Nonparametric Methods Recap…

Aarti Singh

Machine Learning 10-701/15-781
Oct 4, 2010

Nonparametric Methods

- Kernel Density estimate (also Histogram)

$$\widehat{p}(x) = \frac{1}{\Delta} \frac{\sum_{j=1}^{n} K\left(\frac{X_j - x}{\Delta}\right)}{n}$$

Weighted frequency

- Classification - K-NN Classifier

$$\widehat{f}_{kNN}(x) = \arg\max_{y} \; k_y$$

Majority vote

- Kernel Regression

Weighted average

$$\widehat{f}_n(x) = \sum_{i=1}^{n} w_i Y_i \quad \text{where} \quad w_i = \frac{K\left(\frac{X_i - x}{\Delta}\right)}{\sum_{j=1}^{n} K\left(\frac{X_j - x}{\Delta}\right)}$$

Kernel Regression as Weighted Least Squares

$$\min_f \sum_{i=1}^n w_i (f(X_i) - Y_i)^2 \qquad\qquad w_i(X) = \frac{K\left(\frac{X-X_i}{h}\right)}{\sum_{i=1}^n K\left(\frac{X-X_i}{h}\right)}$$

Weighted Least Squares

Kernel regression corresponds to locally constant estimator obtained from (locally) weighted least squares

i.e. set $f(X_i) = \beta$ (a constant)

Kernel Regression as Weighted Least Squares

set $f(X_i) = \beta$ (a constant)

$$\min_{\beta} \sum_{i=1}^{n} w_i(\beta - Y_i)^2$$

β → constant

$$w_i(X) = \frac{K\left(\frac{X - X_i}{h}\right)}{\sum_{i=1}^{n} K\left(\frac{X - X_i}{h}\right)}$$

$$\frac{\partial J(\beta)}{\partial \beta} = 2\sum_{i=1}^{n} w_i(\beta - Y_i) = 0$$

Notice that $\sum_{i=1}^{n} w_i = 1$

$$\Rightarrow \widehat{f}_n(X) = \widehat{\beta} = \sum_{i=1}^{n} w_i Y_i$$

Local Linear/Polynomial Regression

$$\min_f \sum_{i=1}^{n} w_i (f(X_i) - Y_i)^2 \qquad w_i(X) = \frac{K\left(\frac{X - X_i}{h}\right)}{\sum_{i=1}^{n} K\left(\frac{X - X_i}{h}\right)}$$

Weighted Least Squares

Local Polynomial regression corresponds to locally polynomial estimator obtained from (locally) weighted least squares

i.e. set $f(X_i) = \beta_0 + \beta_1 (X_i - X) + \frac{\beta_2}{2!}(X_i - X)^2 + \cdots + \frac{\beta_p}{p!}(X_i - X)^p$

(local polynomial of degree p around X)

More in 10-702 (statistical machine learning)

Summary

- Parametric vs Nonparametric approaches
  - Nonparametric models place very mild assumptions on the data distribution and provide good models for complex data

    Parametric models rely on very strong (simplistic) distributional assumptions

  - Nonparametric models (not histograms) requires storing and computing with the entire data set.

    Parametric models, once fitted, are much more efficient in terms of storage and computation.

Summary

- Instance based/non-parametric approaches

Four things make a memory based learner:

1. *A distance metric, dist(x,X_i)*
   Euclidean (and many more)
2. *How many nearby neighbors/radius to look at?*
   k, Δ/h
3. *A weighting function (optional)*
   W based on kernel K
4. *How to fit with the local points?*
   Average, Majority vote, Weighted average, Poly fit

What you should know…

- Histograms, Kernel density estimation
  - Effect of bin width/ kernel bandwidth
  - Bias-variance tradeoff

- K-NN classifier
  - Nonlinear decision boundaries

- Kernel (local) regression
  - Interpretation as weighted least squares
  - Local constant/linear/polynomial regression

Practical Issues in Machine Learning
Overfitting and Model selection

Aarti Singh

Machine Learning 10-701/15-781
Oct 4, 2010

True vs. Empirical Risk

True 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 point (X,Y)

Empirical Risk: Performance on training data

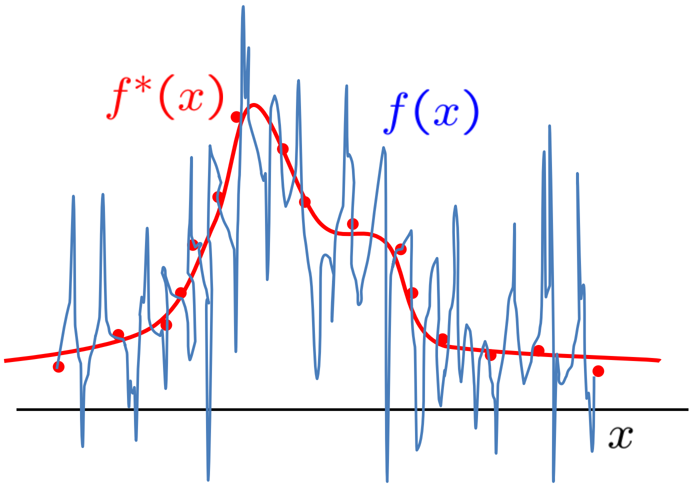
Classification – Proportion of misclassified examples $\frac{1}{n}\sum_{i=1}^{n} 1_{f(X_i) \neq Y_i}$

Regression – Average Squared Error $\frac{1}{n}\sum_{i=1}^{n} (f(X_i) - Y_i)^2$

Overfitting

Is the following predictor a good one?

$$f(x) = \begin{cases} Y_i, & x = X_i \text{ for } i = 1, \ldots, n \\ \text{any value}, & \text{otherwise} \end{cases}$$



What is its empirical risk? (performance on training data)

zero !

What about true risk?

> zero

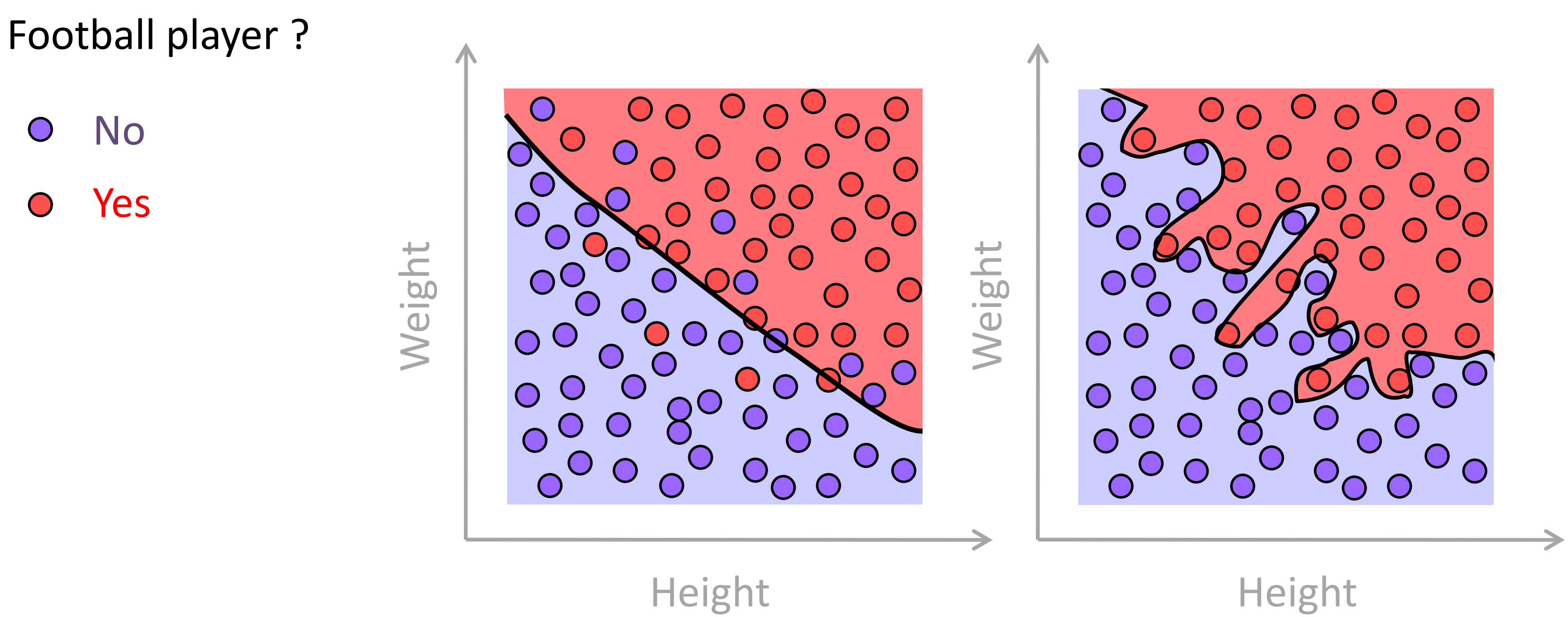
Will predict very poorly on new random test point:

Large generalization error !

