# Recitation

By Field Cady

February 4, 2010

# Statistics Review

## Statistics vs. Probability

- Probability : deriving properties of data from the distribution
- Statistics : deducing properties of the distribution from the data
  - In this sense, Machine Learning is a form of statistics
  - But “statistics” usually refers to classical statistics

## Classical Statistics Terminology

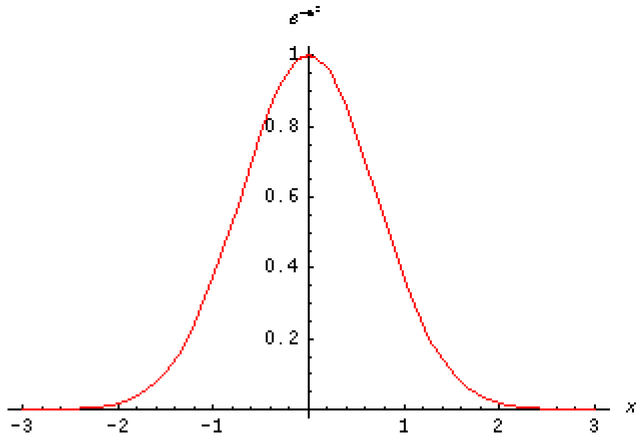
- Statistic : any function of your data
  - Can have logs, roots, arbitrary manipulation
  - Sample mean is just a function where you add and divide
- Estimator : a statistic that is intended to estimate some distribution parameter
  - Consistent estimator : enough data brings you arbitrarily close to the correct number
  - Unbiased estimator : expected value of the estimator is the actual parameter

## Estimators can be biased!

- Know X is Uniform(0,theta), theta unknown
- Data = x1, x2,…, xn
- Want to estimate theta
- Obvious estimator : max(xi)
  - Enough data points makes it close to theta
  - But it’s ALWAYS an underestimate
  - You can work out the math : max*(n+1)/n is unbiased and consistent

# Subtle Example

- Estimating a normal distribution: $\mu$ and $\sigma$
- $\mu^* = (1/n)\Sigma x_i$
- $(\sigma^*)^2 = (1/n)\Sigma(x_i - \mu^*)^2$
- How good are these estimators?



- Sample mean = consistent, unbiased estimator for μ
- Sample variance
  - Consistent – converges to actual variance
  - But it is <u>biased</u>!
- Imagine only 2 data points.
  - μ* exactly between them, used for calculating σ *
  - But using any other μ* would give higher σ *
  - The points won’t be exactly the same distance from μ
  - So we probably underestimate σ!

## MLE vs Bayes Summary

## MLE vs. Bayes

| | MLE | Bayesian |
|---|---|---|
| Basic Idea | Pick distribution to make data likely | Start with belief, and adjust for data |
| Examples | Basic, EM alg, logistic regression | Hierachical bayes models |
| Good | Data-based, often easy | Expert knowledge, mirrors thinking |
| Bad | Overfitting | Personal bias |
| Fixes | Cross-validation | Max-entropy prior |

## Training, Validation, and Testing Data

What the heck?

## Reasons we do it

- Recall : this is a special case of MLE
  - MLE is subject to overfitting
- The more we train a model, the better is becomes at predicting the *training* data
- But our goal is to predict other data that is i.i.d as training data
- Idea : use some training data for testing

## Cross-validation

- So we set aside some of the data as “validation data” to gauge how much to train
- But what if we randomly picked a bad validation set?
- Answer : break data into k sets, and use each set for validation separately

## Testing Data

- Problem : k-fold validation isn’t quite independent of validation data
  - Classifier implicitly trained on validation set, since validation data influences training cutoff
  - So validation data underestimates error
- Answer : have another chunk of data that plays no part at all in training classifier

## Neural Networks

## Motivation 1

- Recall linear regression :

$$unit\ output = \frac{1}{1 + exp(w_0 + \sum_i w_i x_i)}$$

- Decision boundary is a straight line
- They work decently, but we don’t want just linear boundaries
- Crazy idea for a hack : let’s wire a bunch of them together

## Motivation 2

- Human brains are fabulous
- They’re made of neurons, which have multiple inputs and a single output
- Let have an embarrassingly simple input->output function and model a brain

- Note : there’s work going on at using more physically realistic neural nets to get better performance

