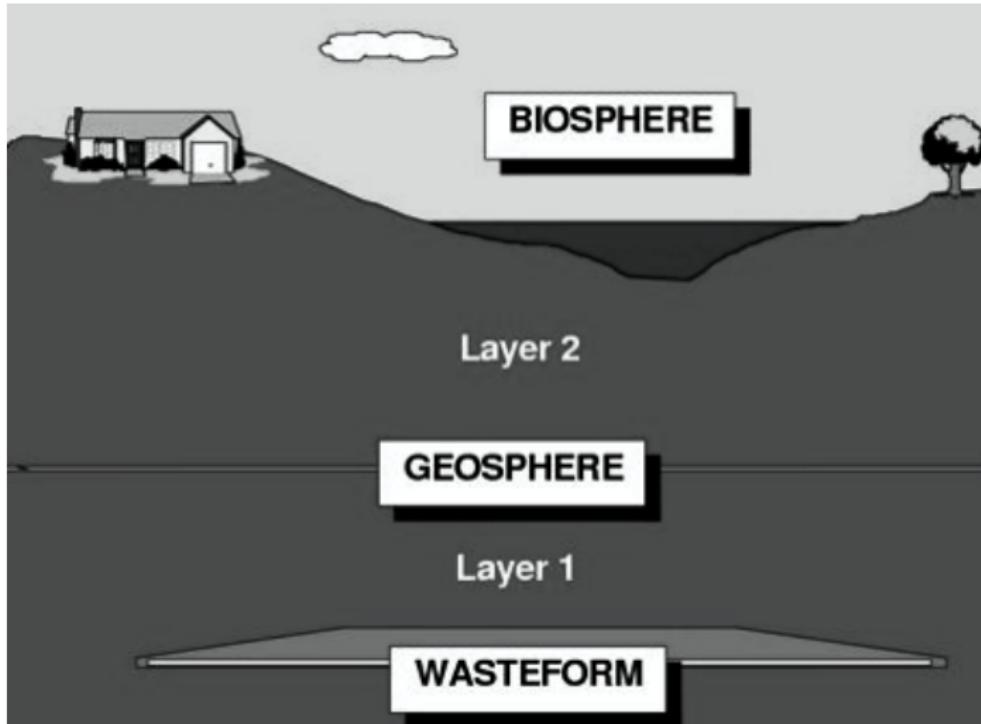
- In 1989, the Probabilistic Systems Assessment Group (PSAG) of the OECD Nuclear Energy Agency (NEA) published a report on the simple, artificial Level E model
- Goal is to model a nuclear waste disposal site
- The Level E model is often used as an example for sensitivity analysis methods, see 2004 Saltelli et al. *Sensitivity Analysis in Practice A Guide to Assessing Scientific Models*
- **Problem:** Analyze sensitivities of the outputs (what model parameters affect the dose of radioactive isotopes in water)

## Example: extract parameters that most affect the level of radioactive isotopes in water

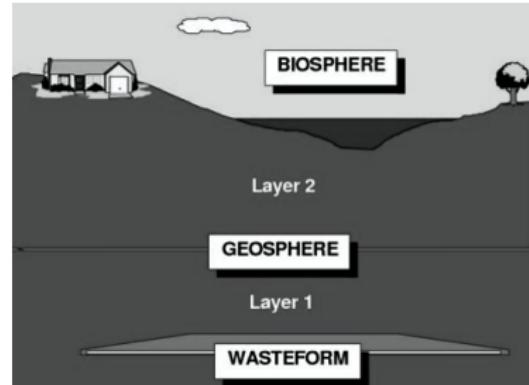
---

- In 1989, the Probabilistic Systems Assessment Group (PSAG) of the OECD Nuclear Energy Agency (NEA) published a report on the simple, artificial Level E model
- Goal is to model a nuclear waste disposal site
- The Level E model is often used as an example for sensitivity analysis methods, see 2004 Saltelli et al. *Sensitivity Analysis in Practice A Guide to Assessing Scientific Models*
- **Problem:** Analyze sensitivities of the outputs (what model parameters affect the dose of radioactive isotopes in water)

