Data Mining

CS57300
Purdue University

March 8, 2018

Real-World Example

- In what follows we will have a crash-course on Convolutional Neural Networks
- Goal is to use them in a real-life experiment
- We will revisit CNNs later

Convolutional Neural Networks (Motivation)

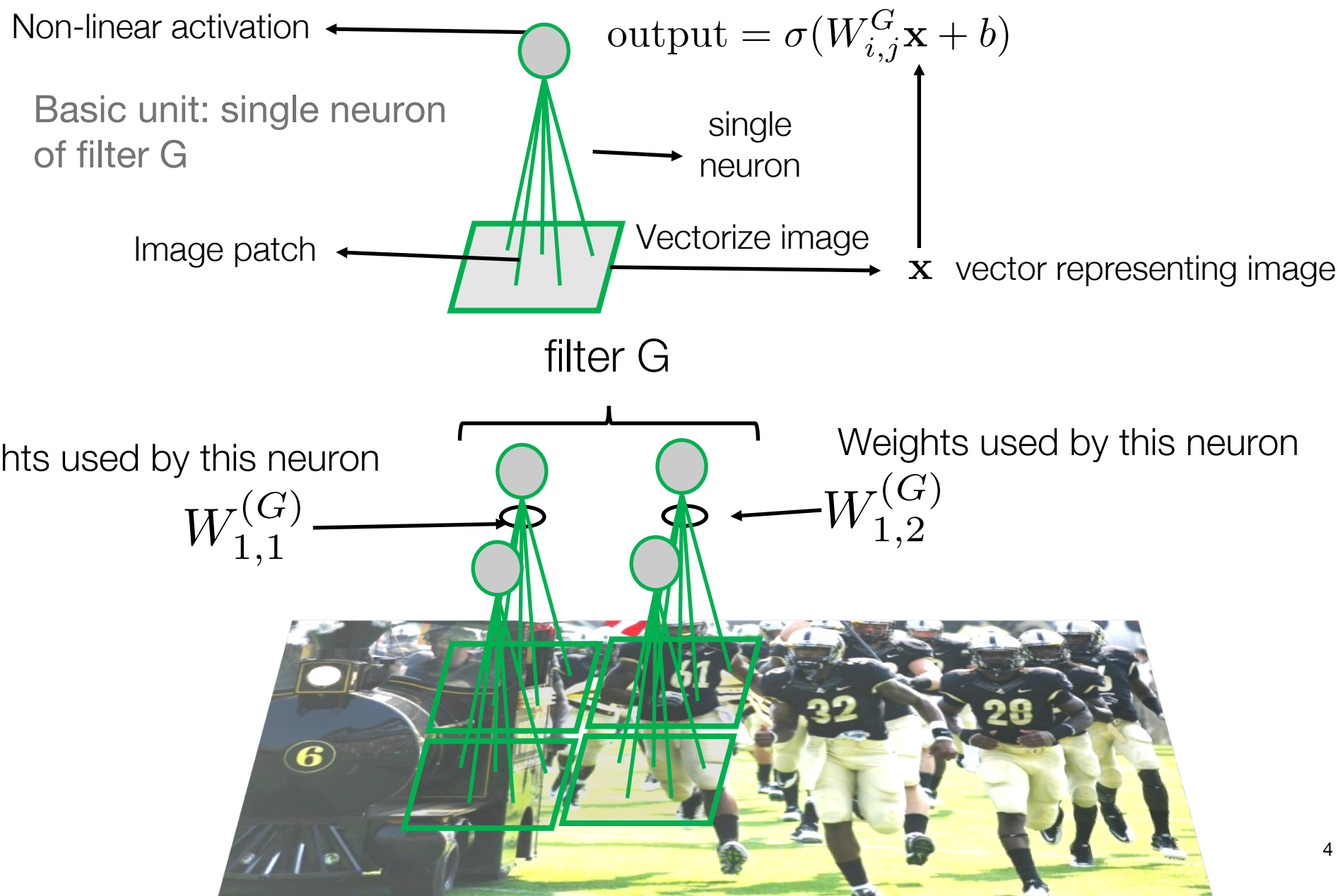
- Feedforward networks use image as input as a vector

- From image to vector… hard to account for spatial correlations in the vector representation (which pixels are next to each other?)

