Since 2004
UET
ĐẠI HỌC CÔNG NGHỆ, ĐHQGHN
VNU-University of Engineering and Technology

Since 1906
VNU
ĐẠI HỌC QUỐC GIA HÀ NỘI
Vietnam National University, Hanoi

INT3405 - Machine Learning

Lecture 7: Model Optimization

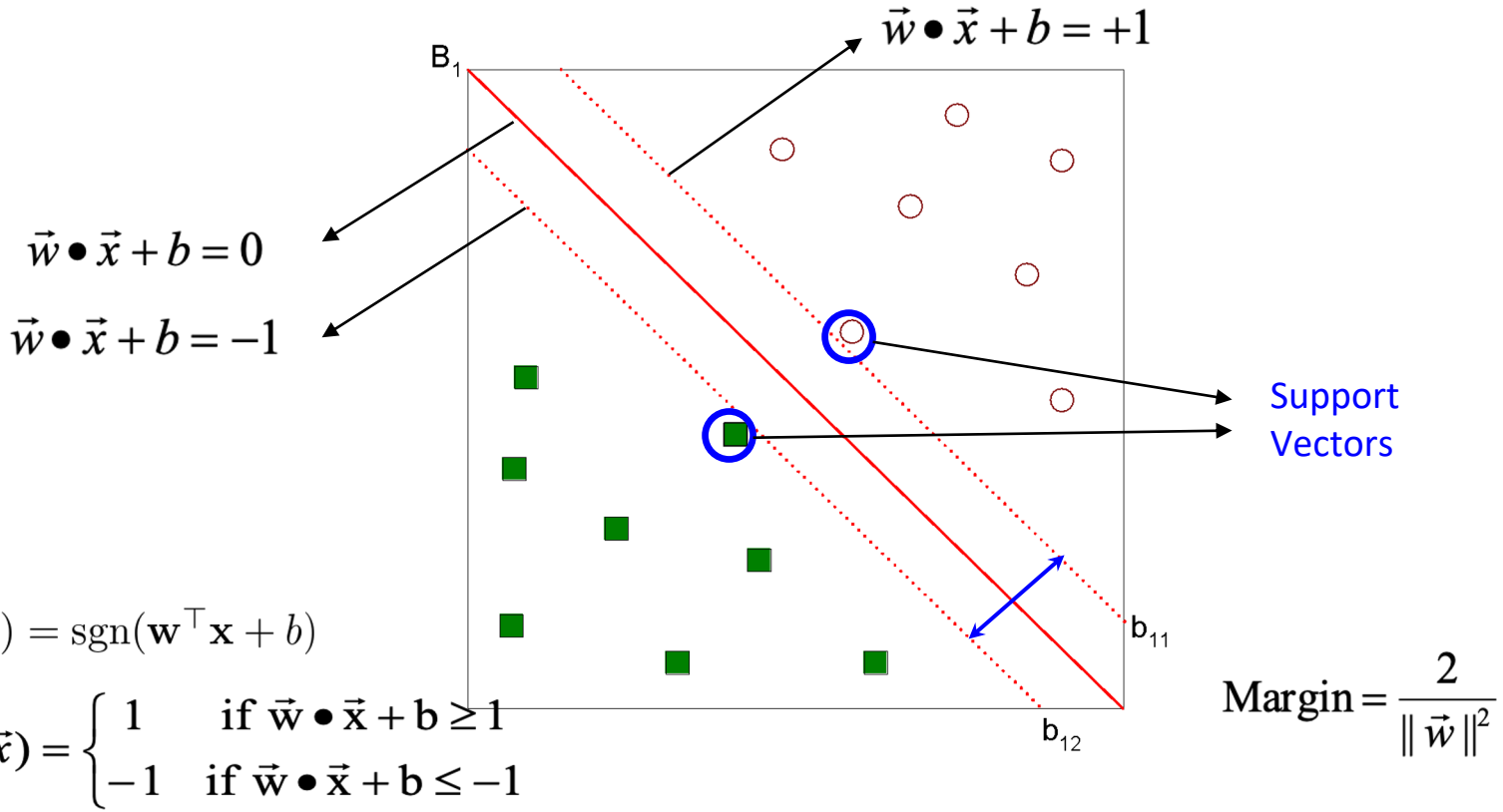
Duc-Trong Le & Viet-Cuong Ta

Hanoi, 10/2023

Outline

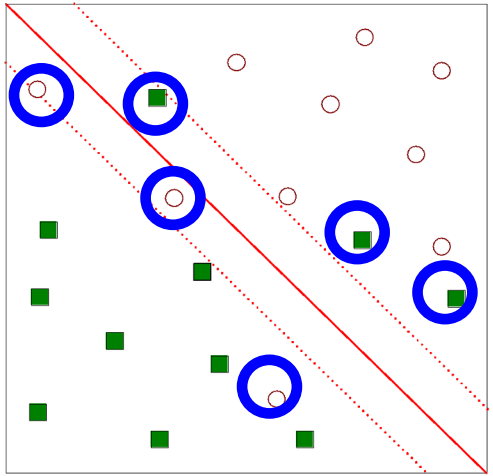
- True Error versus Empirical Error
- Overfitting, Underfitting
- Bias-Variance Tradeoff
- Model Optimization
  - Feature Selection
  - Regularization
  - Model Ensemble

Recap: Support Vector Machines (SVM)



Recap: Soft Margin SVM

- Standard Linear SVM
  - Introduce slack variables
  - Relax the constraints
  - Penalize the relaxation



Primal Problem:

$$\min_{\mathbf{w},b,\xi} \quad \frac{1}{2}\mathbf{w}^\top\mathbf{w} + C\sum_{i=1}^{N}\xi_i$$

$$\text{subject to} \quad y_i(\mathbf{w}^\top\mathbf{x}_i + b) \geq 1 - \xi_i,$$
$$\xi_i \geq 0, i = 1, \ldots, N$$

C is a regularization parameter. Soft margin SVM trade off between maximizing the margin and minimizing the misclassification error rate

True Error versus Empirical Error

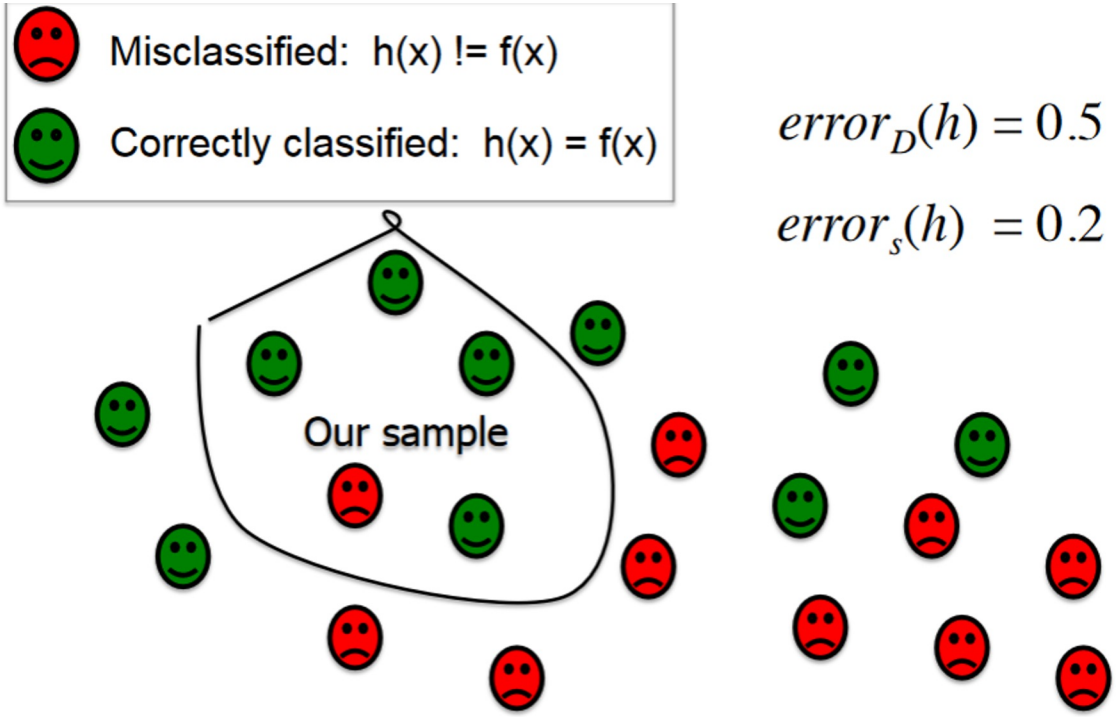
- **True Error/Risk**: target performance measure
  - Classification: probability of misclassification $P(f(X) \neq Y))$
  - Regression: mean squared error $\mathbb{E}[(f(X) - Y)^2]$
  - Performance on a random test/unseen point (X,Y)
- **Empirical Error/risk**: performance on training data
  - Classification: proportion of misclassified examples

$$\frac{1}{n}\sum_{i=1}^{n} \mathbf{1}_{f(X_i)\neq Y_i}$$

  - Regression: average squared error

$$\frac{1}{n}\sum_{i=1}^{n} (f(X_i) - Y_i)^2$$

True Error versus Empirical Error



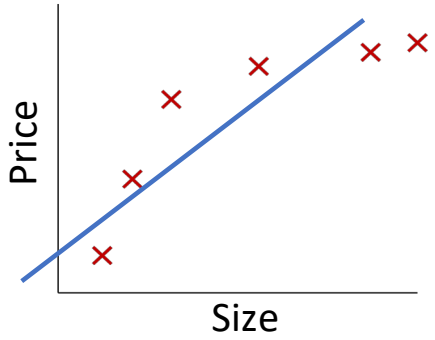
$$error_D(h) = 0.5$$

$$error_s(h) = 0.2$$

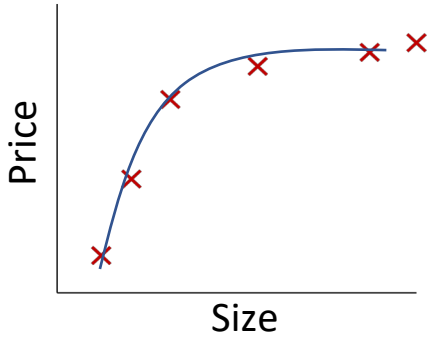
Bryan Pardo, Northwestern University, Machine Learning EECS 349 Fall 2011

Overfitting

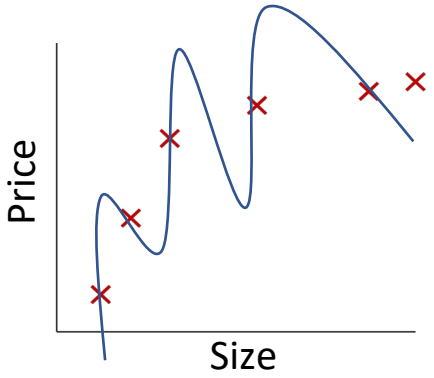
Example: Linear regression (housing prices)



$w_0 + w_1 x$



$w_0 + w_1 x + w_2 x^2$

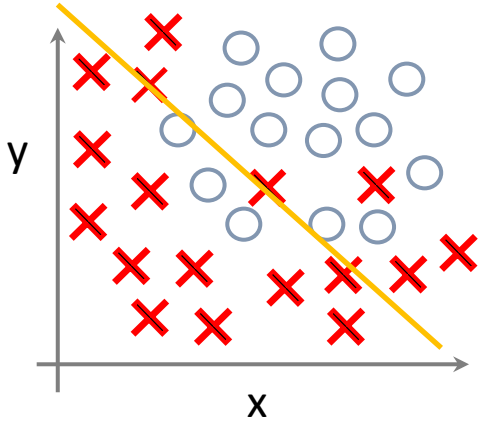


$w_0 + w_1 x + w_2 x^2 + w_3 x^3 + w_4 x^4$

Overfitting: If we have too many features (complicated predictor), the learned hypothesis may fit the training set very well, but fail to generalize to new examples (predict prices on new examples).