# Training Neural Nets

- MLE
  - Assume Gaussian errors
  - Minimize sum of squares
  - Backpropagation algorithm

$$W \leftarrow \arg\min_W \sum_l (y^l - \hat{f}(x^l))^2$$

- MAP :
  - Add penalty term lnP(W)
  - If using Gaussian prior :

$$W \leftarrow \arg\min_W \left[c\sum_i w_i^2\right] + \left[\sum_l (y^l - \hat{f}(x^l))^2\right]$$

## Regression

## Several Interpretations

- Fitting a straight line to data
- A part of logistic regression
- Predicting a continuous variable
  - Rather than discrete classification
  - Will never be “exactly” right, so error rate=1
  - Instead minimize average squared error

## Regularization

- General approach to problems where you penalize “extreme” solutions
- Examples
  - Regression : penalize huge parameters
  - Image denoising : penalize jitteriness
- Can be done in a principled way, but also just as a heuristic
- Can turn an under-determined set of equations into an optimization problem

## What Penalty Function

- Ridge Regression : squared
- Lasso Regression : absolute value

- Important point of contention : what should penalty terms look like?

## Aside : Is Least Squares Good?

- “Penalty” terms are ubiquitous
  - Least squares is just minimizing a penalty
  - Regularization penalty
  - Soon : k-means clustering minimizes squares
- Why do we use squares in the term?
  - Answer 1 : it’s easy
  - Answer 2 : it has a pretty theoretical interpretation if you use a Gaussian prior
- But is this really good?

# Least Squares

- Effect of squares : emphasize outliers
  - Point 10x farther away is 100x more important
- But shouldn’t we fit our model to the “normal” data points?
- Gaussians have “thin tails”
  - Few outliers, so big deviations very important

## Not Squares

- But real data often has “heavy tails”
  - Major outliers more common
- Exponentials, Laplacians, etc. have more outliers
- Minimizing sum of absolute values = Laplacian prior
- Absolute value penalty function, like in Lasso, makes outliers less important
- “Robust Statistics” deals with this

## Overfitting : Bias and Variance

## Risk and Error

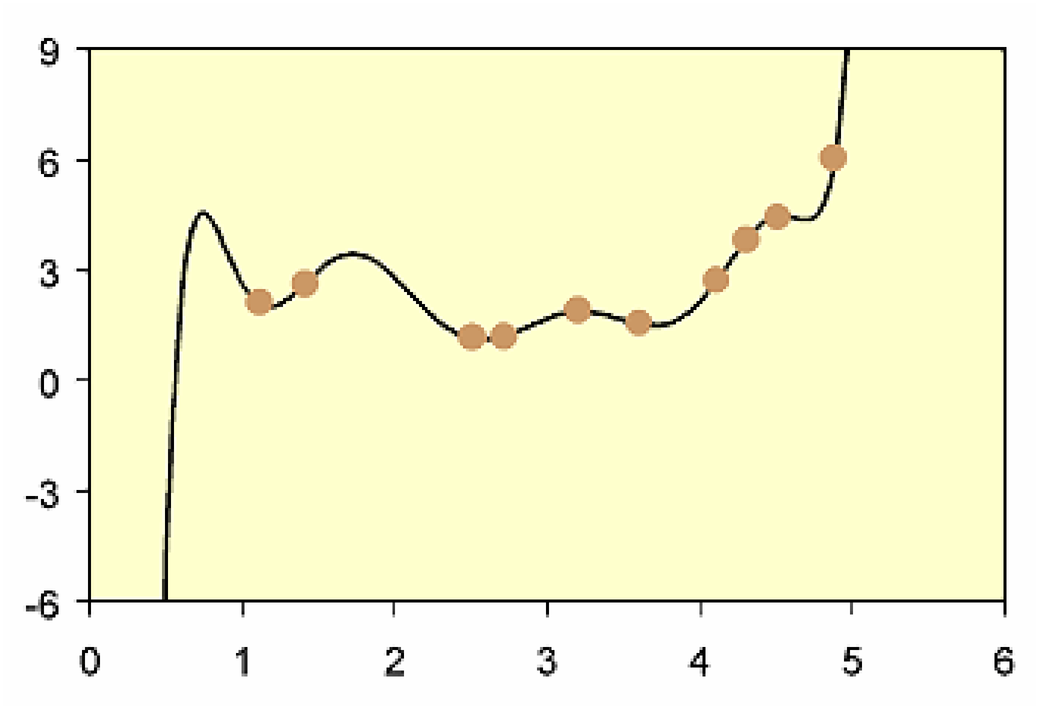
- Risk for classification $P(f(X) \neq Y)$
  - Probability data is misclassified
- Error for regression $\mathbb{E}[(f(X) - Y)^2]$
  - We'll never guess continuous values exactly, but a good regression function will be close
  - (Is this a good metric?  Food for thought…)