- Even harder to account for location and colors

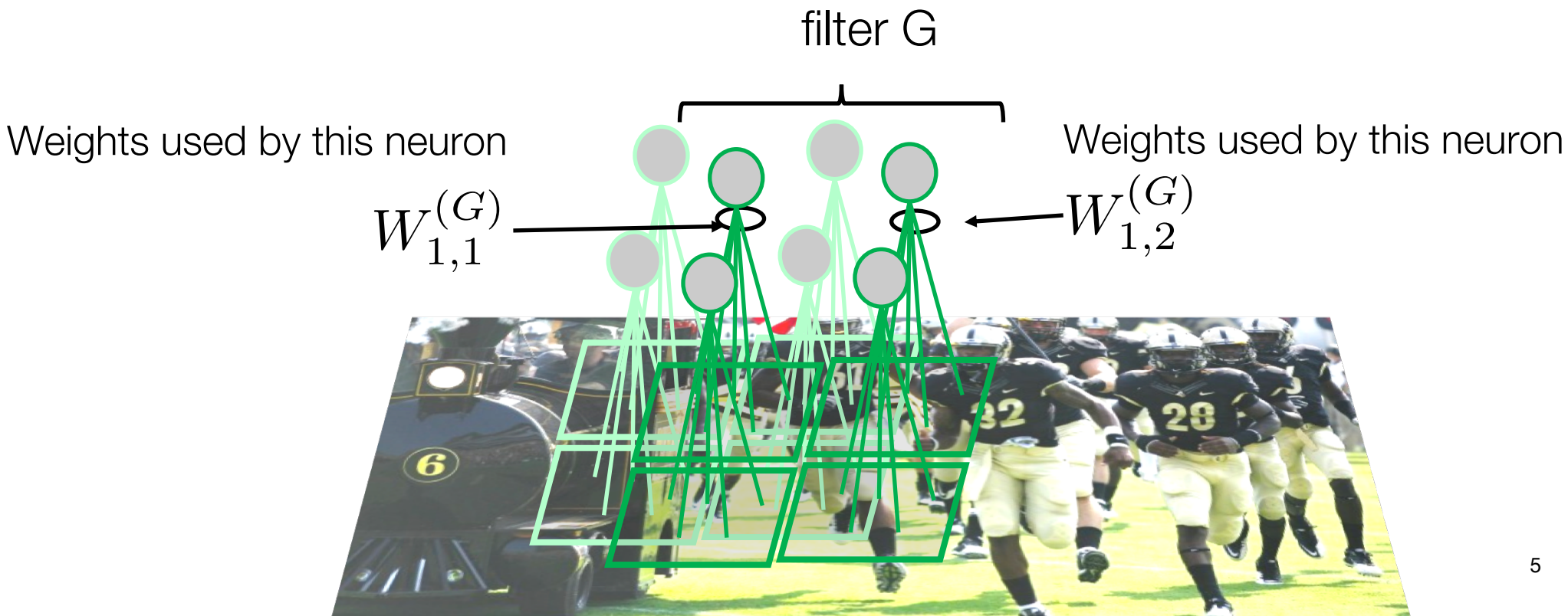
Convolutional Neural Network (CNN)

Transforms large image patch into one output



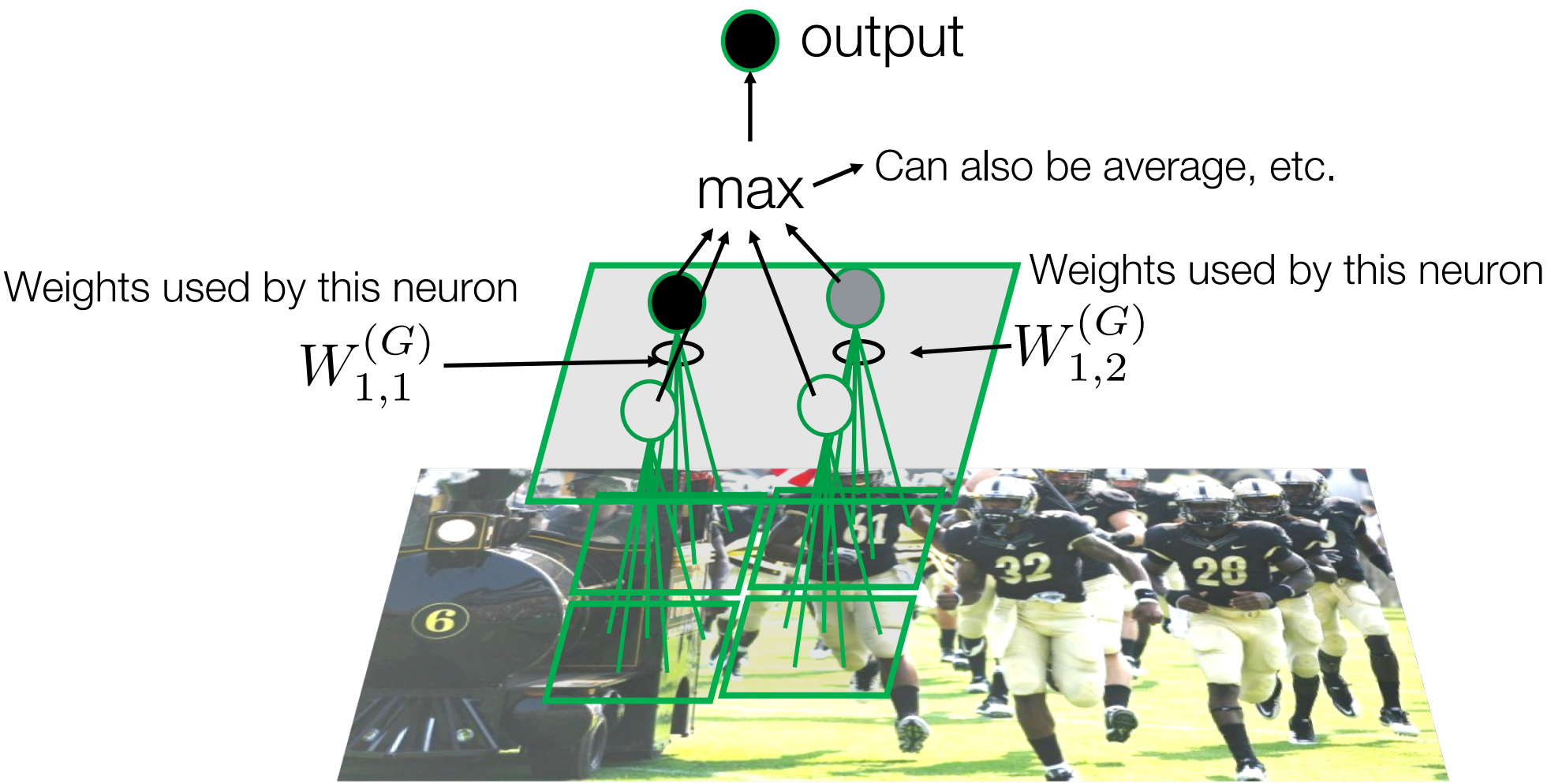
Convolutional Neural Network (CNN)