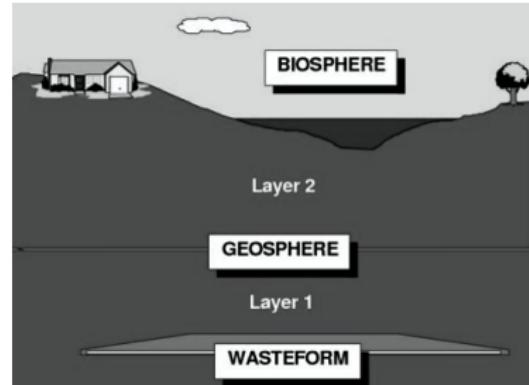
- In 1989, the Probabilistic Systems Assessment Group (PSAG) of the OECD Nuclear Energy Agency (NEA) published a report on the simple, artificial Level E model
- Goal is to model a nuclear waste disposal site
- The Level E model is often used as an example for sensitivity analysis methods, see 2004 Saltelli et al. *Sensitivity Analysis in Practice A Guide to Assessing Scientific Models*
- **Problem:** Analyze sensitivities of the outputs (what model parameters affect the dose of radioactive isotopes in water)



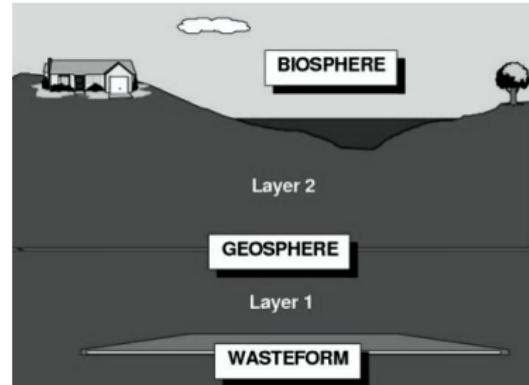
- High-level nuclear waste is disposed of deep underground in a disposal vault
- These radionuclides remain in the wasteform, decaying over time, until the containers fail
- Then groundwater leaches radionuclides out of the wasteform at a constant fractional rate



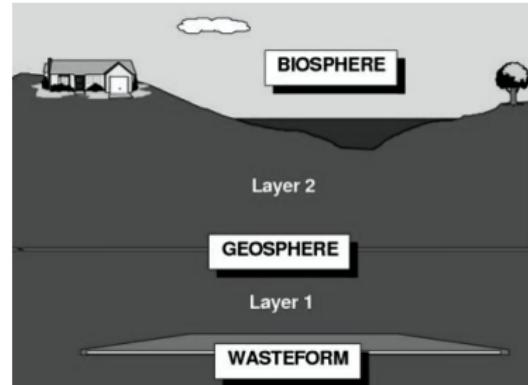
- High-level nuclear waste is disposed of deep underground in a disposal vault
- These radionuclides remain in the wasteform, decaying over time, until the containers fail
- Then groundwater leaches radionuclides out of the wasteform at a constant fractional rate



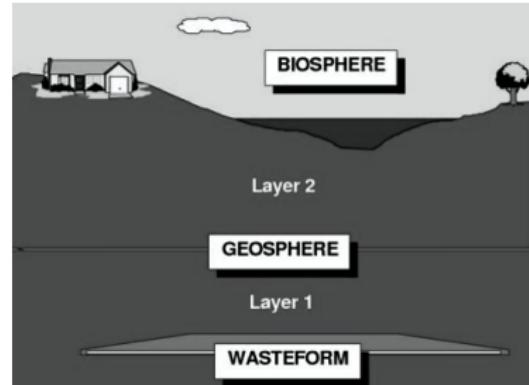
- High-level nuclear waste is disposed of deep underground in a disposal vault
- These radionuclides remain in the wasteform, decaying over time, until the containers fail
- Then groundwater leaches radionuclides out of the wasteform at a constant fractional rate