Overfitting versus Underfitting

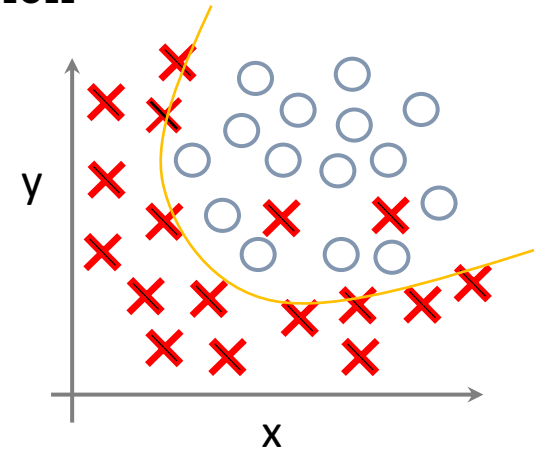
Example: Logistic regression



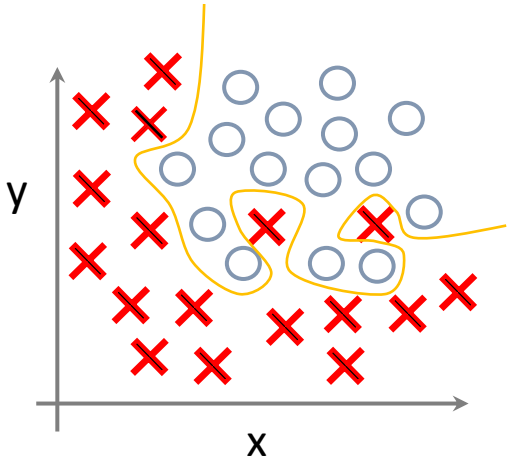
$$h(x) = g(w_0 + w_1 x_1 + w_2 x_2)$$

(g = sigmoid function)

"Underfitting"



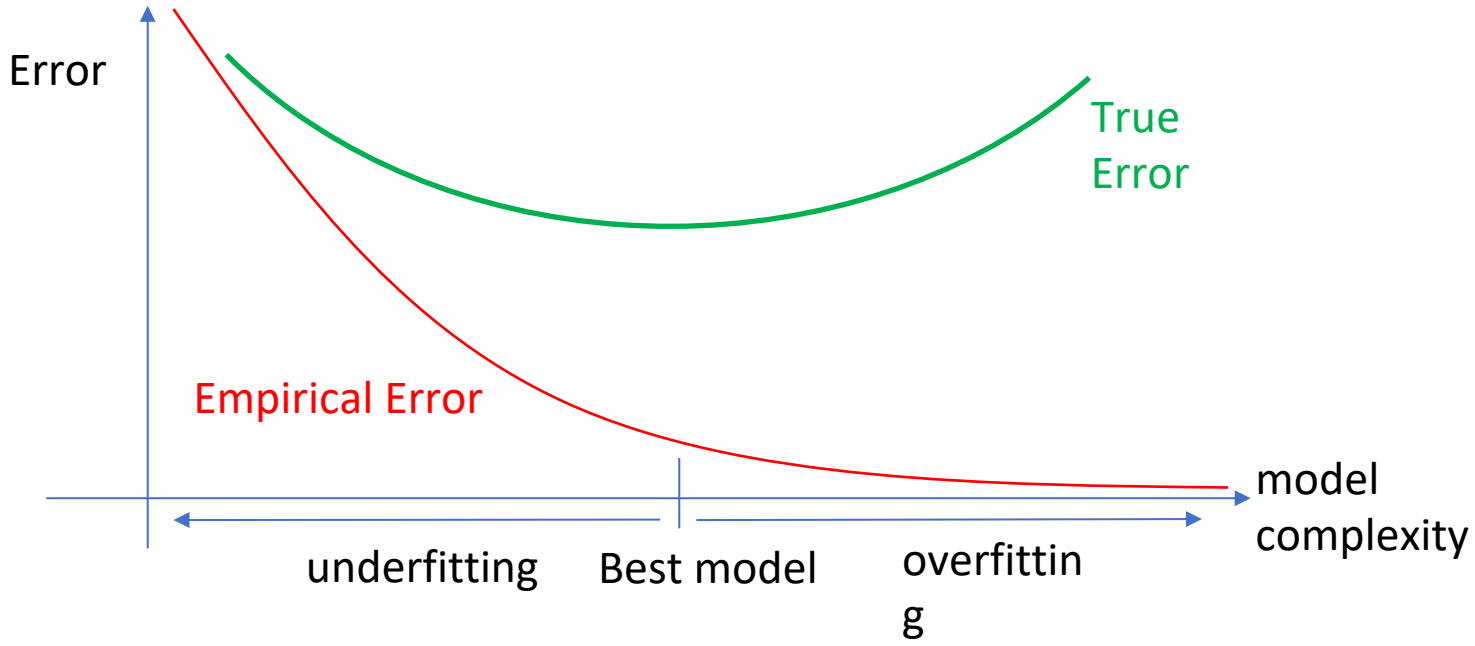
$$g(w_0 + w_1 x_1 + w_2 x_2 + w_3 x_1^2 + w_4 x_2^2 + w_5 x1 x_2)$$



$$g(w_0 + w_1 x_1 + w_2 x_1^2 + w_3 x_1^2 x_2 + w_4 x_1^2 x_2^2 + w_5 x1^2 x_2^3 + w_6 x1^3 x_2 + \ldots)$$

"Overfitting"

Model Complexity



Empirical error (training error) is no longer a good indicator of true error

Examples of Model Complexity

- Examples of Model Spaces with increasing complexity:
  - Regression with polynomials of order k=0,1,2,...

    Higher degree => higher complexity
  - Decision Trees with depth k or with k leaves

    Higher depth/ More # leaves => Higher complexity
  - KNN classifiers with varying neighbourhood sizes k =1,2,3...

    Small neighbourhood => Higher complexity

Risk Analysis (1)

- True Error/Risk vs Empirical Error/Risk

$$R(f) = \mathbb{E}_{XY}[\text{loss}(f(X), Y)] \quad \widehat{R}(f) = \frac{1}{n}\sum_{i=1}^{n} \text{loss}(f(X_i), Y_i)$$

- Optimal Predictor

$$f^* = \arg\min R(f)$$

- Empirical Error Minimization over class $\mathcal{F}$

$$\widehat{f}_n = \arg\min_{f \in \mathcal{F}} \widehat{R}(f)$$

Risk Analysis (2)

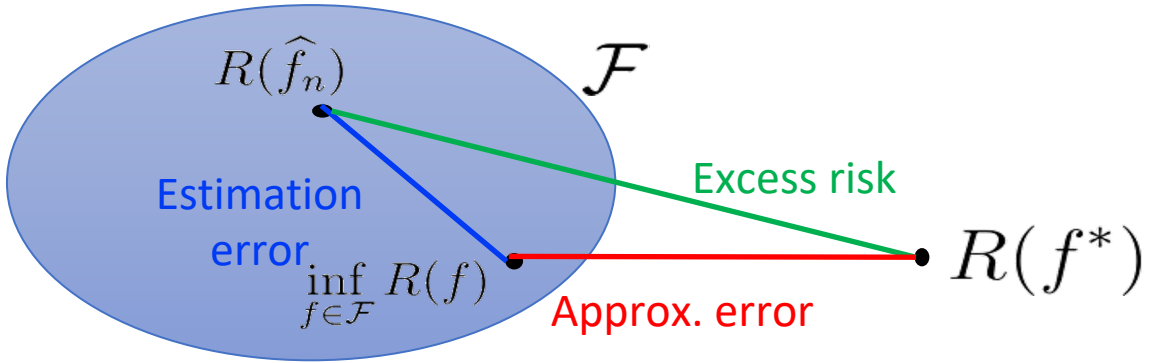
- Excess Risk

$$R(\widehat{f}_n) - R(f^*) = R(\widehat{f}_n) - \inf_{f \in \mathcal{F}} R(f) + \inf_{f \in \mathcal{F}} R(f) - R(f^*)$$

Estimation error — Approximation error

Due to randomness of training data (finite sample size)

Due to restriction of model class

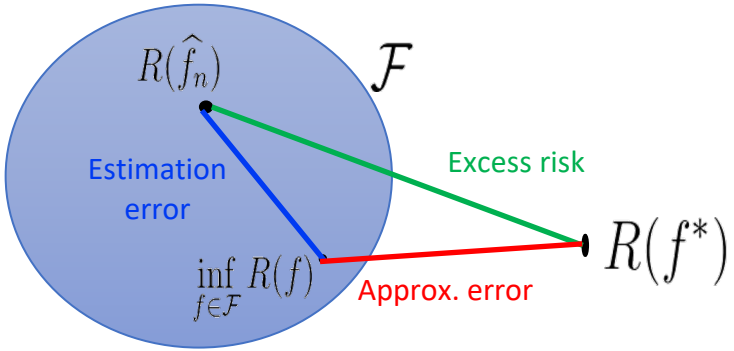


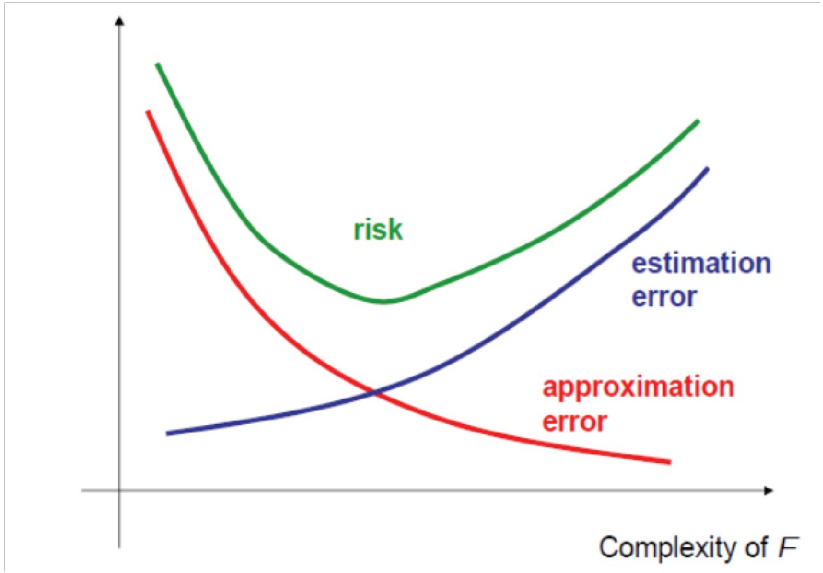
Risk Analysis (2)

$$R(\widehat{f}_n) - R(f^*) = R(\widehat{f}_n) - \inf_{f \in \mathcal{F}} R(f) + \inf_{f \in \mathcal{F}} R(f) - R(f^*)$$