## True vs. Empirical

- Empirical risk/error : how badly we do on our training data
- True risk/error : how badly we’ll do on new data

- These are not the same!!!!

## Example



- Empirical error : 0
- True error : very high

## The Basic Conflict

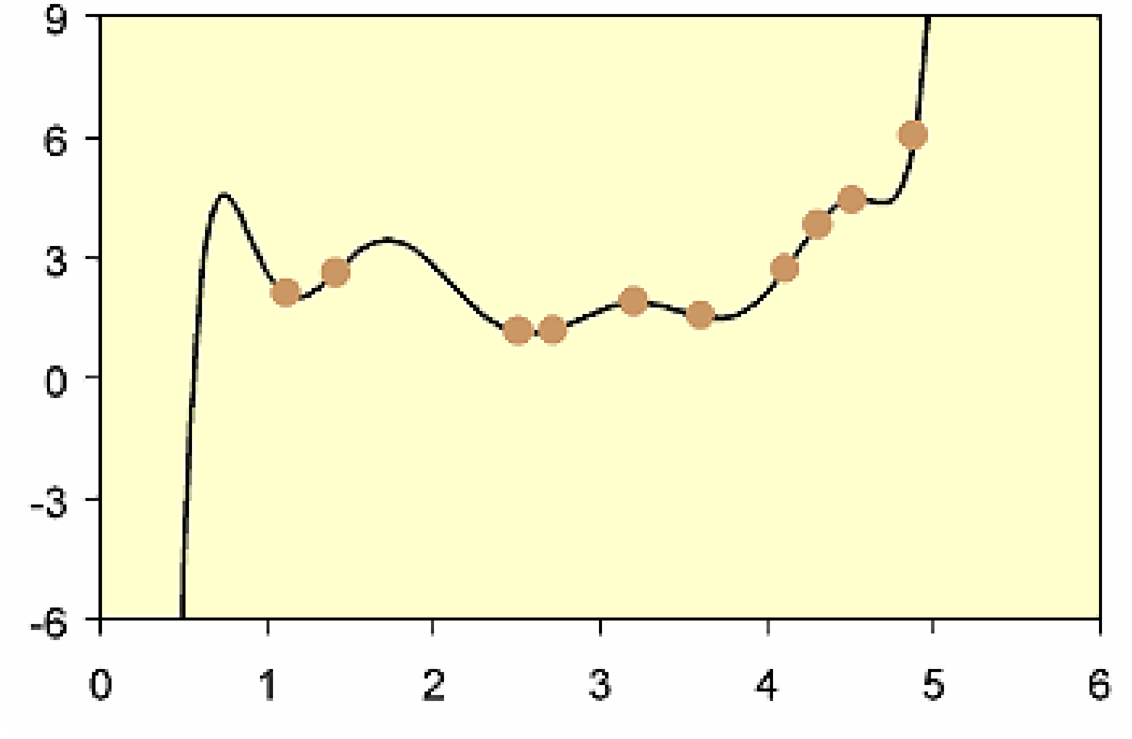
- Training picks a single classifier from a "family" of classifiers

- Observation 1 : A more expressive family can represent more general distributions

- But : A more expressive family is more likely to contain a bad distribution that fits that data really well