Overfitting

If we allow very complicated predictors, we could overfit the training data.

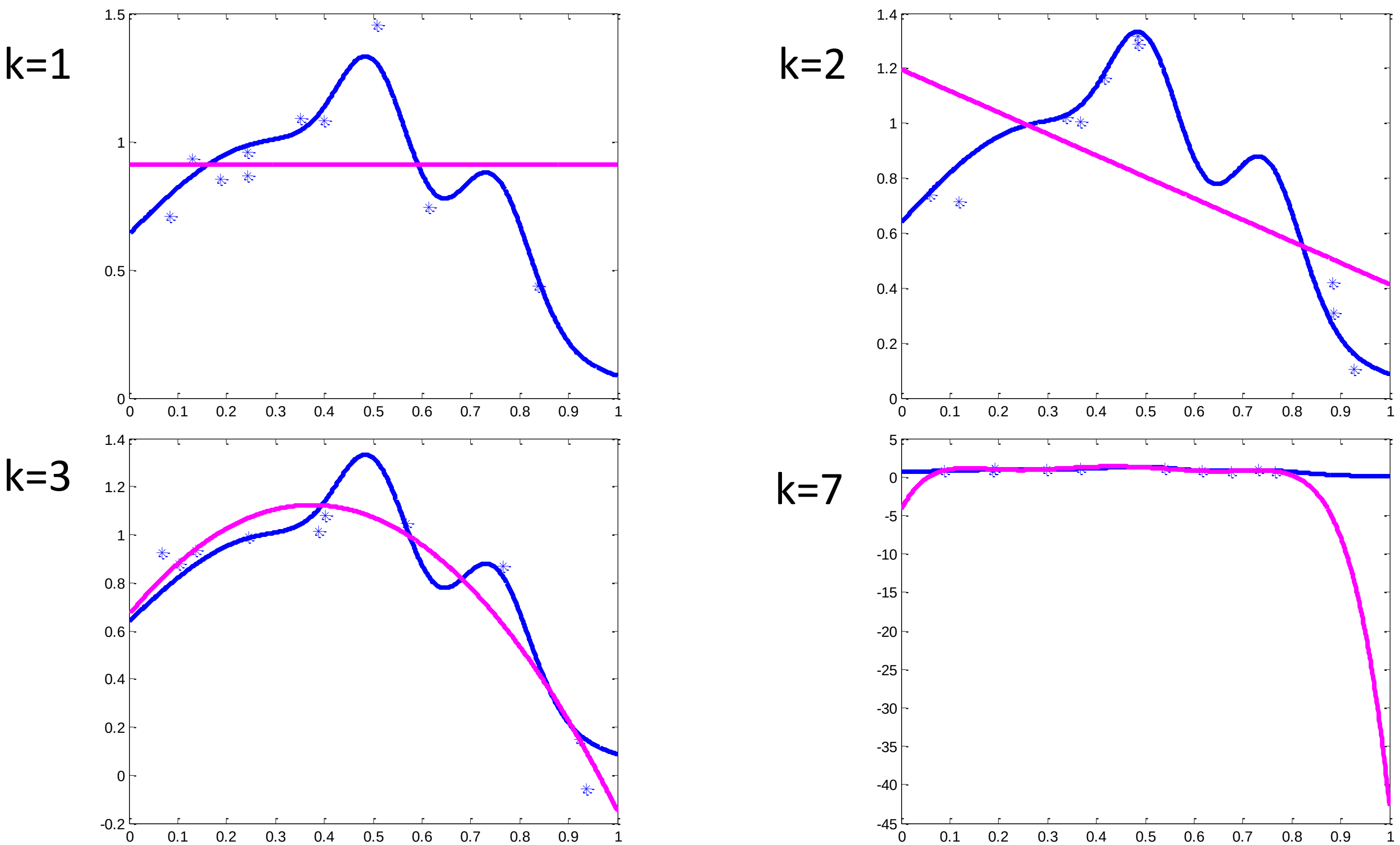
Examples: Classification (0-NN classifier)



Overfitting

If we allow very complicated predictors, we could overfit the training data.

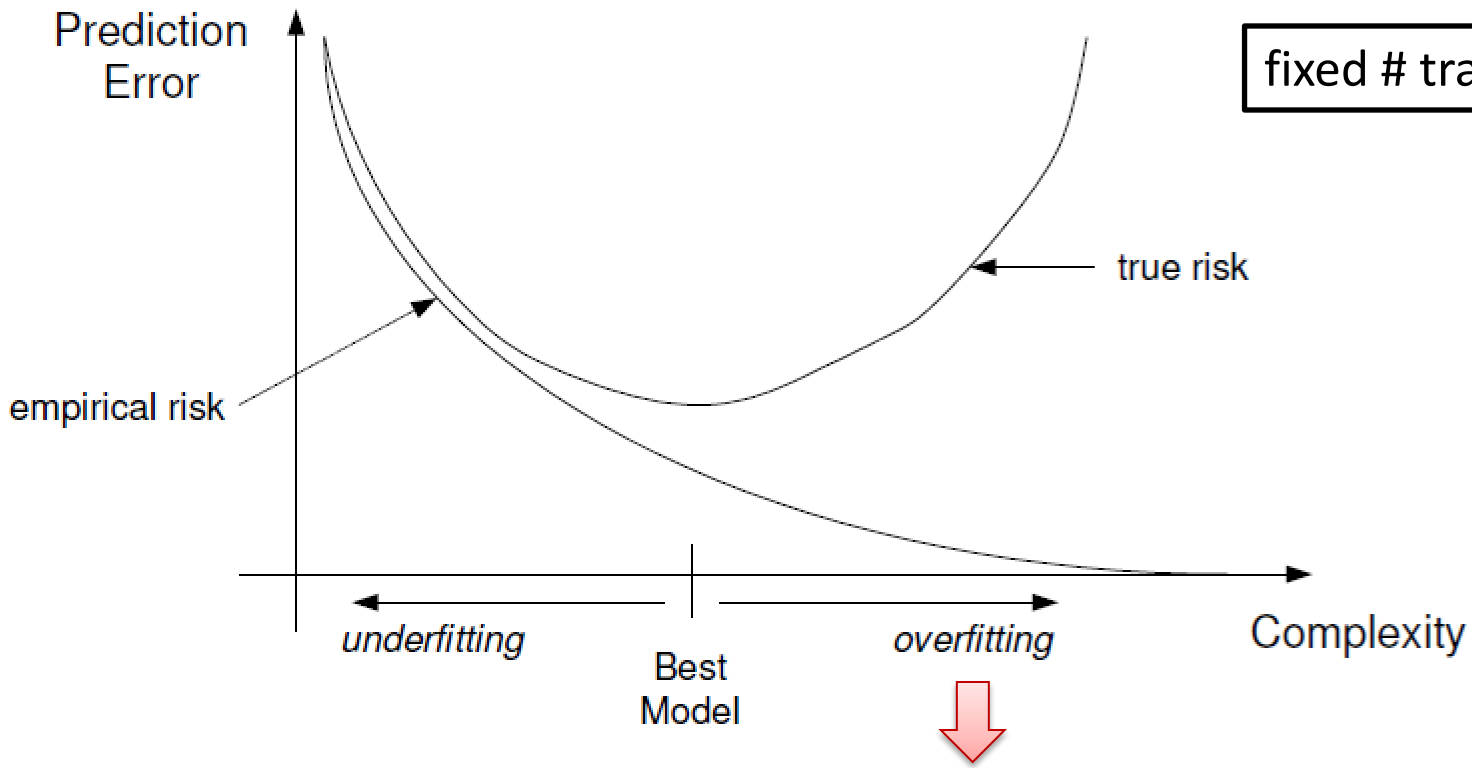
Examples: Regression (Polynomial of order k – degree up to k-1)