Estimation error

Approximation error





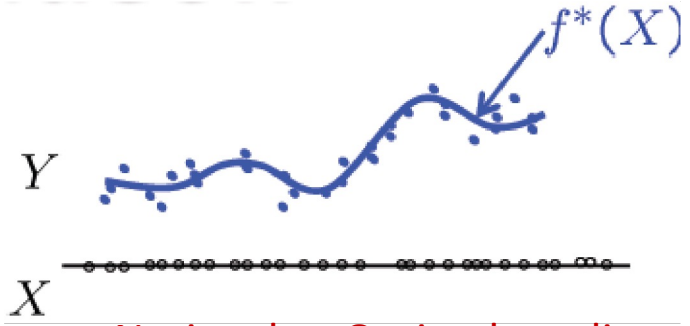
Bias - Variance Trade-off

- Regression:

$$Y = f^*(X) + \epsilon \qquad \epsilon \sim \mathcal{N}(0, \sigma^2)$$

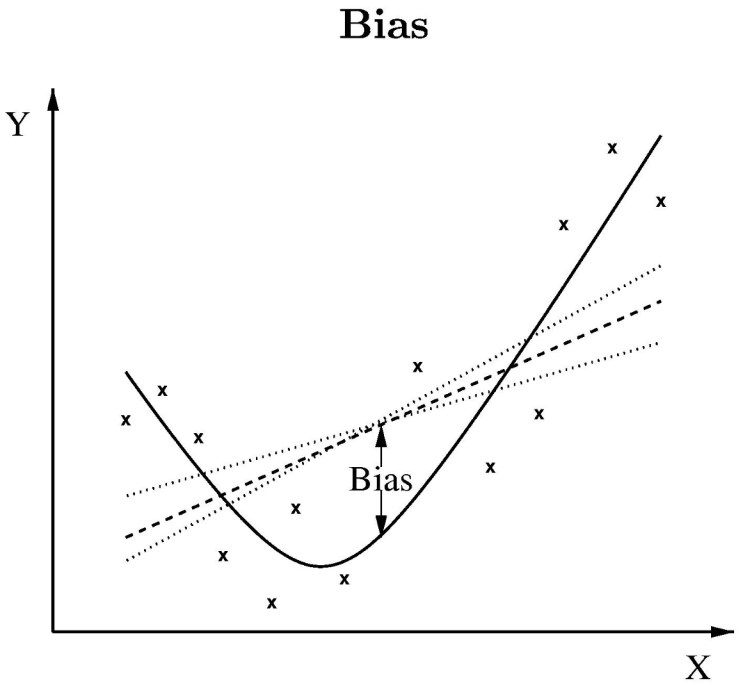
D_n - training data of size n

$$R(f^*) = \mathbb{E}_{XY}[(f^*(X) - Y)^2] = \mathbb{E}[\epsilon^2] = \sigma^2$$

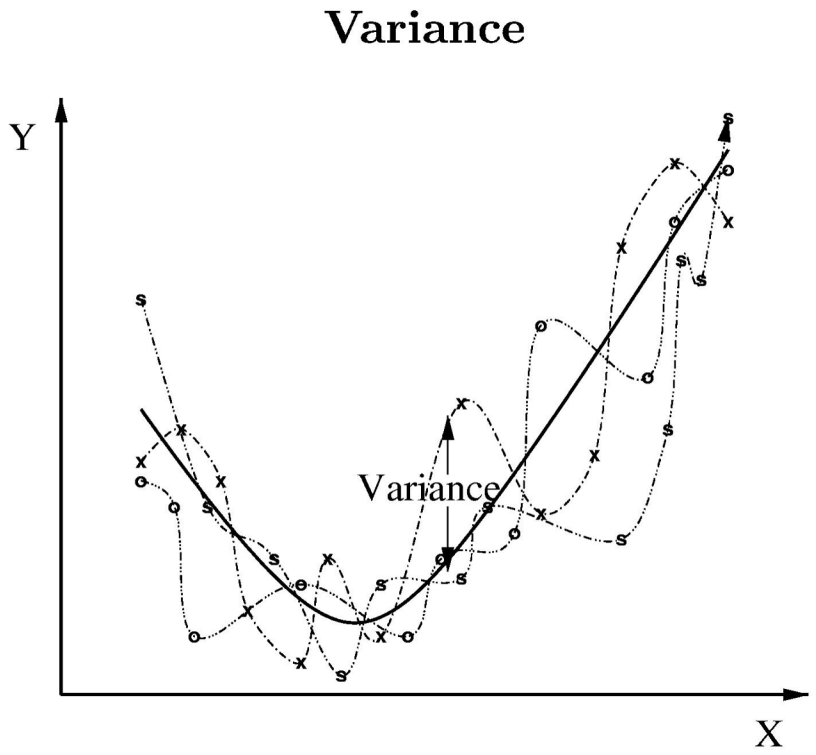


Notice that Optimal predictor does not have zero error

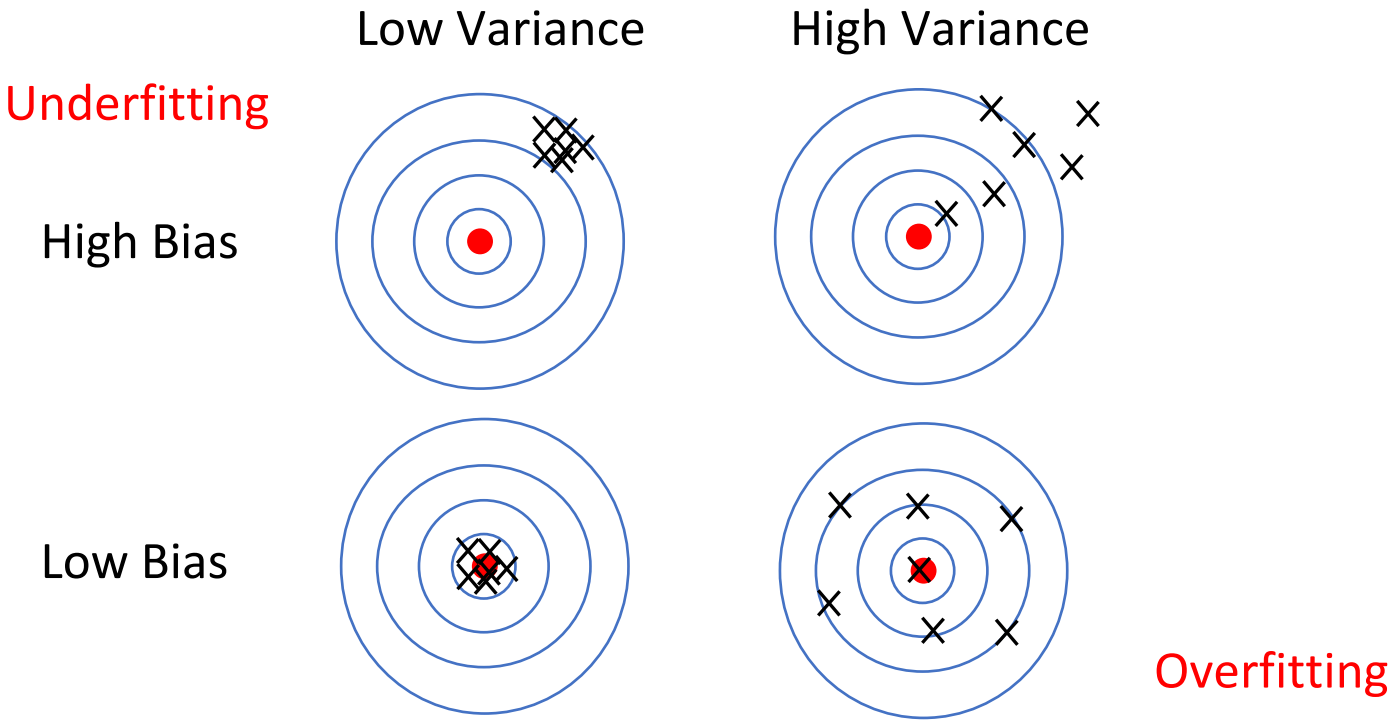
Bias



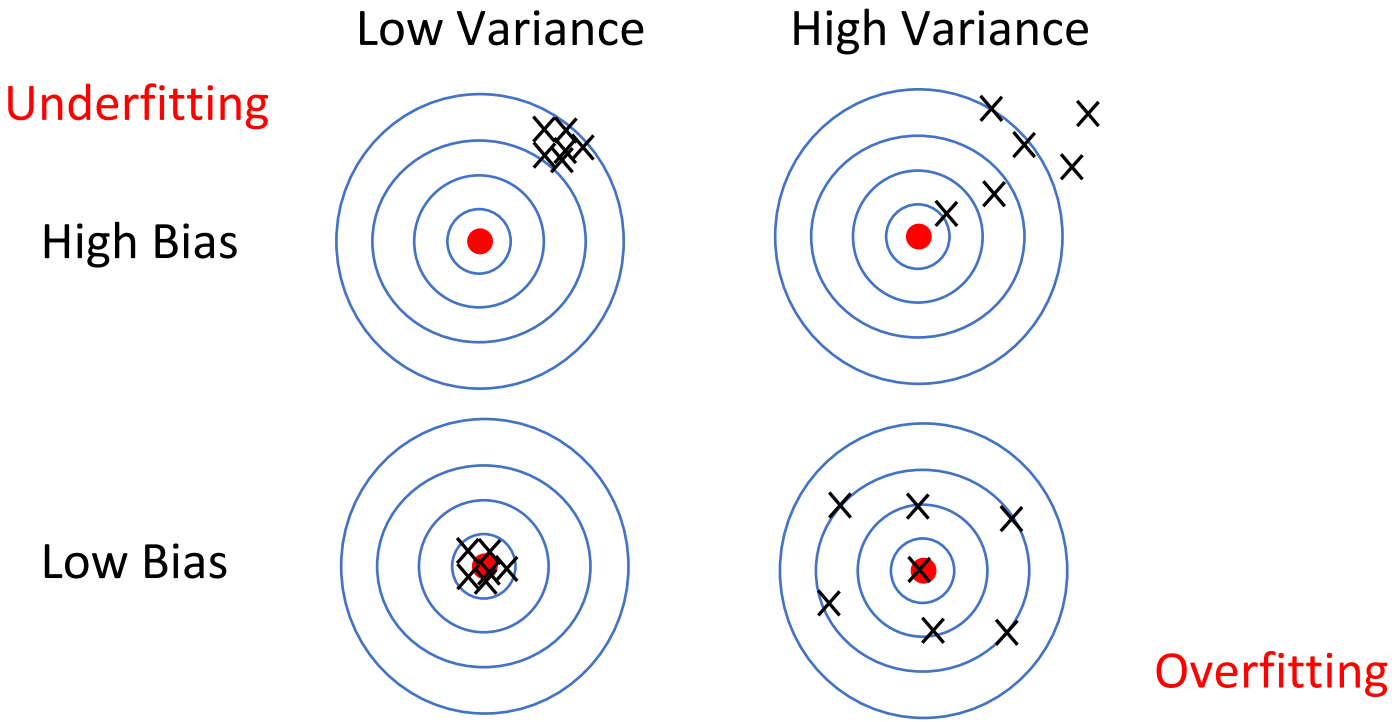
Variance



Bias and Variance: Intuition

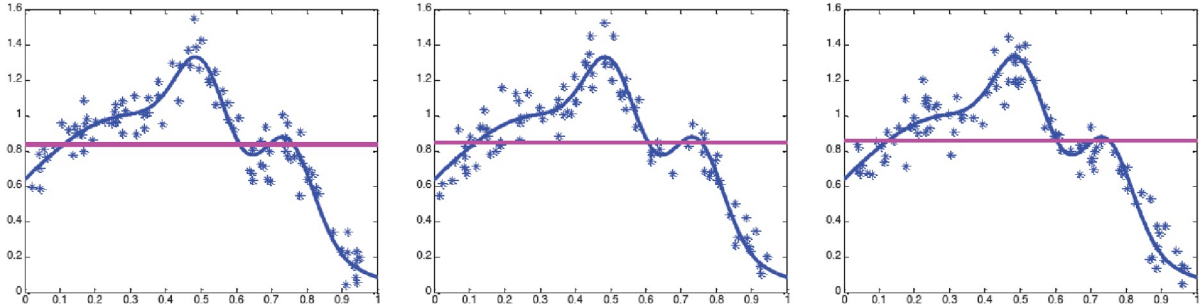