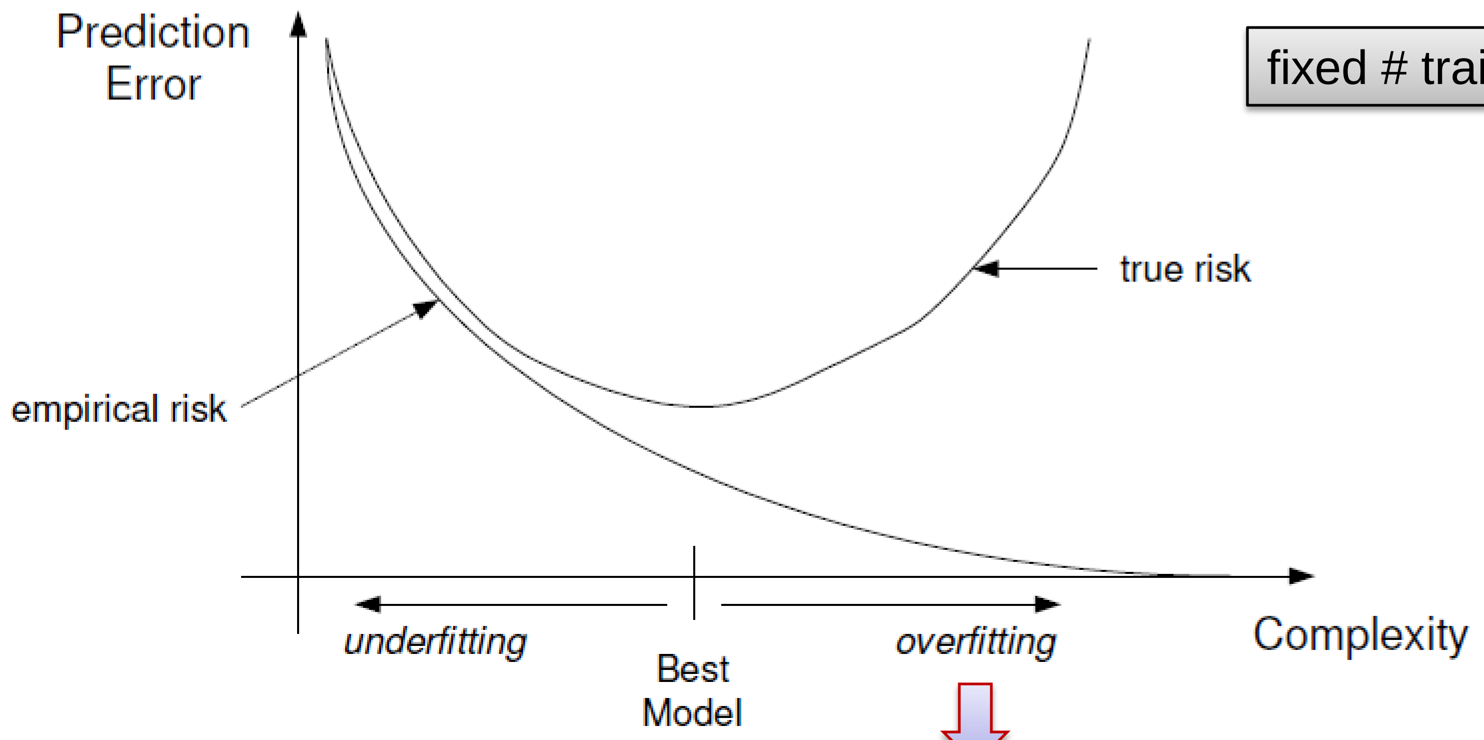
## Example

- Fitting a straight line is imperfect, but higher order polynomials are not always better…



# 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

# Excess Risk

- Some risk is inherent to the distribution
  - How close are we to this lower bound?

$$E\left[R(\widehat{f}_n)\right] - R^*$$

- Lower bound independent of our data
- For this discussion, we have exactly n points

- Recall that $\widehat{f}_n$ is just a statistic of our data; it is a random variable
- Distribution depends on n
- Then its performance will be too

$$\left.\begin{aligned} R(\widehat{f}_n) &= P_{XY}(\widehat{f}_n(X) \neq Y) \\ \\ R(\widehat{f}_n) &= \mathbb{E}_{XY}[(\widehat{f}_n(X) - Y)^2] \end{aligned}\right\} \quad \widehat{f}_n \text{ depends on random training dataset}$$