Effect of Model Complexity

If we allow very complicated predictors, we could overfit the training data.

fixed # training data



Empirical risk is no longer a good indicator of true risk

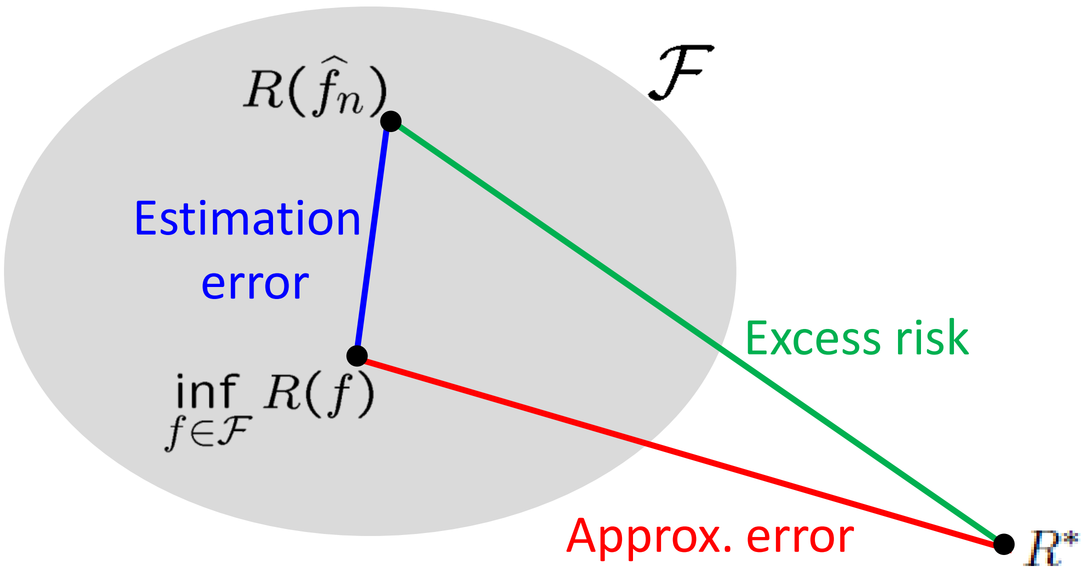
Behavior of True Risk

Want $\widehat{f}_n$ to be as good as optimal predictor f^*

Excess Risk $$E\left[R(\widehat{f}_n)\right] - R^* = \underbrace{\left(E[R(\widehat{f}_n)] - \inf_{f\in\mathcal{F}} R(f)\right)}_{\text{estimation error}} + \underbrace{\left(\inf_{f\in\mathcal{F}} R(f) - R^*\right)}_{\text{approximation error}}$$

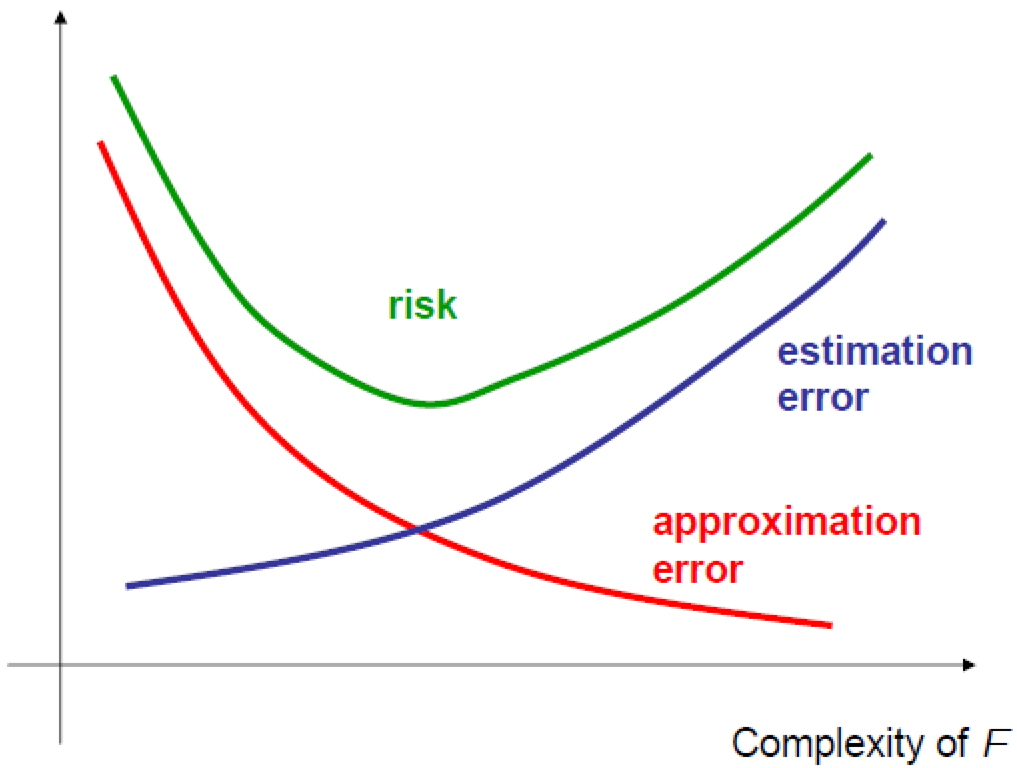
finite sample size + noise ← Due to randomness of training data

Due to restriction of model class

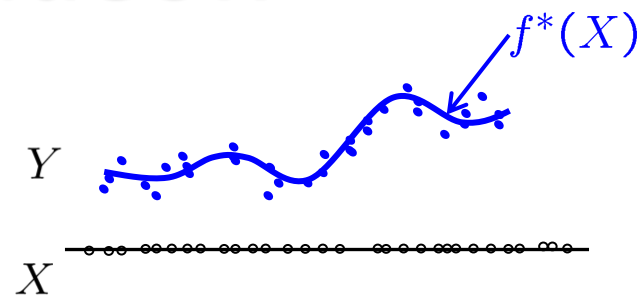


Behavior of True Risk

$$E\left[R(\widehat{f}_n)\right] - R^* \quad = \quad \underbrace{\left(E[R(\widehat{f}_n)] - \inf_{f\in\mathcal{F}} R(f)\right)}_{\text{estimation error}} + \underbrace{\left(\inf_{f\in\mathcal{F}} R(f) - R^*\right)}_{\text{approximation error}}$$



Bias – Variance Tradeoff



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

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

Notice: Optimal predictor does not have zero error

$$\mathbb{E}_{D_n}[R(\widehat{f}_n)] = \mathbb{E}_{X,Y,D_n}[(\widehat{f}_n(X) - Y)^2]$$