Bias and Variance: Intuition

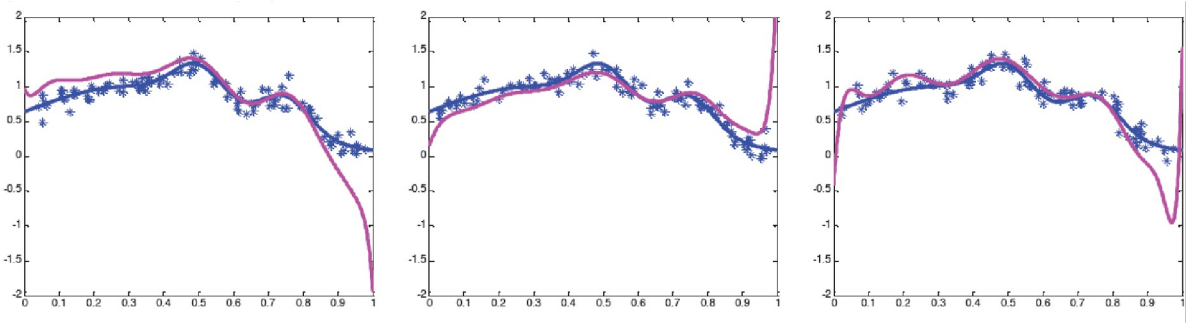


Bias -Variance Trade-off

- High bias, Low variance – poor approximation

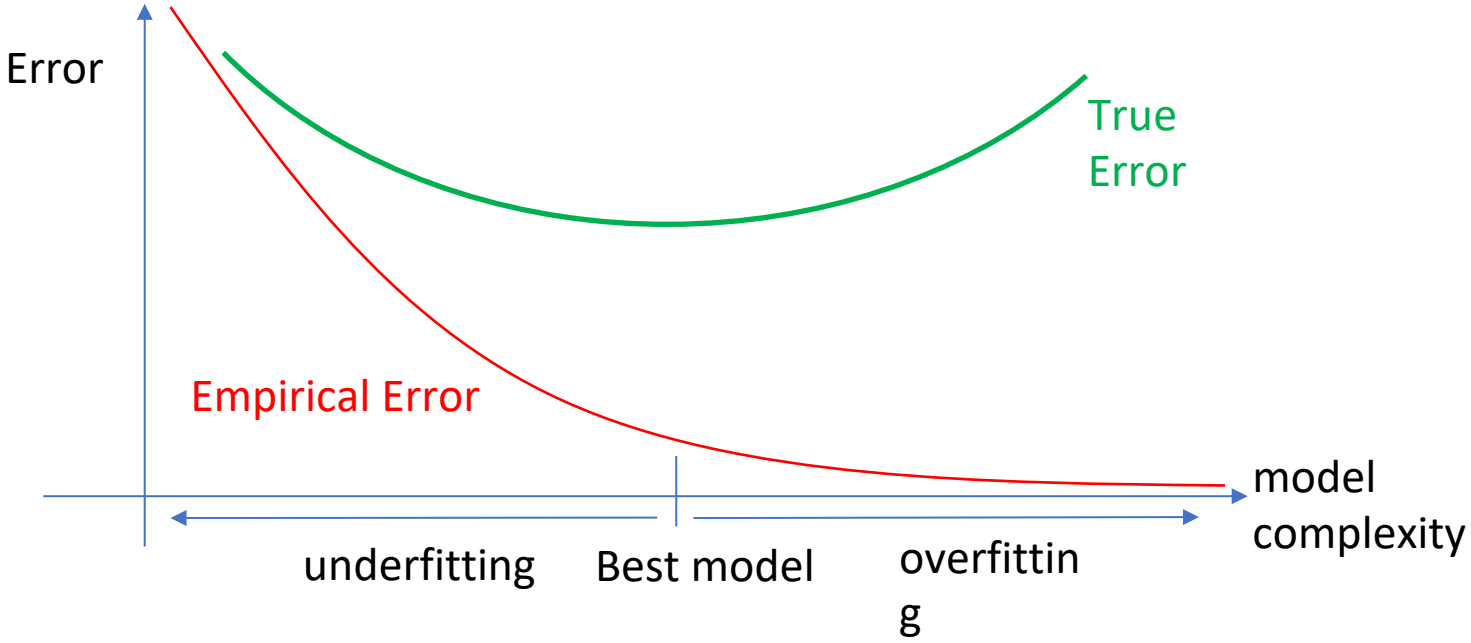


- Low bias, high variance – good approximation

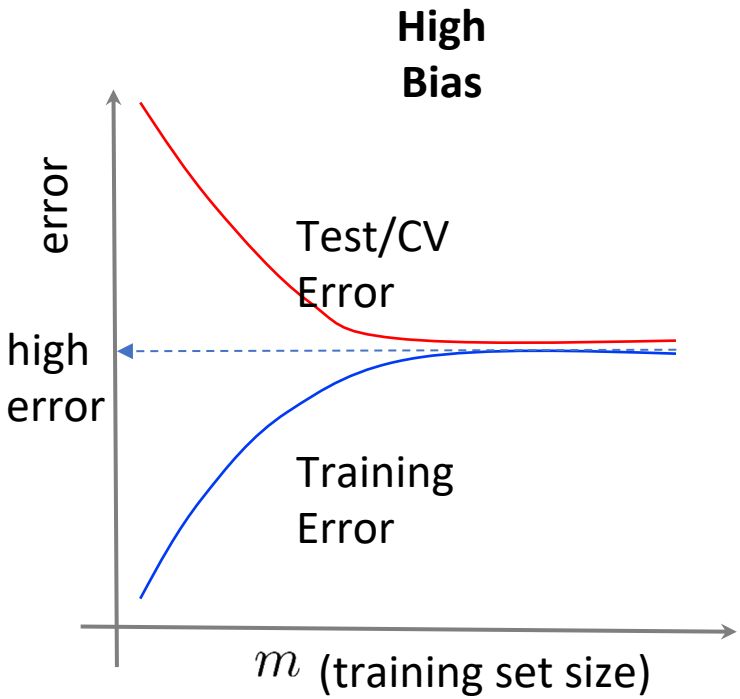


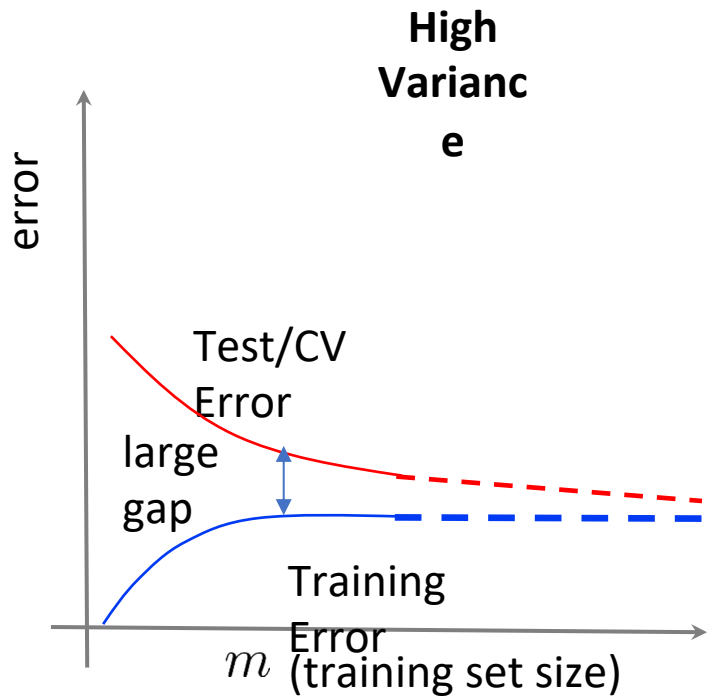
3 Independent training datasets

Model Optimization



Learning Curve



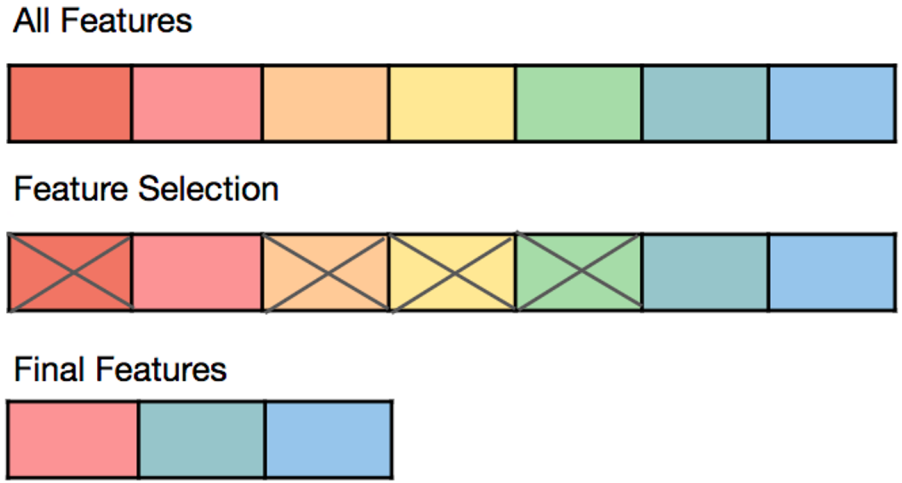


How to address Over-fitting

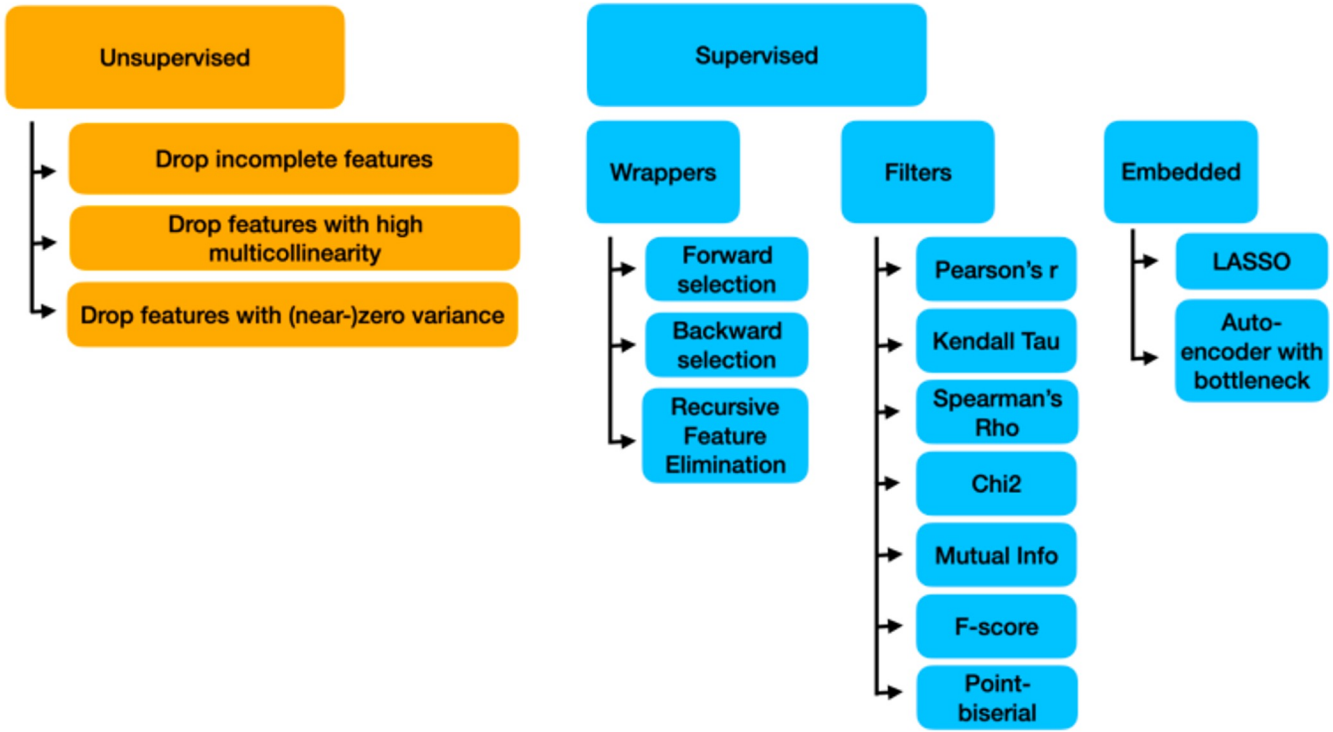
- **Reduce number of features**
  - *Feature selection*
  - Model selection algorithms
- **Regularization**