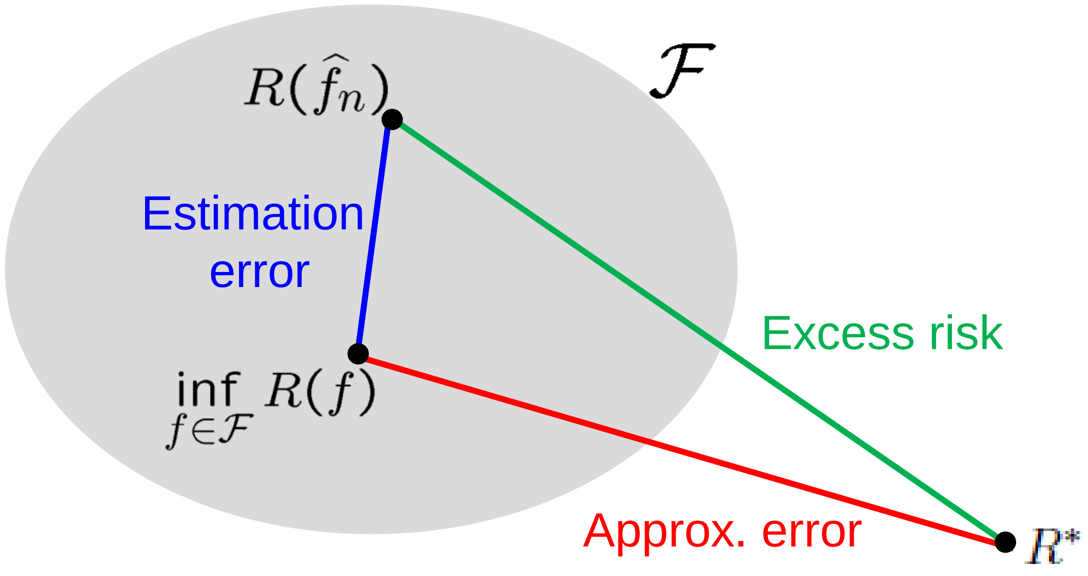
# Behavior of True Risk

Want predictor based on training da$\widehat{f}_n$ to be as good as optimal pre $f^*$ctor

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

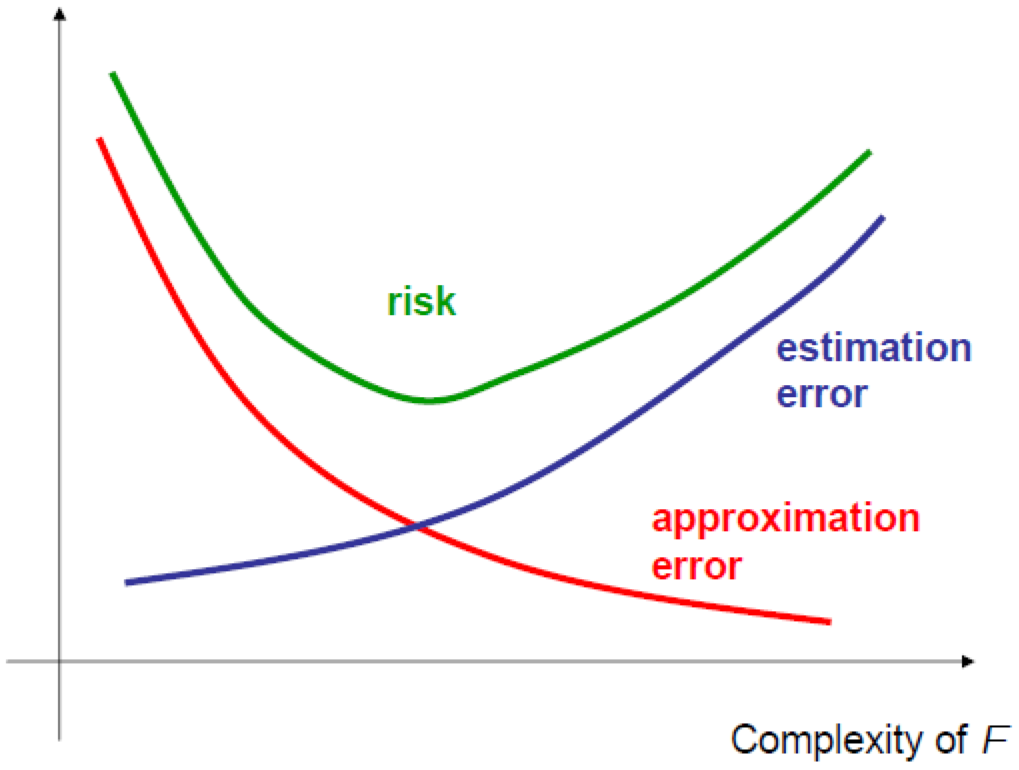
$$\text{finite sample size + noise} \longleftarrow \text{Due to randomness of training data}$$