D_n - training data of size n

$$\vdots$$

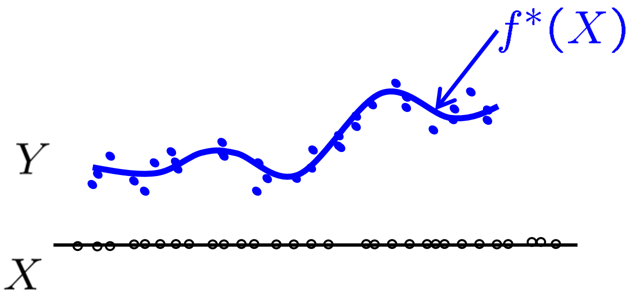
$$= \underbrace{\mathbb{E}_{X,Y,D_n}[(\widehat{f}_n(X) - \mathbb{E}_{D_n}[\widehat{f}_n(X)])^2]}_{\text{variance}} + \underbrace{\mathbb{E}_{X,Y}[(\mathbb{E}_{D_n}[\widehat{f}_n(X)] - f^*(X))^2]}_{\text{bias\^{}2}} + \underbrace{\sigma^2}_{\text{Noise var}}$$

Excess Risk = $\mathbb{E}_{D_n}[R(\widehat{f}_n)] - R^*$ = variance + bias^2

variance: Random component $\equiv$ est err

bias^2: $\equiv$ approx err

Bias – Variance Tradeoff: Derivation



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

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

Notice: Optimal predictor does not have zero error

D_n - training data of size n

$$\mathbb{E}_{D_n}[R(\widehat{f}_n)] = \mathbb{E}_{X,Y,D_n}[(\widehat{f}_n(X) - Y)^2]$$

$$= \mathbb{E}_{X,Y,D_n}\left[(\widehat{f}_n(X) - \mathbb{E}_{D_n}[\widehat{f}_n(X)] + \mathbb{E}_{D_n}[\widehat{f}_n(X)] - Y)^2\right]$$

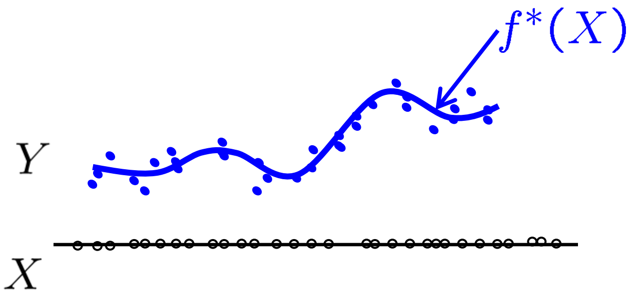
$$= \mathbb{E}_{X,Y,D_n}\left[(\widehat{f}_n(X) - \mathbb{E}_{D_n}[\widehat{f}_n(X)])^2 + (\mathbb{E}_{D_n}[\widehat{f}_n(X)] - Y)^2 + 2(\widehat{f}_n(X) - \mathbb{E}_{D_n}[\widehat{f}_n(X)])(\mathbb{E}_{D_n}[\widehat{f}_n(X)] - Y)\right]$$

$$= \mathbb{E}_{X,Y,D_n}\left[(\widehat{f}_n(X) - \mathbb{E}_{D_n}[\widehat{f}_n(X)])^2\right] + \mathbb{E}_{X,Y,D_n}\left[(\mathbb{E}_{D_n}[\widehat{f}_n(X)] - Y)^2\right]$$

$$+ \mathbb{E}_{X,Y}\left[2(\mathbb{E}_{D_n}[\widehat{f}_n(X)] - \mathbb{E}_{D_n}[\widehat{f}_n(X)])(\mathbb{E}_{D_n}[\widehat{f}_n(X)] - Y)\right]$$

0

Bias – Variance Tradeoff: Derivation



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

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

Notice: Optimal predictor does not have zero error

$$\mathbb{E}_{D_n}[R(\widehat{f}_n)] = \mathbb{E}_{X,Y,D_n}[(\widehat{f}_n(X) - Y)^2]$$

D_n - training data of size n

$$= \mathbb{E}_{X,Y,D_n}\left[(\widehat{f}_n(X) - \mathbb{E}_{D_n}[\widehat{f}_n(X)])^2\right] + \mathbb{E}_{X,Y,D_n}\left[(\mathbb{E}_{D_n}[\widehat{f}_n(X)] - Y)^2\right]$$

variance – how much does the predictor vary about its mean for different training datasets

Now, lets look at the second term:

$$\mathbb{E}_{X,Y,D_n}\left[(\mathbb{E}_{D_n}[\widehat{f}_n(X)] - Y)^2\right] = \mathbb{E}_{X,Y}\left[(\mathbb{E}_{D_n}[\widehat{f}_n(X)] - Y)^2\right]$$

Note: this term doesn't depend on D_n

Bias – Variance Tradeoff: Derivation

$$\mathbb{E}_{X,Y}\left[(\mathbb{E}_{D_n}[\widehat{f}_n(X)] - Y)^2\right] = \mathbb{E}_{X,Y}\left[(\mathbb{E}_{D_n}[\widehat{f}_n(X)] - f^*(X) - \epsilon)^2\right]$$

$$= \mathbb{E}_{X,Y}\left[(\mathbb{E}_{D_n}[\widehat{f}_n(X)] - f^*(X))^2 + \epsilon^2 - 2\epsilon(\mathbb{E}_{D_n}[\widehat{f}_n(X)] - f^*(X))\right]$$

$$= \mathbb{E}_{X,Y}\left[(\mathbb{E}_{D_n}[\widehat{f}_n(X)] - f^*(X))^2\right] + \mathbb{E}_{X,Y}\left[\epsilon^2\right] - 2\mathbb{E}_{X,Y}\left[\epsilon(\mathbb{E}_{D_n}[\widehat{f}_n(X)] - f^*(X))\right]$$

0 since noise is independent and zero mean

$$= \mathbb{E}_{X,Y}\left[(\mathbb{E}_{D_n}[\widehat{f}_n(X)] - f^*(X))^2\right] + \mathbb{E}_{X,Y}\left[\epsilon^2\right]$$

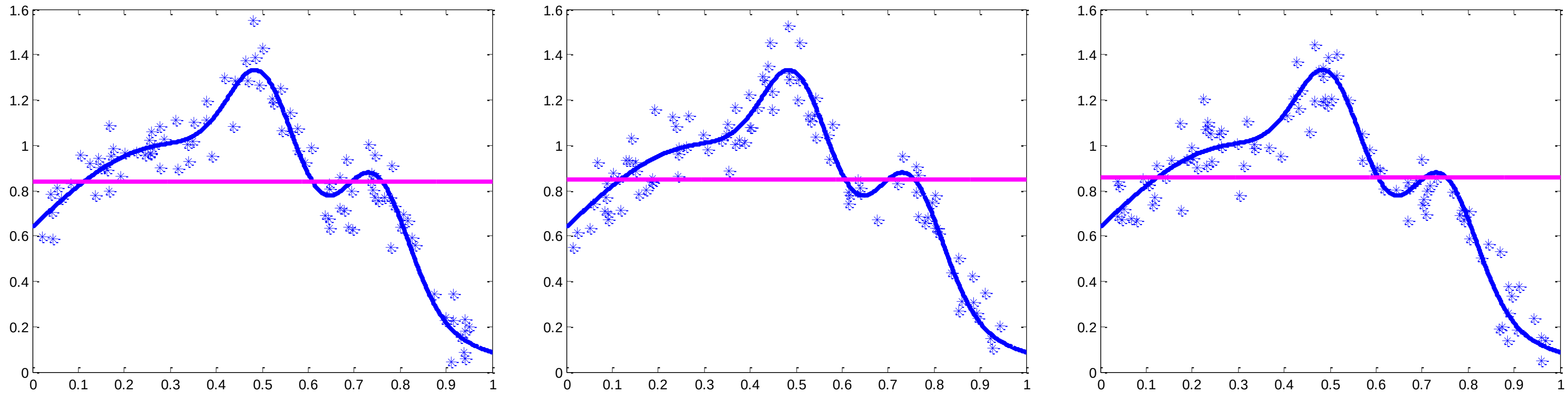
bias^2 – how much does the mean of the predictor differ from the optimal predictor