  - Incorporate model complexity for optimization, penalize complex models using prior knowledge
  - Keep all the features, but reduce magnitude/values of model parameters
  - Works well when we have a lot of features, each of which contributes a bit to the prediction

Feature Selection

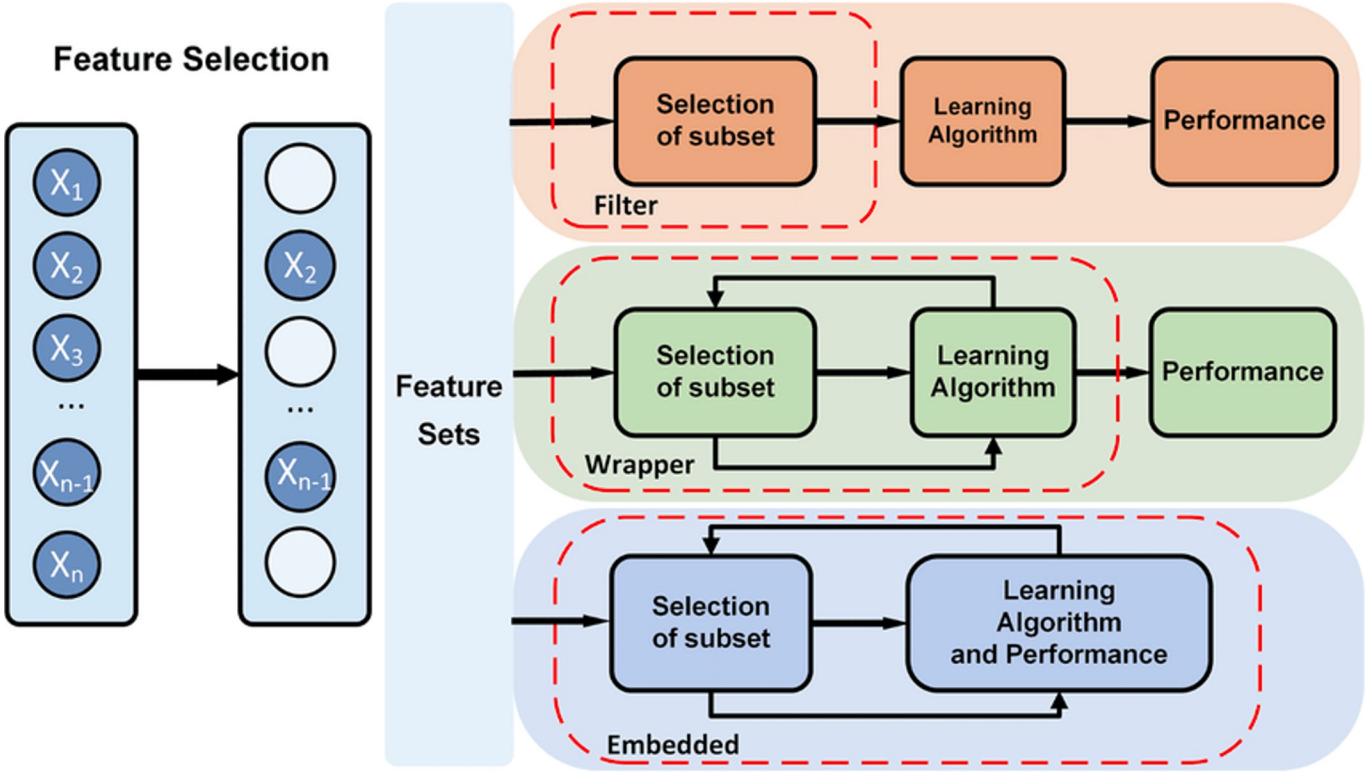
Idea: Find the best set of features that allows one to build optimized models **=> Reduce the model complexity**



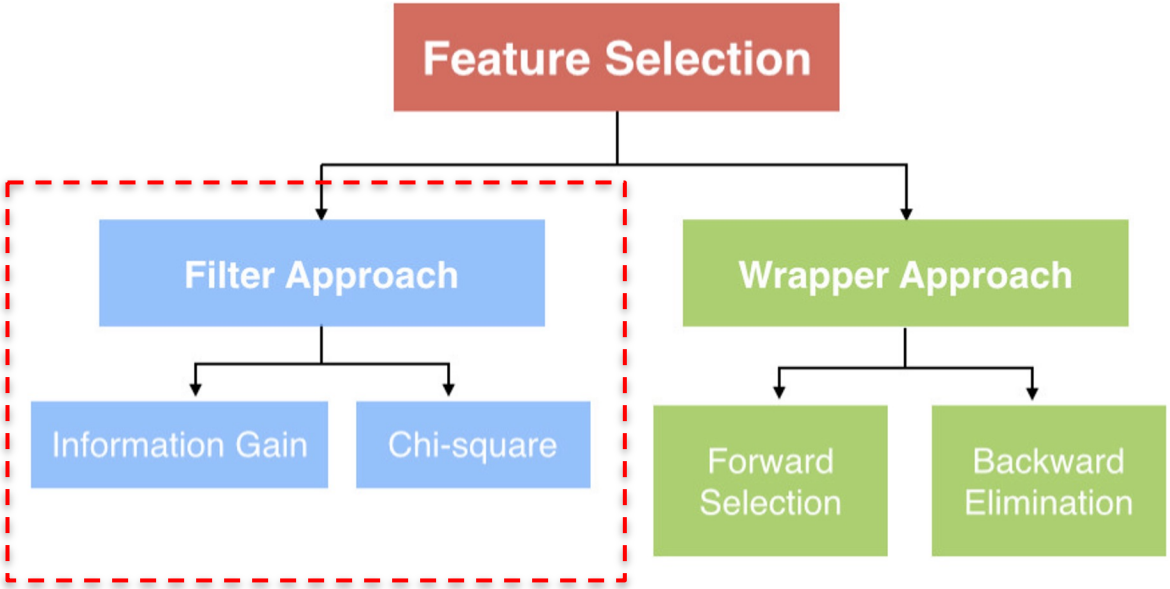
Feature Selection Methods



Supervised Feature Selection



Feature Selection - Filter Methods



Idea: Compute the importance of the feature **=> Choose the most important ones**