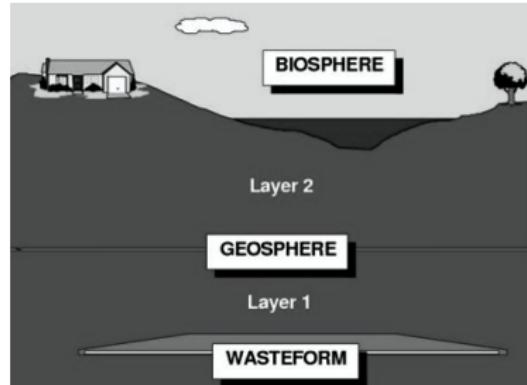
- Radionuclides leaching from the wasteform move slowly from the disposal vault in a long, gently rising path
- The geosphere is divided into two layers, because of two rock strata with different properties
- Radionuclides must penetrate both layers to reach a local stream



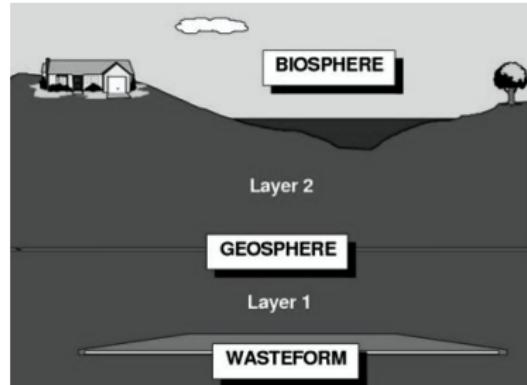
- Radionuclides leaching from the wasteform move slowly from the disposal vault in a long, gently rising path
- The geosphere is divided into two layers, because of two rock strata with different properties
- Radionuclides must penetrate both layers to reach a local stream



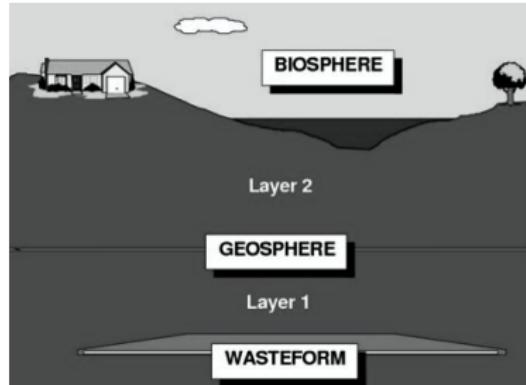
- Radionuclides leaching from the wasteform move slowly from the disposal vault in a long, gently rising path
- The geosphere is divided into two layers, because of two rock strata with different properties
- Radionuclides must penetrate both layers to reach a local stream



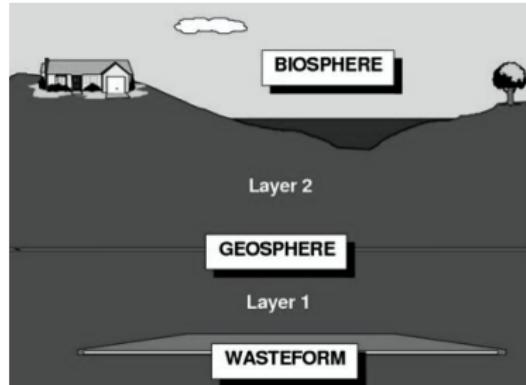
- Radionuclides leaving the geosphere seep into a stream
- The concentration in the stream depends on the amount of flow in the stream
- People living nearby take their drinking water from the stream; the radioactive dose they receive is proportional to the concentration



- Radionuclides leaving the geosphere seep into a stream
- The concentration in the stream depends on the amount of flow in the stream
- People living nearby take their drinking water from the stream; the radioactive dose they receive is proportional to the concentration