noise variance

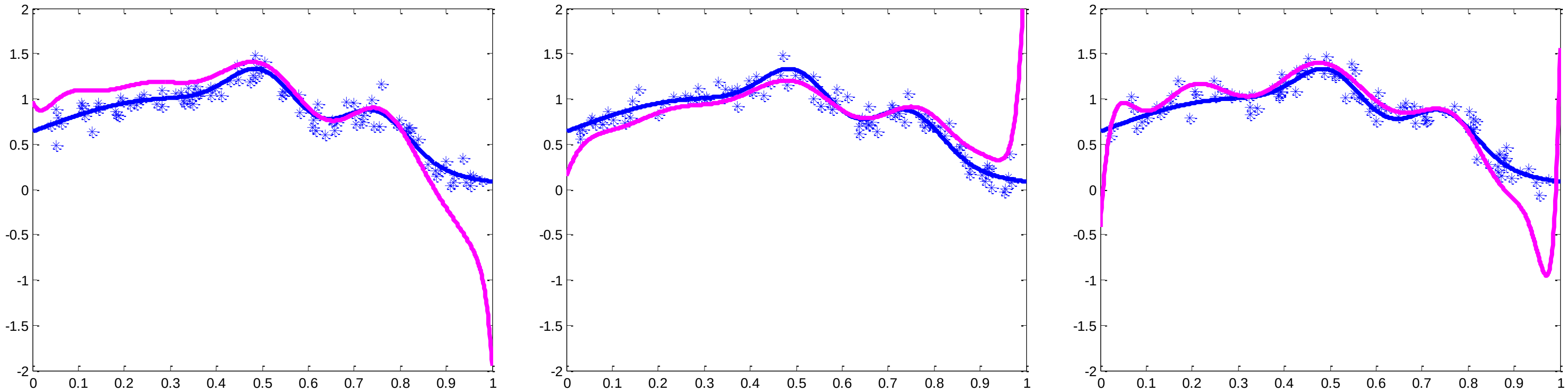
Bias – Variance Tradeoff

3 Independent training datasets

Large bias, Small variance – poor approximation but robust/stable



Small bias, Large variance – good approximation but instable



Examples of Model Spaces

Model Spaces with increasing complexity:

- Nearest-Neighbor classifiers with varying neighborhood sizes k = 1,2,3,…
  Small neighborhood => Higher complexity
- Decision Trees with depth k or with k leaves
  Higher depth/ More # leaves => Higher complexity
- Regression with polynomials of order k = 0, 1, 2, …
  Higher degree => Higher complexity
- Kernel Regression with bandwidth h
  Small bandwidth => Higher complexity

How can we select the right complexity model ?

Model Selection

Setup:

Model Classes $\{\mathcal{F}_\lambda\}_{\lambda \in \Lambda}$ of increasing complexity $\mathcal{F}_1 \prec \mathcal{F}_2 \prec \ldots$

$$\min_{\lambda} \min_{f \in \mathcal{F}_\lambda} J(f, \lambda)$$

We can select the right complexity model in a data-driven/adaptive way:

- ❑ Cross-validation
- ❑ Structural Risk Minimization
- ❑ Complexity Regularization
- ❑ *Information Criteria* - AIC, BIC, Minimum Description Length (MDL)

Hold-out method

We would like to pick the model that has smallest generalization error.

Can judge generalization error by using an independent sample of data.

Hold – out procedure:

n data points available $D \equiv \{X_i, Y_i\}_{i=1}^n$

1) Split into two sets: Training dataset $D_T = \{X_i, Y_i\}_{i=1}^m$ Validation dataset $D_V = \{X_i, Y_i\}_{i=m+1}^n$ NOT test Data !!

2) Use D_T for training a predictor from each model class:

$$\widehat{f}_\lambda = \arg\min_{f \in \mathcal{F}_\lambda} \widehat{R}_T(f)$$

→ Evaluated on training dataset D_T

Hold-out method

3) Use *Dv* to select the model class which has smallest empirical error on D_V

$$\widehat{\lambda} = \arg\min_{\lambda \in \Lambda} \widehat{R}_V(\widehat{f}_\lambda)$$

→ Evaluated on validation dataset D_V

4) Hold-out predictor

$$\widehat{f} = \widehat{f}_{\widehat{\lambda}}$$

Intuition: Small error on one set of data will not imply small error on a randomly sub-sampled second set of data

Ensures method is "stable"

Hold-out method

Drawbacks:

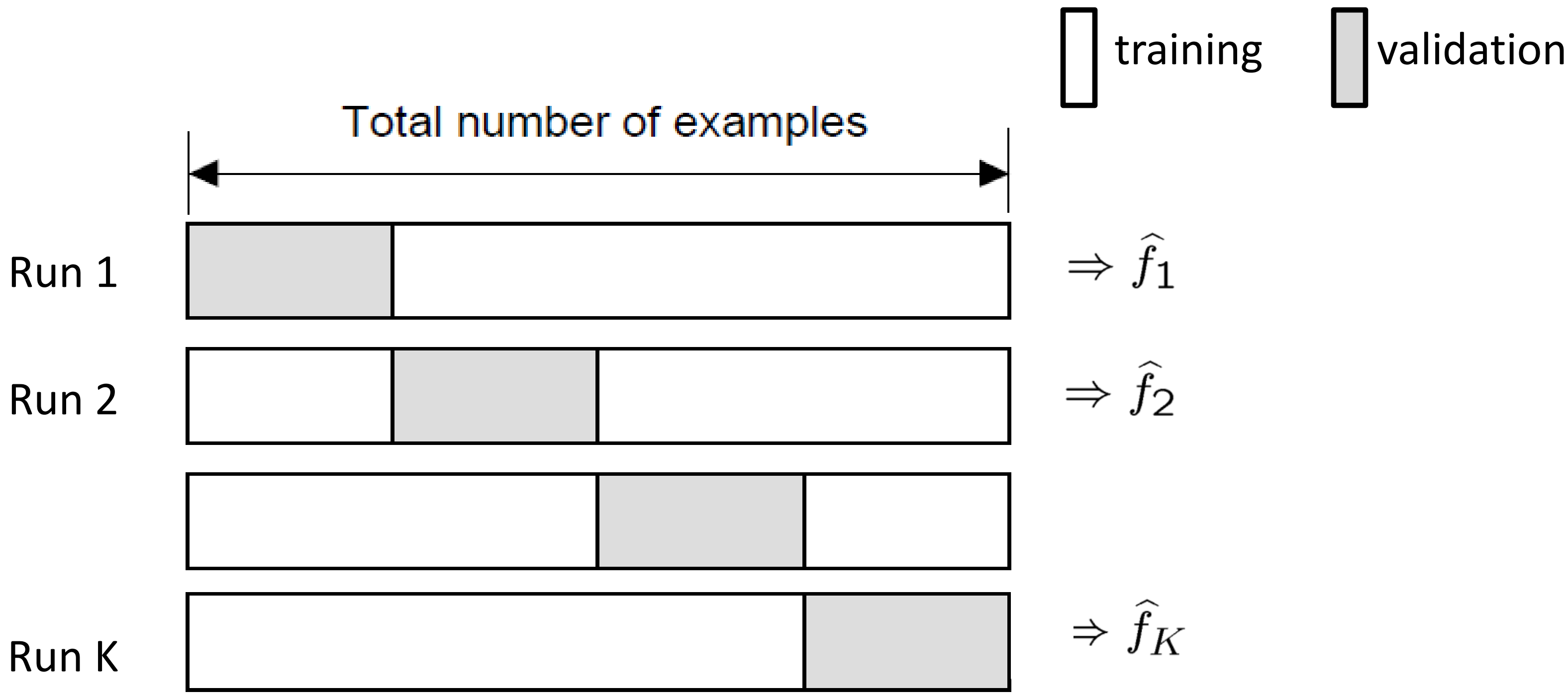
- May not have enough data to afford setting one subset aside for getting a sense of generalization abilities
- Validation error may be misleading (bad estimate of generalization error) if we get an "unfortunate" split

Limitations of hold-out can be overcome by a family of random sub-sampling methods at the expense of more computation.