Feature Selection - Information Gain

$$E(T,X) = \sum_{c \in X} P(c)E(c)$$

		Play Golf		
		Yes	No	
Outlook	Sunny	3	2	5
	Overcast	4	0	4
	Rainy	2	3	5
				14

$$Gain(T,X) = Entropy(T) - Entropy(T,X)$$

G(PlayGolf, Outlook) = **E**(PlayGolf) – **E**(PlayGolf, Outlook)

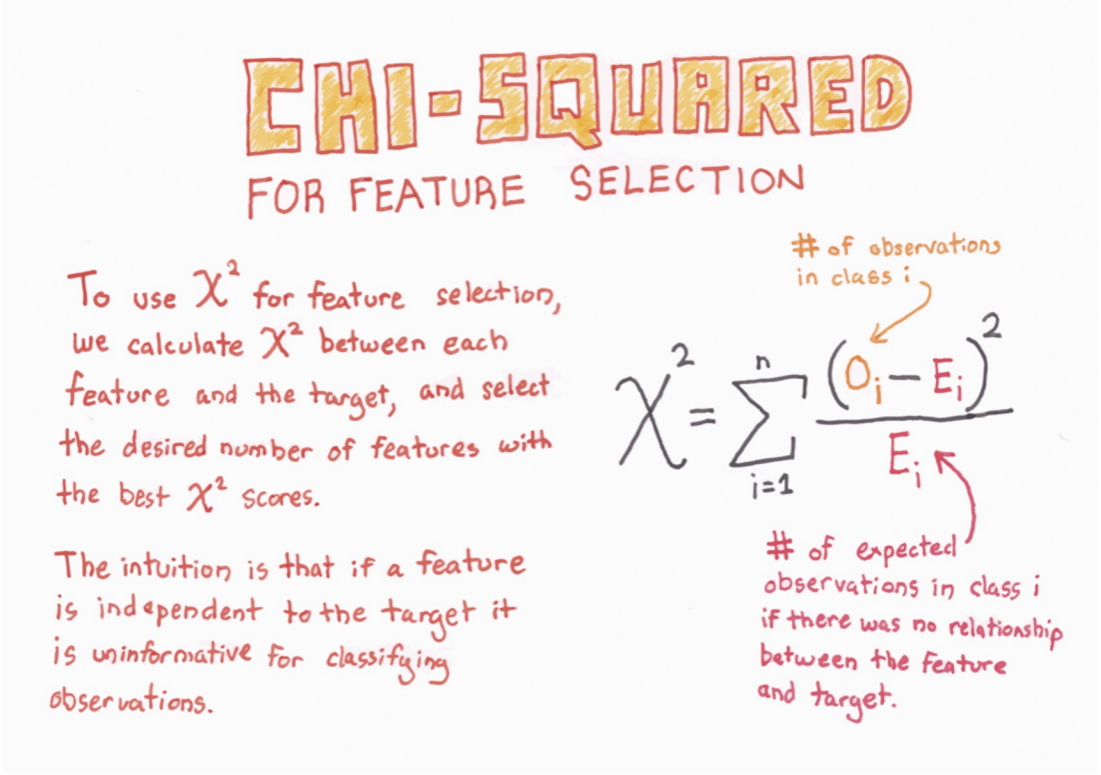
= 0.940 – 0.693 = 0.247

E(PlayGolf, Outlook) = **P**(Sunny)*__E__(3,2) + **P**(Overcast)*__E__(4,0) + **P**(Rainy)*__E__(2,3)

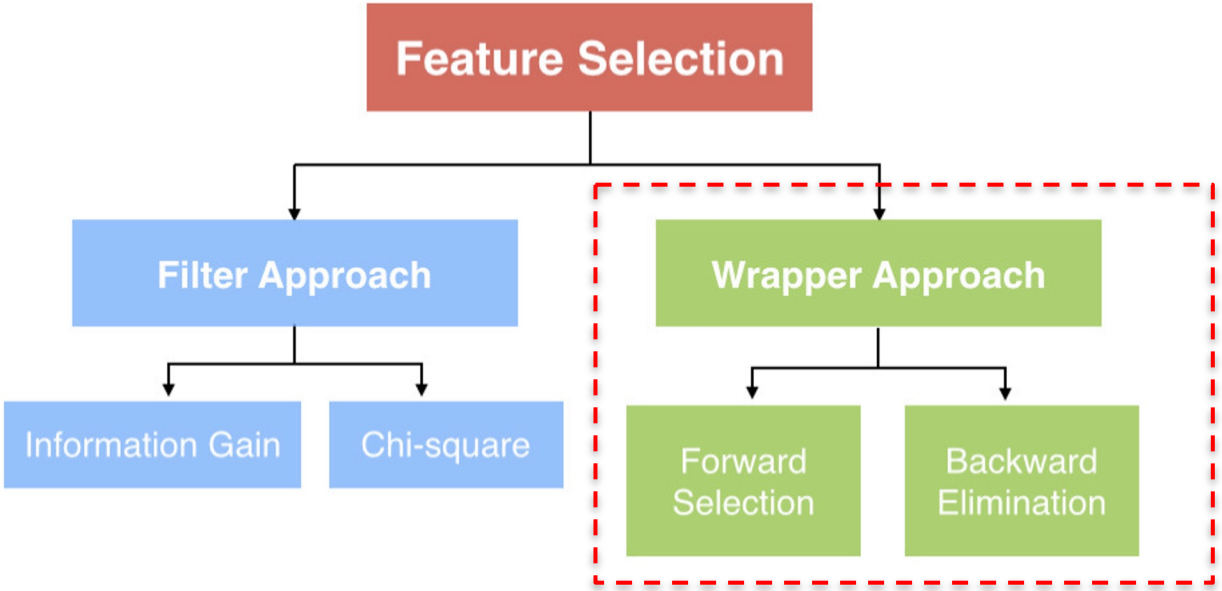
= (5/14)*0.971 + (4/14)*0.0 + (5/14)*0.971

= 0.693

Feature Selection - Chi-square



Feature Selection - Wrapper Methods



Idea: Gradually choose **the most important features**

Feature Selection - Regularization

- Regularized learning framework

$$\widehat{f}_n = \arg\min_{f \in \mathcal{F}} \left\{ \widehat{R}_n(f) + C(f) \right\}$$

Cost of model / model complexity

- Penalize complex models using prior knowledge.
- Two Examples
  - Regularized Linear Regression (rigid regression)
  - Regularized Logistic Regression

Regularized Linear Regression

- Linear Regression

$$\min_{\mathbf{w}} \frac{1}{2m} \sum_{i=1}^{m} (h(\mathbf{x}_i) - y_i)^2$$

- Regularized Linear Regression

$$\min_{\mathbf{w}} \frac{1}{2m} \sum_{i=1}^{m} (h(\mathbf{x}_i) - y_i)^2 + \phi(\mathbf{w})$$

- Choice of **regularizer**
  - "Rigid Regression"

$$\phi(\mathbf{w}) = \lambda \|\mathbf{w}\|_2^2$$

  - "Lasso" (Least absolute shrinkage and selection operator)

$$\phi(\mathbf{w}) = \lambda \|\mathbf{w}\|_1$$

Regularized Logistic Regression

$$\min_{\mathbf{w}} \sum_{i=1}^{N} \ln\left(1 + \exp(-y_i \mathbf{w}^\top \mathbf{x}_i)\right) + \phi(\mathbf{w})$$

- ℓ_2 -regularized Logistic Regression

$$\min_{\mathbf{w}} \sum_{i=1}^{N} \ln\left(1 + \exp(-y_i \mathbf{w}^\top \mathbf{x}_i)\right) + \lambda \|\mathbf{w}\|_2^2$$

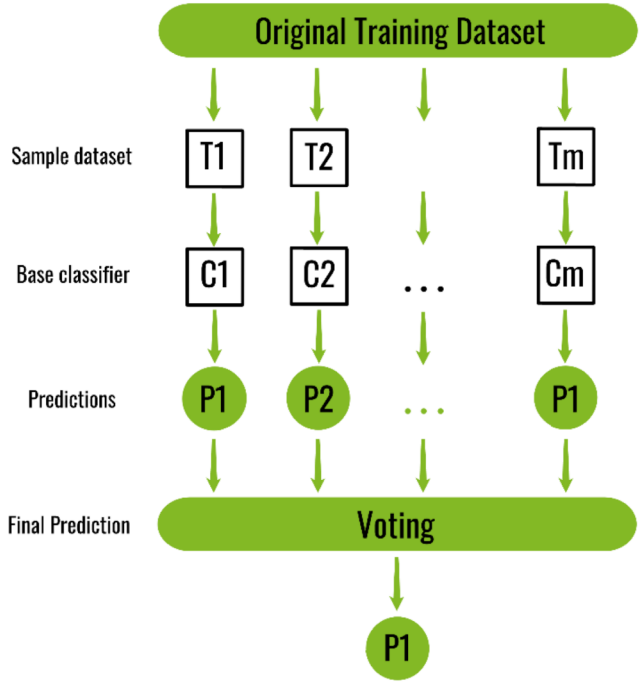
- ℓ_1 -regularized Logistic Regression ("Sparse Logistic Regressions")

$$\min_{\mathbf{w}} \sum_{i=1}^{N} \ln\left(1 + \exp(-y_i \mathbf{w}^\top \mathbf{x}_i)\right) + \lambda \|\mathbf{w}\|_1$$

Model Ensemble

- Basic Idea: Instead of learning one mode, learning several and combine them

- Typically improves the accuracy, often by a lot



Why does It Work?