Due to restriction of model class



## Behavior of True Risk

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



# Bias – Variance Tradeoff

$$\begin{aligned}
\mathbb{E}_{X,Y}\left[(\mathbb{E}_D[\widehat{f}_n(X)] - Y)^2\right] &= \mathbb{E}_{X,Y}\left[(\mathbb{E}_D[\widehat{f}_n(X)] - f^*(X) - \epsilon)^2\right] \\
&= \mathbb{E}_{X,Y}\Big[(\mathbb{E}_D[\widehat{f}_n(X)] - f^*(X))^2 + \epsilon^2 \\
&\qquad - 2\epsilon(\mathbb{E}_D[\widehat{f}_n(X)] - f^*(X))\Big] \\
&= \mathbb{E}_{X,Y}\left[(\mathbb{E}_D[\widehat{f}_n(X)] - f^*(X))^2\right] + \mathbb{E}_{X,Y}\left[\epsilon^2\right] \\
&\qquad \cancel{-2\mathbb{E}_{X,Y}\left[\epsilon(\mathbb{E}_D[\widehat{f}_n(X)] - f^*(X))\right]}
\end{aligned}$$

**0** since noise is independent and zero mean

$$= \underbrace{\mathbb{E}_{X,Y}\left[(\mathbb{E}_D[\widehat{f}_n(X)] - f^*(X))^2\right]}_{\text{bias^2 – how our predictor differs from the optimal}} + \underbrace{\mathbb{E}_{X,Y}\left[\epsilon^2\right]}_{\text{noise variance}}$$

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

## Bottom Line

- Approximation error
  - From using a restrictive family of classifiers
- Estimation error
  - From not using best classifier in family
- Approximation and Estimation error combine to make Bias
- Variance
  - inherent to distribution

## Model Selection

- Goal : minimize generalization error
- Philosophy : Occam’s Razor, K.I.S.S, etc.
- Hold-out approach
  - Have training and validation data.
  - Pick model that does best on validation data
- Fancier Approaches
  - Penalize models that are likely to overfit
  - Several ways to do this

# Hold-out method

Goal : pick model to minimize generalization (True) error.

Idea : train models on some data, judge them based on the rest

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:

$$\widehat{f}_\lambda = \arg\min_{f \in \mathcal{F}_\lambda} \underbrace{\widehat{R}_T(f)}_{\text{Evaluated on training dataset } D_T}$$

Evaluated on training dataset *D*T

# Hold-out method

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

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

$\xrightarrow{}$ Evaluated on validation dataset *Dv*

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:

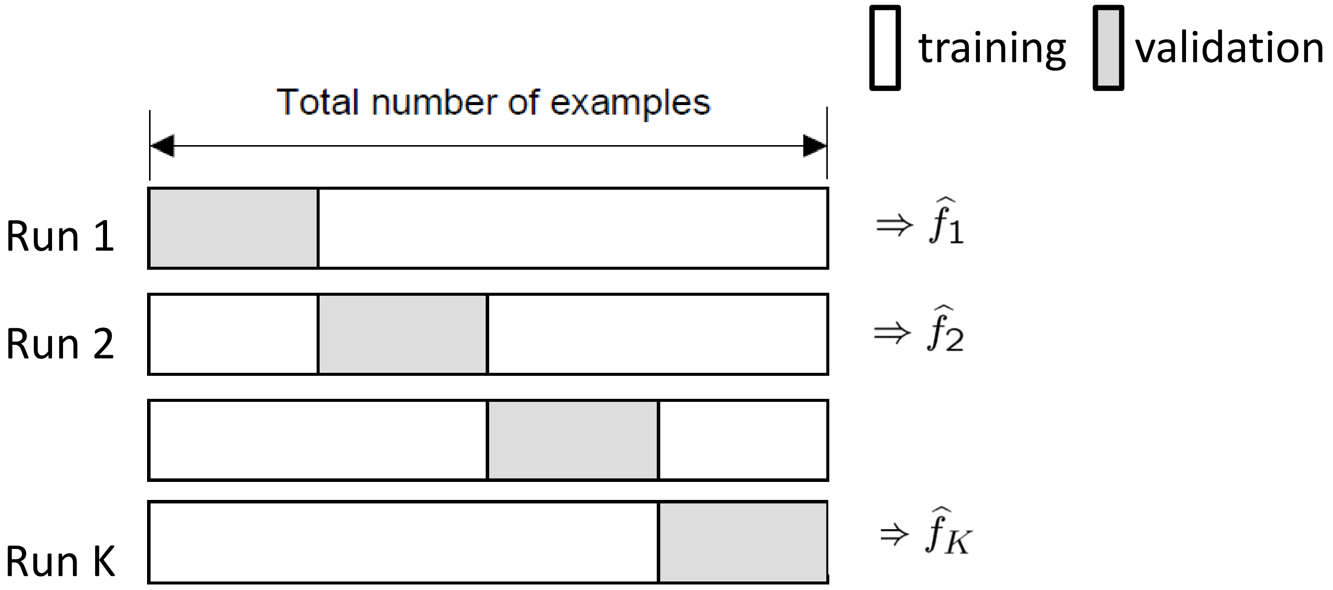
- May not have enough data to afford setting one subset aside for getting a sense of generalization abilities
- Validation error may be misleading (bad estimate of generalization error) if we get an “unfortunate” split

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.



▯ training
▨ validation
Total number of examples
Run 1
Run 2
Run K
$\Rightarrow \widehat{f}_1$
$\Rightarrow \widehat{f}_2$
$\Rightarrow \widehat{f}_K$

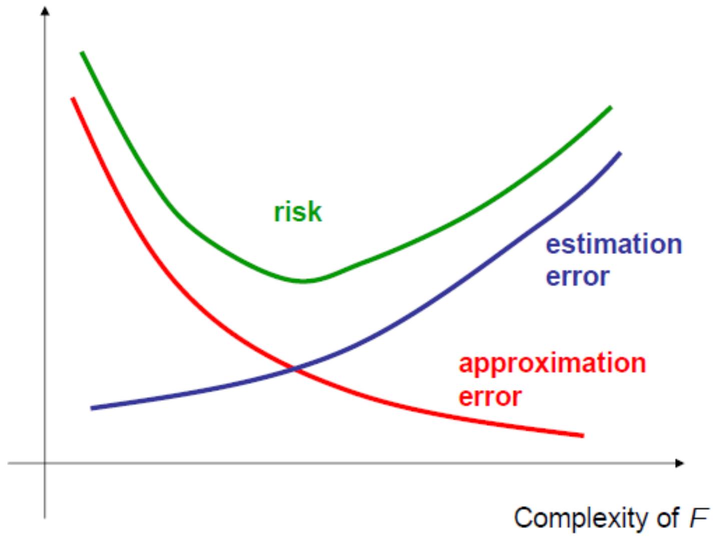
For each model family, train K models and average their errors
Pick the model family that performed best
Re-train this family on all of the data

## Fancier Approaches

- Minimizing empirical risk is, um, risky
- Add a “penalty term” of some sort to the empirical risk
- This term should indicated how prone a model is to overfitting