Cross-validation

K-fold cross-validation

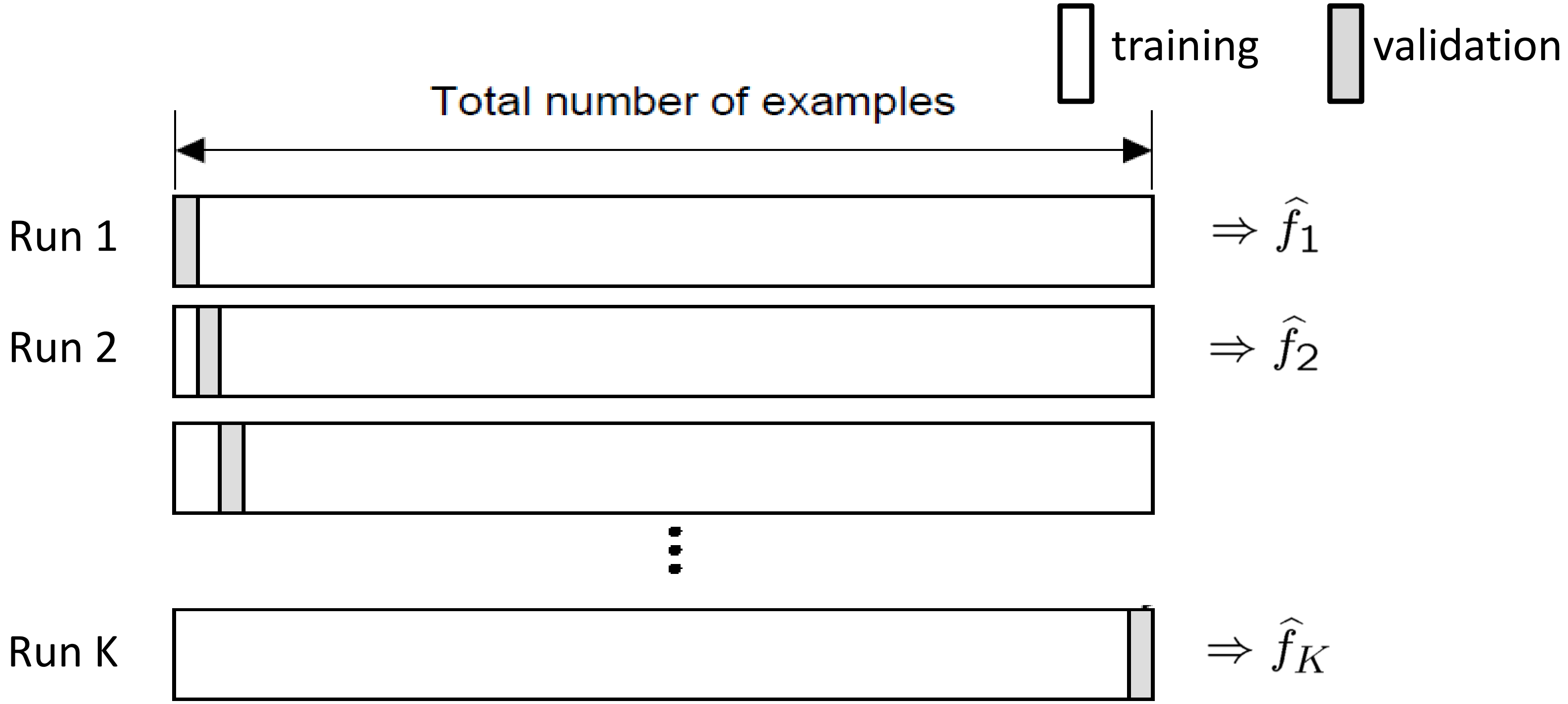
Create K-fold partition of the dataset.
Form K hold-out predictors, each time using one partition as validation and rest K-1 as training datasets.
Final predictor is average/majority vote over the K hold-out estimates.



Cross-validation

Leave-one-out (LOO) cross-validation

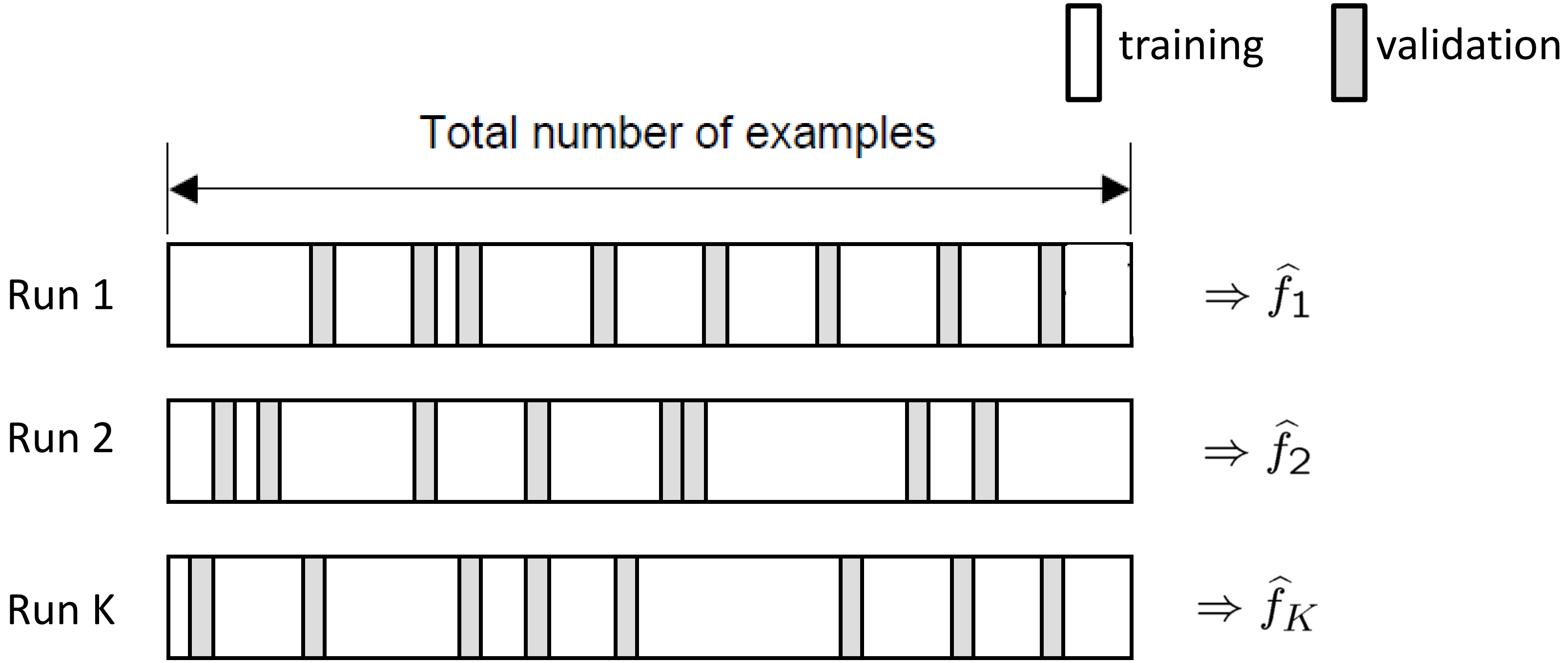
Special case of K-fold with K=n partitions
Equivalently, train on n-1 samples and validate on only one sample per run for n runs



Cross-validation

Random subsampling

Randomly subsample a fixed fraction αn ($0 < \alpha < 1$) of the dataset for validation.
Form hold-out predictor with remaining data as training data.
Repeat K times
Final predictor is average/majority vote over the K hold-out estimates.