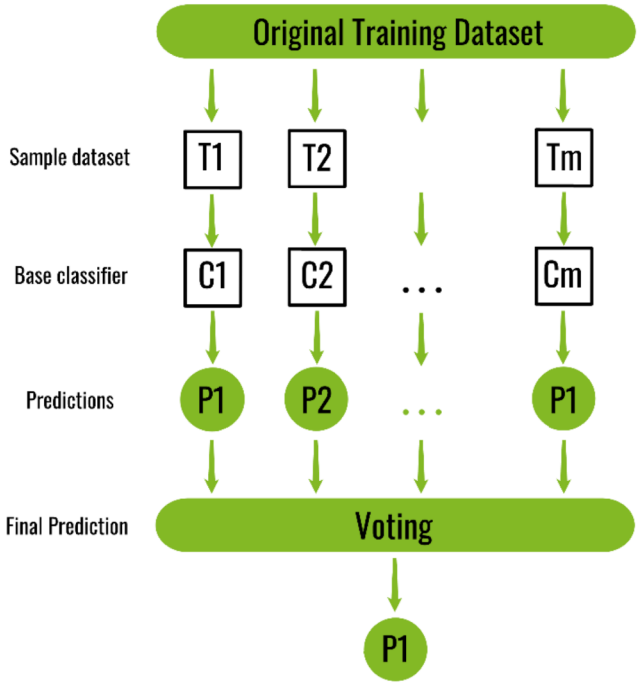
Suppose there are 25 base classifiers

- Each classifier has error rate, ε = 0.35
- Assume classifiers are independent
- Probability that the ensemble classifier makes a wrong prediction (i.e.,13 out of the 25 classifiers misclassified)

$$\sum_{i=13}^{25} \binom{25}{i} \varepsilon^i (1-\varepsilon)^{25-i} = 0.06$$

Bagging Classifiers

- In general, sampling from p(h|D) is difficult
  - P(h|D) is difficult to compute
  - P(h|D) is impossible to compute for non-probabilistic classifier such as SVM
- Bagging Classifiers:
  - Realize sampling p(h|D) by sampling training examples



Boostrap Sampling

- Bagging = Boostrap aggregating
- Boostrap sampling: given set D containing n training examples
  - Create D_i by drawing *n* examples at random with replacement from D
  - D_i expects to leave out about 0.37 of examples from D

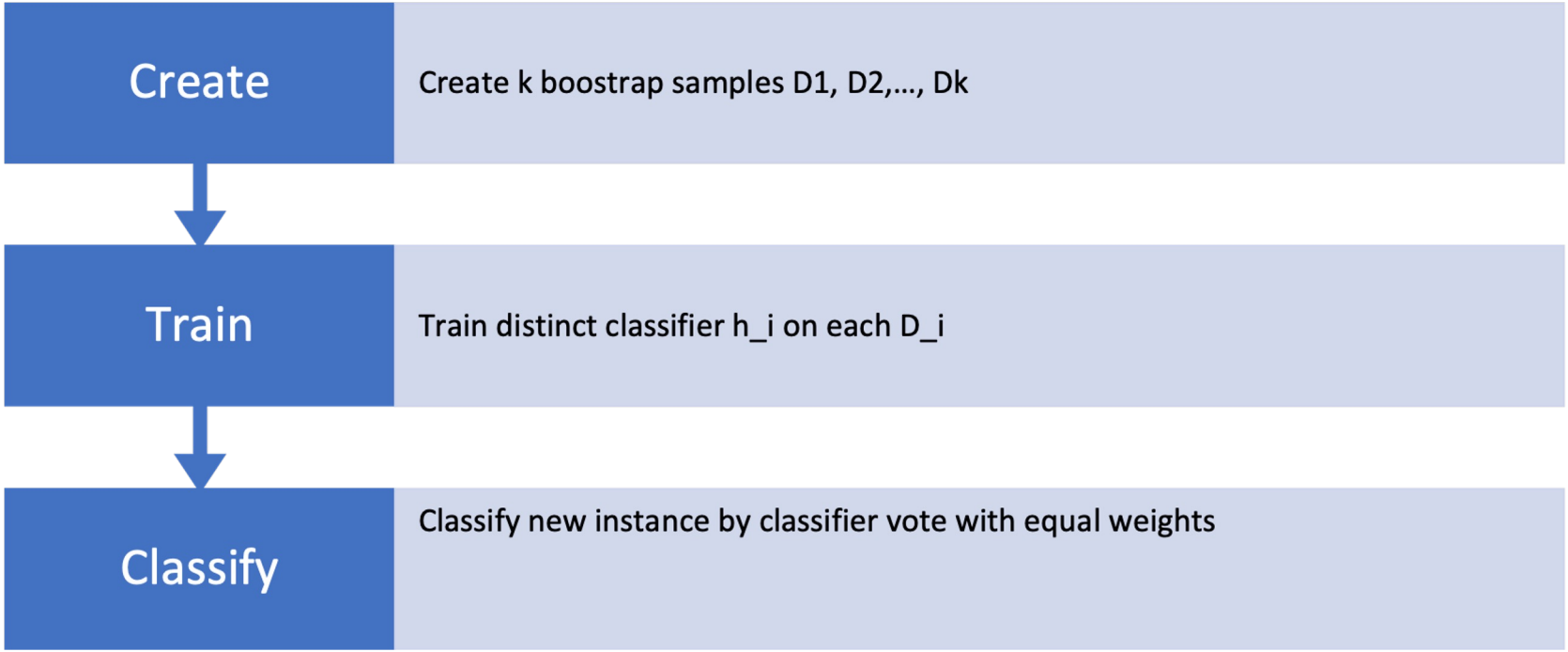
Bagging

- Sampling with replacement

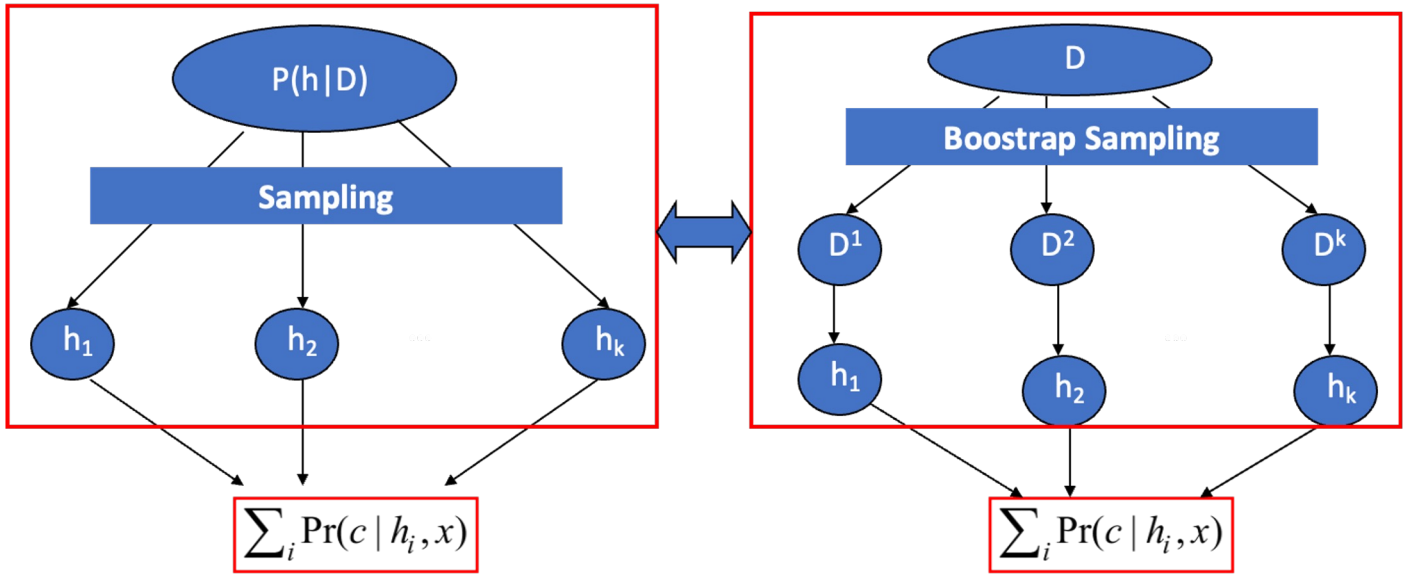
Original Data	1	2	3	4	5	6	7	8	9	10
Bagging (Round 1)	7	8	10	8	2	5	10	10	5	9
Bagging (Round 2)	1	4	9	1	2	3	2	7	3	2
Bagging (Round 3)	1	8	5	10	5	5	9	6	3	7

- Build classifier on each bootstrap sample
- Each sample has probability (1 – 1/n) of being remained
- All training data has probability (1 – 1/n)^n of being remained
  - This value tends to be **1/e ~ 0.37** for large n

Bagging Algorithm



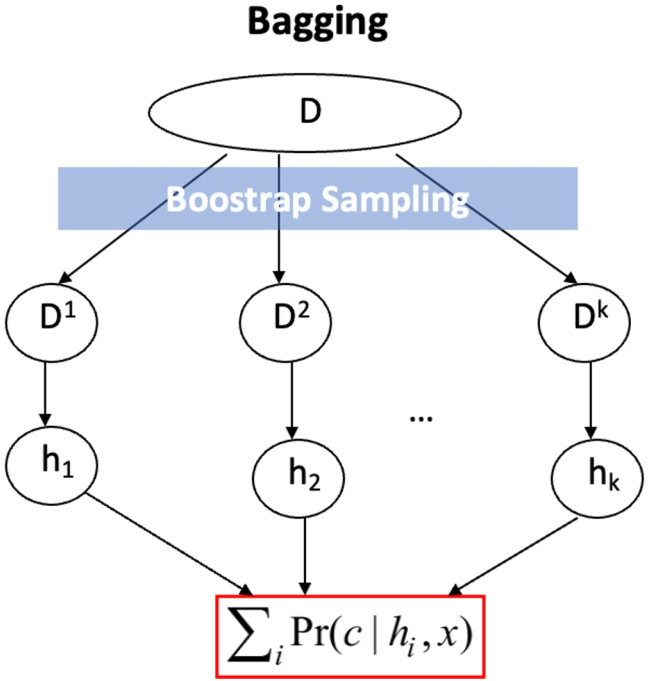
Bagging ~ Bayesian Average



Boostrap sampling is almost equivalent to sampling from posterior P(h|D)

Inefficiency with Bagging

- **Inefficient boostrap sampling:**
  - Every example has equal chance to be sampled
  - No distinction between "easy" examples and "difficult" examples
- **Inefficient model combination:**
  - A constant weight for each classifier
  - No distinction between accurate classifiers and inaccurate classifiers

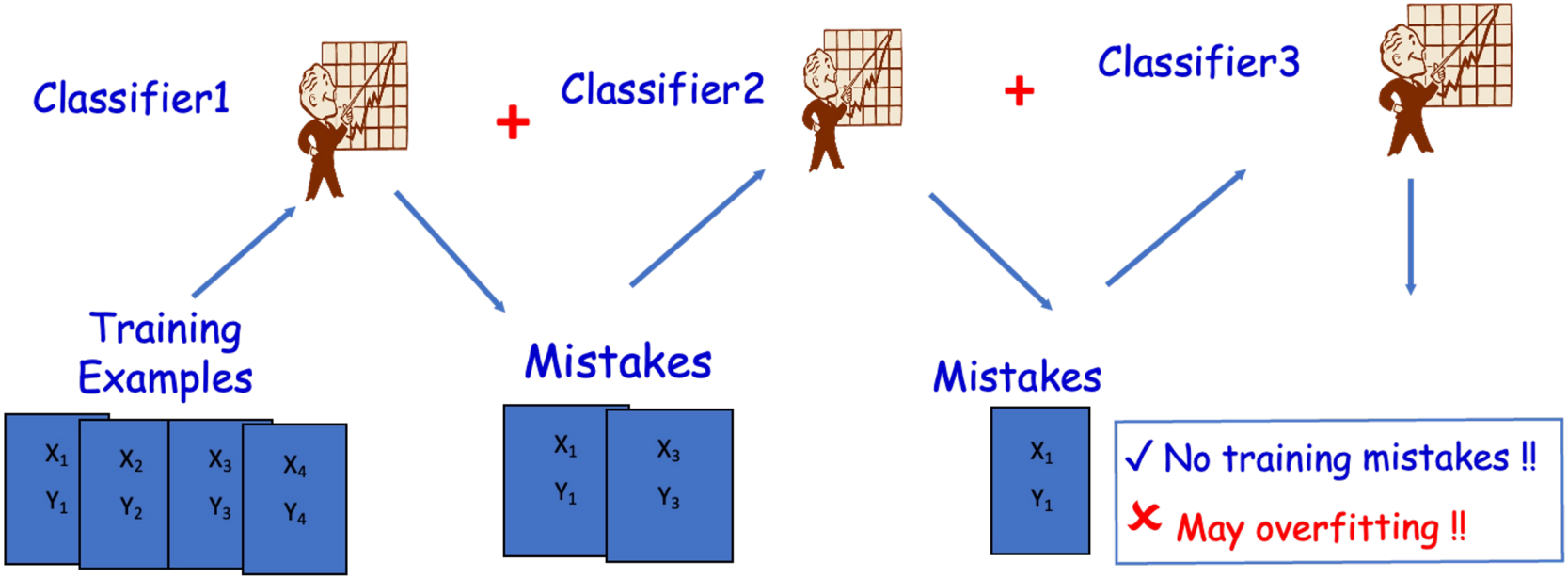


Improve the Efficiency of Bagging

- **Better sampling strategy**
  - Focus on the examples that are difficult to classify