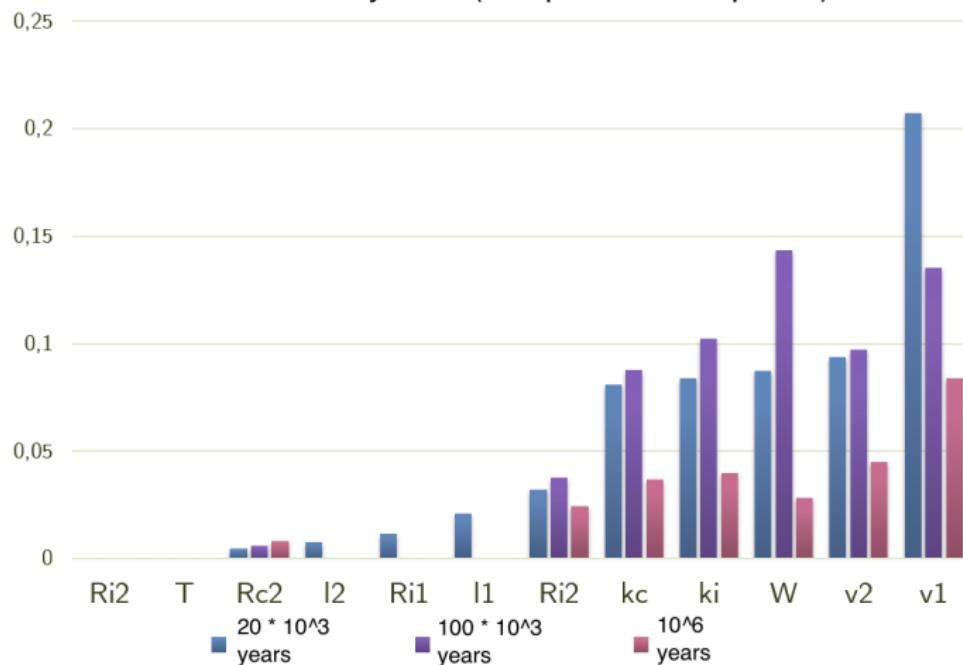
- Radionuclides leaving the geosphere seep into a stream
- The concentration in the stream depends on the amount of flow in the stream
- People living nearby take their drinking water from the stream; the radioactive dose they receive is proportional to the concentration



## Model has 12 input parameters

Notation	Definition	Distribution	Range	Units
$T$	Containment time	Uniform	[100, 1000]	yr
$k_I$	Leach rate for iodine	Log-Uniform	[ $10^{-3}$ , $10^{-2}$ ]	mol/yr
$k_C$	Leach rate for Np chain nuclides	Log-Uniform	[ $10^{-6}$ , $10^{-5}$ ]	mol/yr
$V^{(1)}$	Water velocity in the first geosphere layer	Log-Uniform	[ $10^{-3}$ , $10^{-1}$ ]	m/yr
$l^{(1)}$	Length of the first geosphere layer	Uniform	[100, 500]	m
$R_I^{(1)}$	Retention factor for iodine in the first layer	Uniform	[1, 5]	—
$R_C^{(1)}$	Retention factor for the chain elements in the first layer	Uniform	[3, 30]	—
$v^{(2)}$	Water velocity in the second geosphere layer	Log-Uniform	[ $10^{-2}$ , $10^{-1}$ ]	m/yr
$l^{(2)}$	Length of the second geosphere layer	Uniform	[50, 200]	m
$R_I^{(2)}$	Retention factor for iodine in the second layer	Uniform	[1, 5]	—
$R_C^{(2)}$	Retention factor for the chain elements in the second layer	Uniform	[3, 30]	—
$W$	Stream flow rate	Log-Uniform	[ $10^5$ , $10^7$ ]	$\text{m}^3/\text{yr}$

## Influence of different parameters on radioactivity level (sample with 200 points)



**The most influential factors at 100k years are:**

- Water's velocity of geosphere's first layer ( $V^{(1)}$ )
- Stream flow rate ( $W$ )

**Have no influence at any time:**

- Containment time ( $T$ )