Estimating generalization error

Generalization error $\mathbb{E}_D[R(\widehat{f}_n)]$

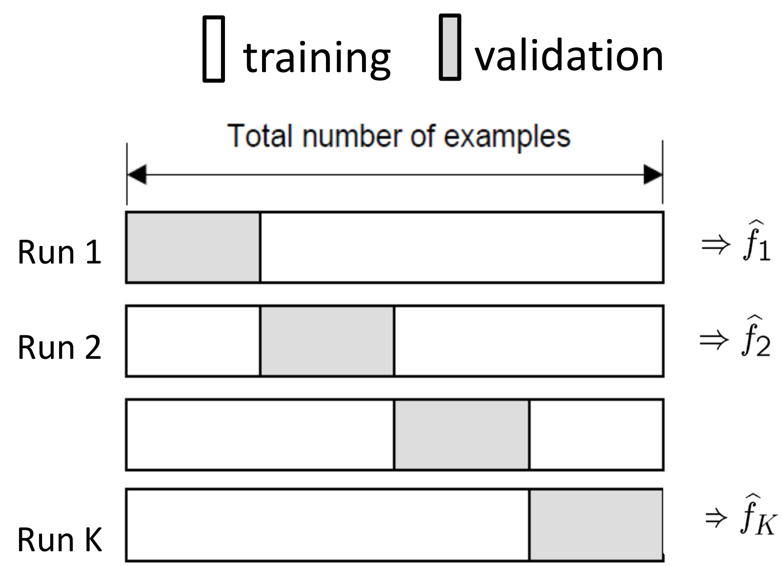
Hold-out ≡ 1-fold: Error estimate = $\widehat{R}_V(\widehat{f}_T)$

K-fold/LOO/random sub-sampling: Error estimate = $\frac{1}{K}\sum_{k=1}^{K}\widehat{R}_{V_k}(\widehat{f}_{T_k})$

We want to estimate the error of a predictor based on n data points.

If K is large (close to n), bias of error estimate is small since each training set has close to n data points.

However, variance of error estimate is high since each validation set has fewer data points and $\widehat{R}_{V_k}$ might deviate a lot from the mean.



Practical Issues in Cross-validation

How to decide the values for K and α?

- Large K
  - + The bias of the error estimate will be small
  - - The variance of the error estimate will be large (few validation pts)
  - - The computational time will be very large as well (many experiments)

- Small K
  - + The # experiments and, therefore, computation time are reduced
  - + The variance of the error estimate will be small (many validation pts)
  - - The bias of the error estimate will be large

Common choice: $K = 10$, $\alpha = 0.1$ ☺

Structural Risk Minimization

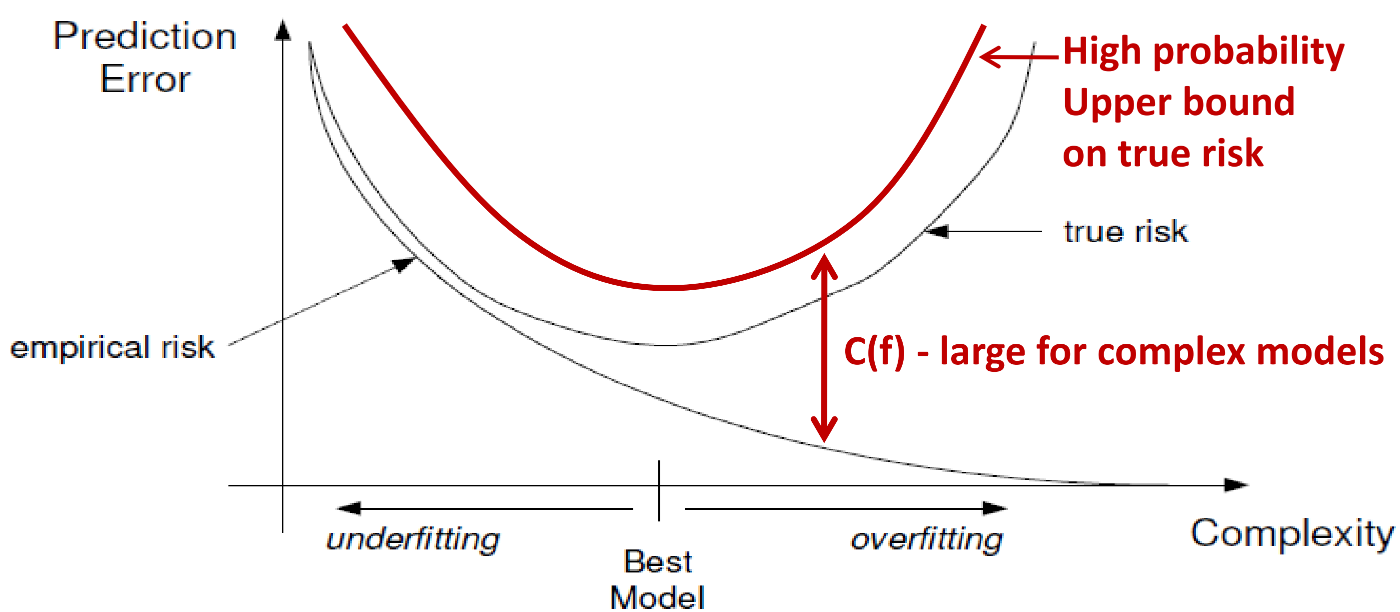
Penalize models using bound on **deviation of true and empirical risks**.

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

Bound on deviation from true risk

With high probability, $|R(f) - \widehat{R}_n(f)| \leq C(f) \quad \forall f \in \mathcal{F}$

Concentration bounds (later)



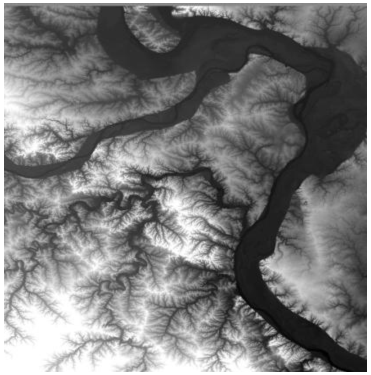
Structural Risk Minimization

Deviation bounds are typically pretty loose, for small sample sizes. In practice,

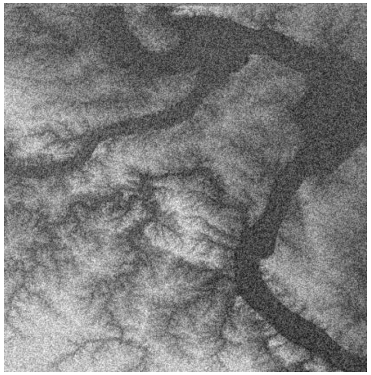
$$\widehat{f}_n = \arg\min_{f \in \mathcal{F}} \left\{ \widehat{R}_n(f) + \lambda C(f) \right\}$$

Choose by cross-validation!

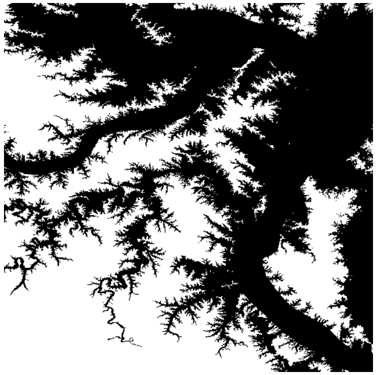
Problem: Identify flood plain from noisy satellite images



Noiseless image



Noisy image



True Flood plain
(elevation level > x)

Structural Risk Minimization

Deviation bounds are typically pretty loose, for small sample sizes. In practice,

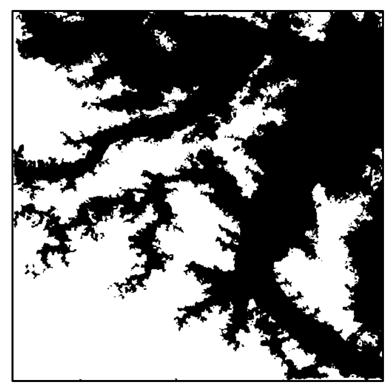
$$\widehat{f}_n = \arg\min_{f\in\mathcal{F}} \left\{ \widehat{R}_n(f) + \lambda C(f) \right\}$$

Choose by cross-validation!