- **Better combination strategy**
  - Accurate model should be assigned larger weights

Intuition


Classifier1
+
Classifier2
+
Classifier3
Training Examples
Mistakes
Mistakes
✓ No training mistakes !!
✗ May overfitting !!

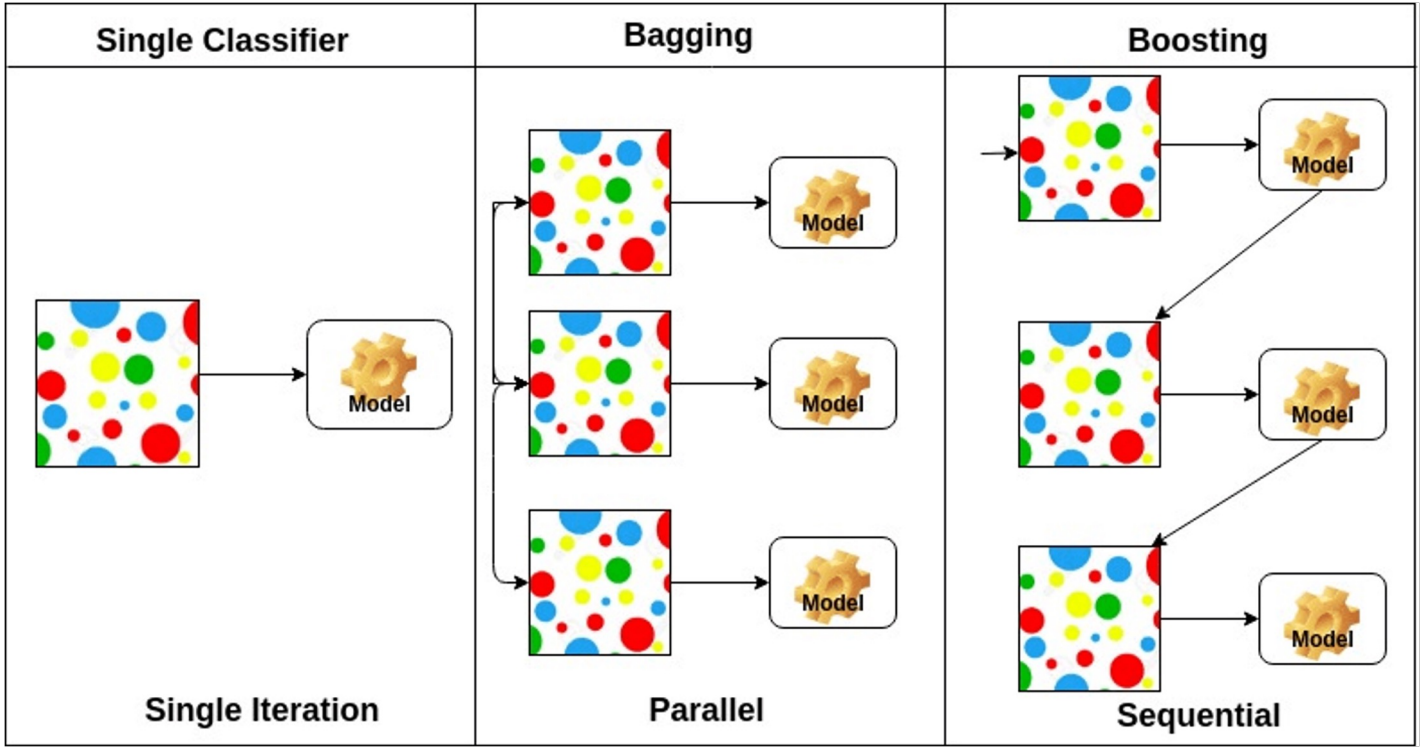
Boosting: Example

- Instances that are wrongly classified will have their weights increased
- Instances that are correctly classified will have their weights decreased

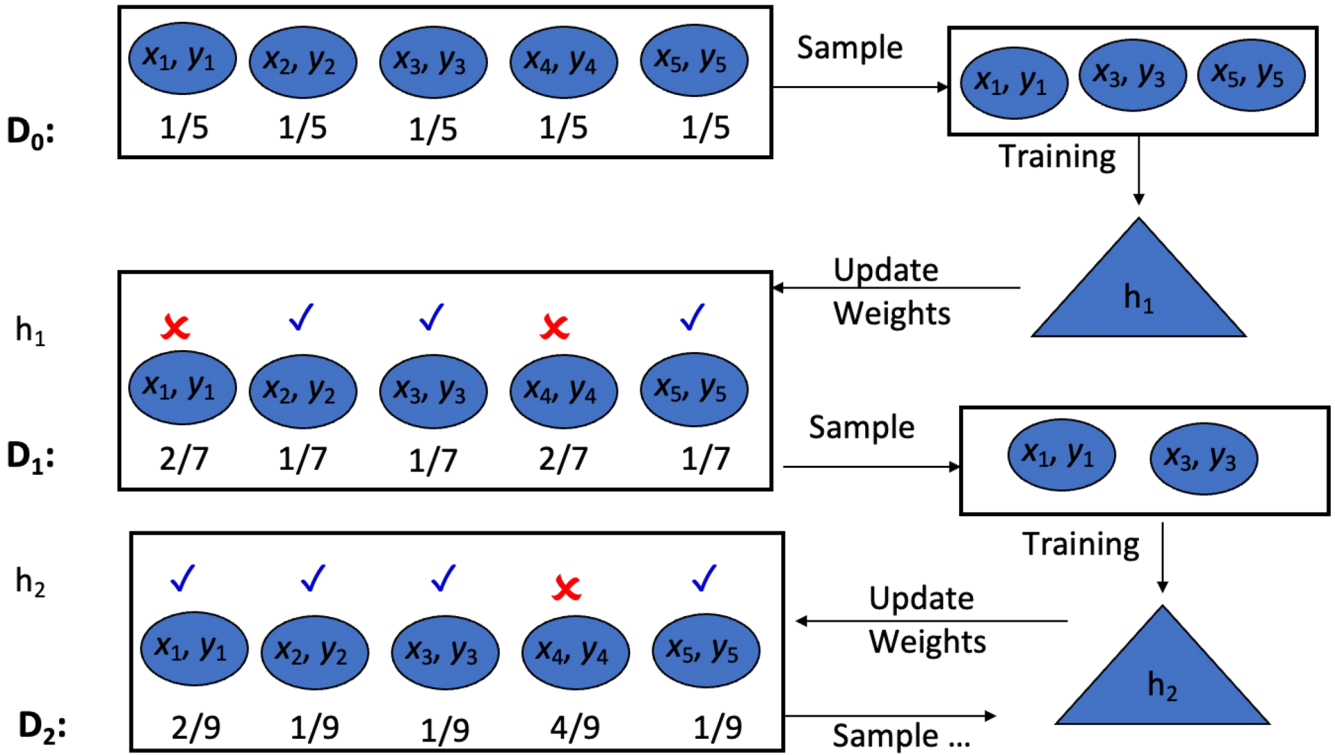
Original Data	1	2	3	4	5	6	7	8	9	10
Boosting (Round 1)	7	3	2	8	7	9	4	10	6	3
Boosting (Round 2)	5	4	9	4	2	5	1	7	4	2
Boosting (Round 3)	4	4	8	10	4	5	4	6	3	4

- Example 4 is hard to classify
- Its weight is increased, therefore it is more likely to be chosen again in subsequent rounds

AdaBoost

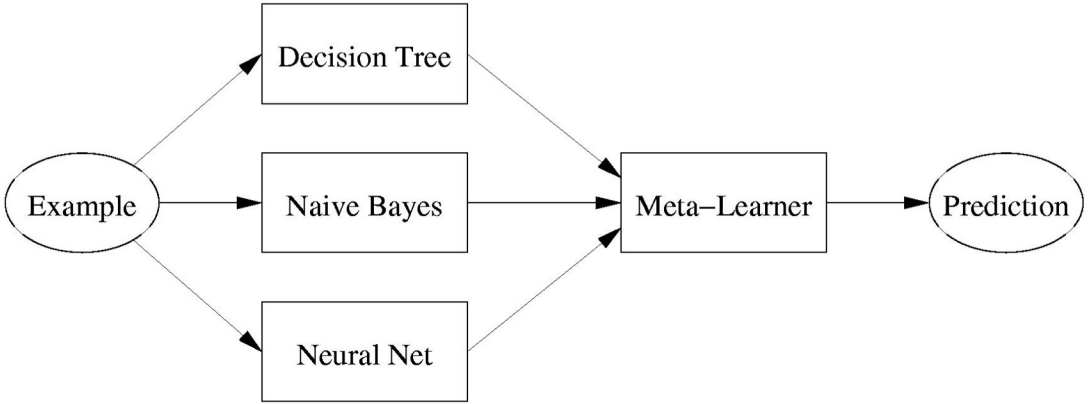

Single Classifier
Bagging
Boosting
Model
Model
Model
Model
Model
Model
Model
Single Iteration
Parallel
Sequential

AdaBoost Example



Stacking

- Apply multiple base learners
  (e.g.: decision trees, naive Bayes, neural nets)
- Meta-learner: Inputs = Base learner predictions
- Training by leave-one-out cross-validation:
  Meta-L. inputs = Predictions on left-out examples



Summary

- True Error versus Empirical Error
- Overfitting, Underfitting
- Bias-Variance Tradeoff
- Model Optimization
  - Feature Selection
  - Regularization
  - Model Ensemble

Thank you