# Structural Risk Minimization

Recall : Best model minimizes true risk



Problem : We don't know true error

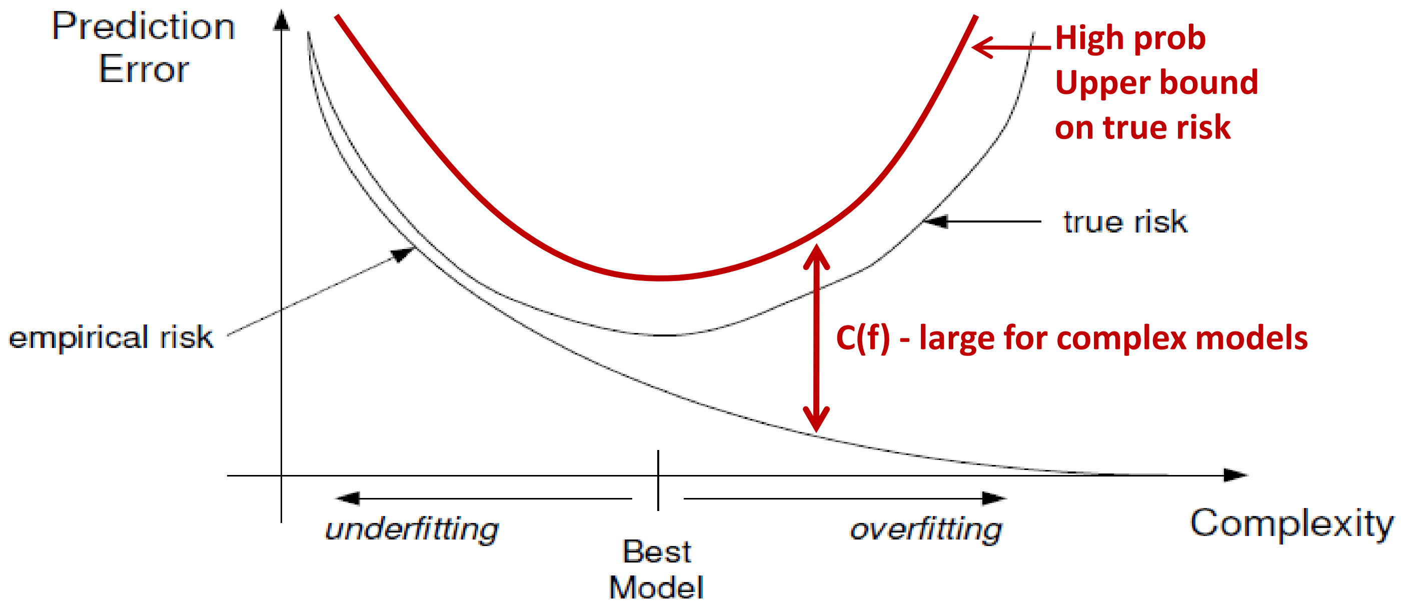
Idea : Bound the true error as a function of complexity and minimize that

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

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



In practice, theoretical bounds are way too high, so we scale them

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

Choose by cross-validation!

# Complexity Regularization

Penalize complex models using **prior knowledge**.

$$\widehat{f}_n \quad = \quad \arg\min_{f\in\mathcal{F}} \left\{ \widehat{R}_n(f) + \underset{\hookrightarrow \text{Cost of model (log prior)}}{C(f)} \right\}$$

Bayesian viewpoint:

prior probability of *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*, e.g. linear classifiers, ≡ uniform prior on *f* ϵ *F*, zero probability for other predictors

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

## Note the common pattern

- Start with MLE
  - Minimize –(probability of data)
  - But this overfits
- So we add a penalty term for “bad-looking” models
  - Least squares
  - Model complexity
- This has a Bayesian interpretation

# Information Criteria

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

$$\widehat{f}_n = \arg\min_{f \in \mathcal{F}} \left\{ \widehat{R}_n(f) + \underbrace{C(f)}_{\text{\# bits needed to describe } f \text{ (description length)}} \right\}$$

MDL (Minimum Description Length)

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

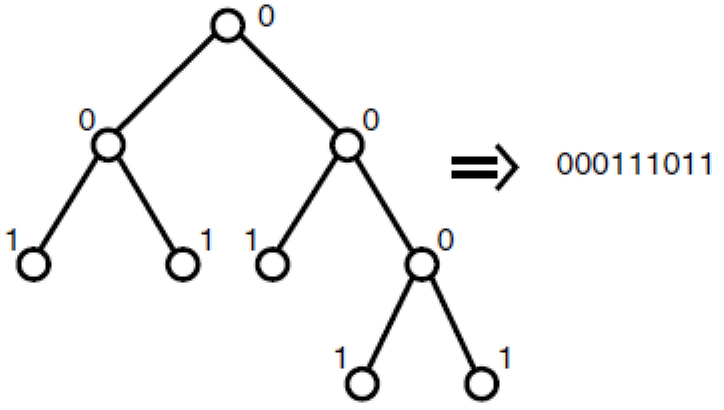
$\mathcal{F}^T = \bigcup_{k \geq 1} \mathcal{F}_k^T$ prefix encode each element $f$ of $\mathcal{F}^T$

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

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