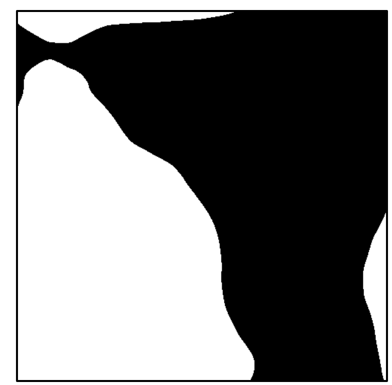
Problem: Identify flood plain from noisy satellite images

True Flood plain
(elevation level > x)

Zero penalty

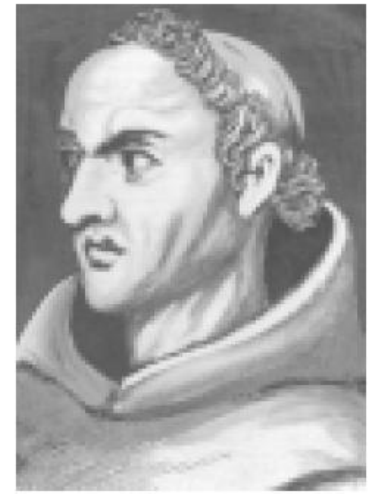


CV penalty



Theoretical penalty

Occam's Razor



William of Ockham (1285-1349) *Principle of Parsimony:*

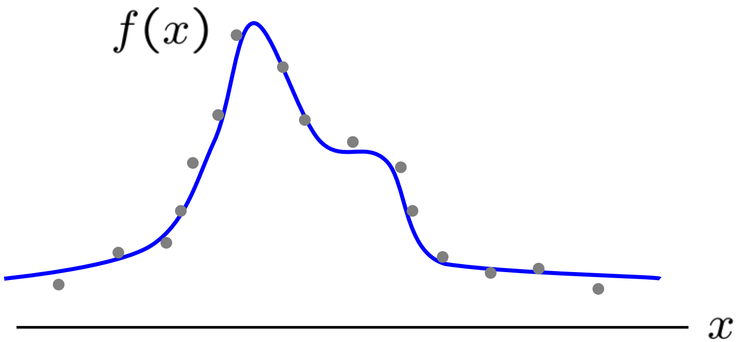
"One should not increase, beyond what is necessary, the number of entities required to explain anything."

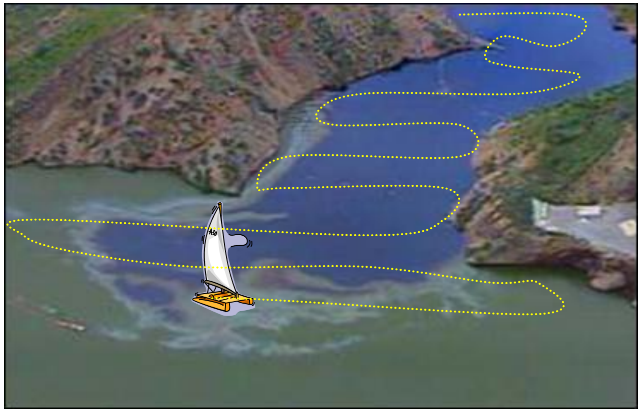
Alternatively, seek the simplest explanation.

Penalize complex models based on

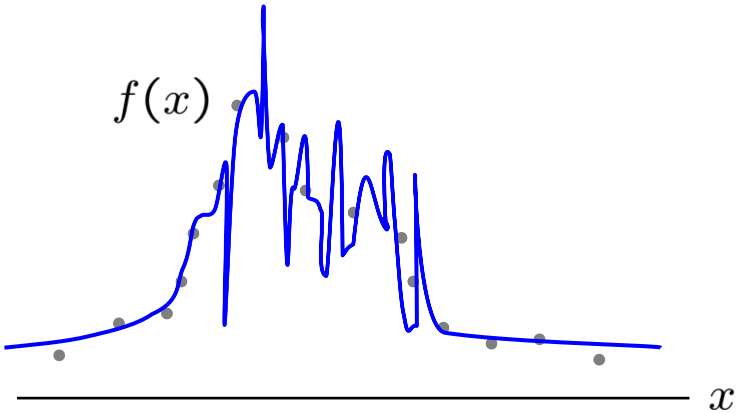
- Prior information (bias)
- Information Criterion (MDL, AIC, BIC)

Importance of Domain knowledge





Oil Spill Contamination



Distribution of photon arrivals

Compton Gamma-Ray Observatory Burst and Transient Source Experiment (BATSE)

Complexity Regularization

Penalize complex models using **prior knowledge**.

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

Cost of model (log prior)

Bayesian viewpoint:

prior probability of *f*, p*(f)* ≡ $e^{-C(f)}$

cost is small if *f* is highly probable, cost is large if *f* is improbable

ERM (empirical risk minimization) over a restricted class *F*
≡ uniform prior on *f* ϵ *F*, zero probability for other predictors

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

Complexity Regularization

Penalize complex models using **prior knowledge**.

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

Cost of model
(log prior)

Examples: MAP estimators
Regularized Linear Regression - Ridge Regression, Lasso

$$\widehat{\theta}_{\text{MAP}} = \arg\max_{\theta} \log p(D|\theta) + \log p(\theta)$$

$$\widehat{\beta}_{\text{MAP}} = \arg\min_{\beta} \sum_{i=1}^{n} (Y_i - X_i\beta)^2 + \lambda ||\beta||$$

Penalize models based on some norm of regression coefficients

How to choose tuning parameter λ? **Cross-validation**

Information Criteria – AIC, BIC

Penalize complex models based on their **information content**.

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

→ # bits needed to describe *f* (description length)

AIC (Akiake IC) C(*f*) = # parameters

Allows # parameters to be infinite as # training data n become large

BIC (Bayesian IC) C(*f*) = # parameters * log n

Penalizes complex models more heavily – limits complexity of models as # training data n become large

Information Criteria - MDL

Penalize complex models based on their **information content**.

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

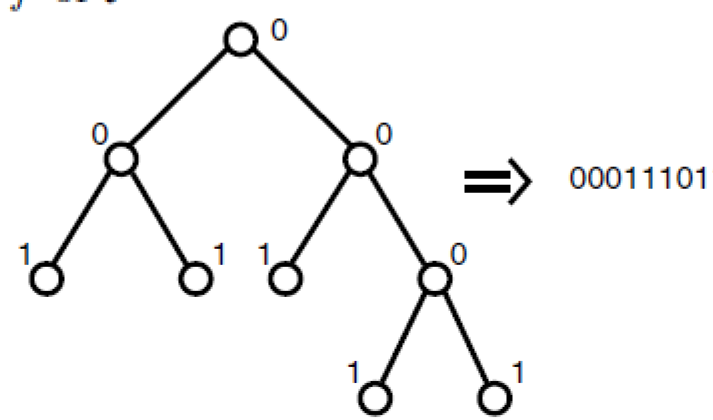
→ # bits needed to describe f (description length)

MDL (Minimum Description Length)

Example: Binary Decision trees $\mathcal{F}_k^T = \{\text{tree classifiers with } k \text{ leafs}\}$

$\mathcal{F}^T = \bigcup_{k \geq 1} \mathcal{F}_k^T$ prefix encode each element f of $\mathcal{F}^T$

$$C(f) = 3k - 1 \text{ bits}$$



k leaves => 2k – 1 nodes

2k – 1 bits to encode tree structure

\+ k bits to encode label of each leaf (0/1)

5 leaves => 9 bits to encode structure

Summary

True and Empirical Risk

Over-fitting

Approx err vs Estimation err, Bias vs Variance tradeoff

Model Selection, Estimating Generalization Error

- Hold-out, K-fold cross-validation
- Structural Risk Minimization
- Complexity Regularization
- Information Criteria – AIC, BIC, MDL