Cover the rest of the image by sliding the filter



Pooling

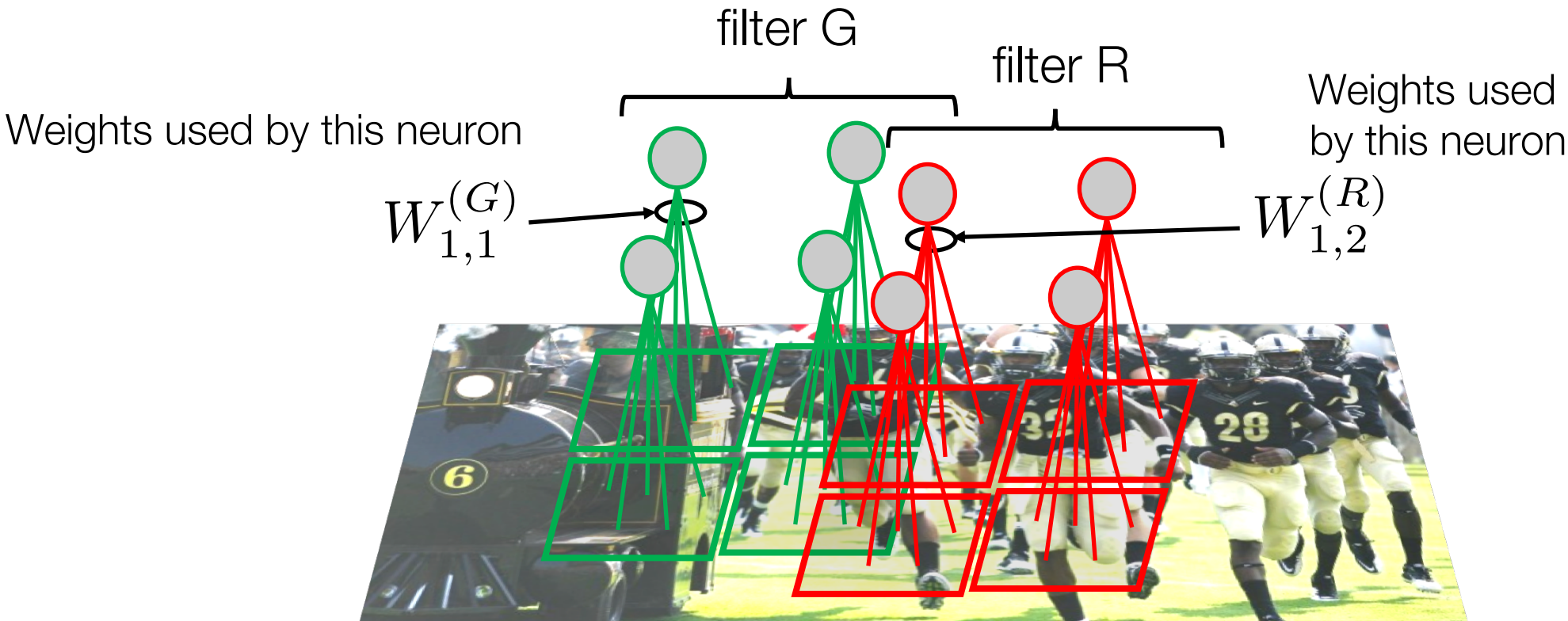
- Max pooling is a way to get a single output out of a filter



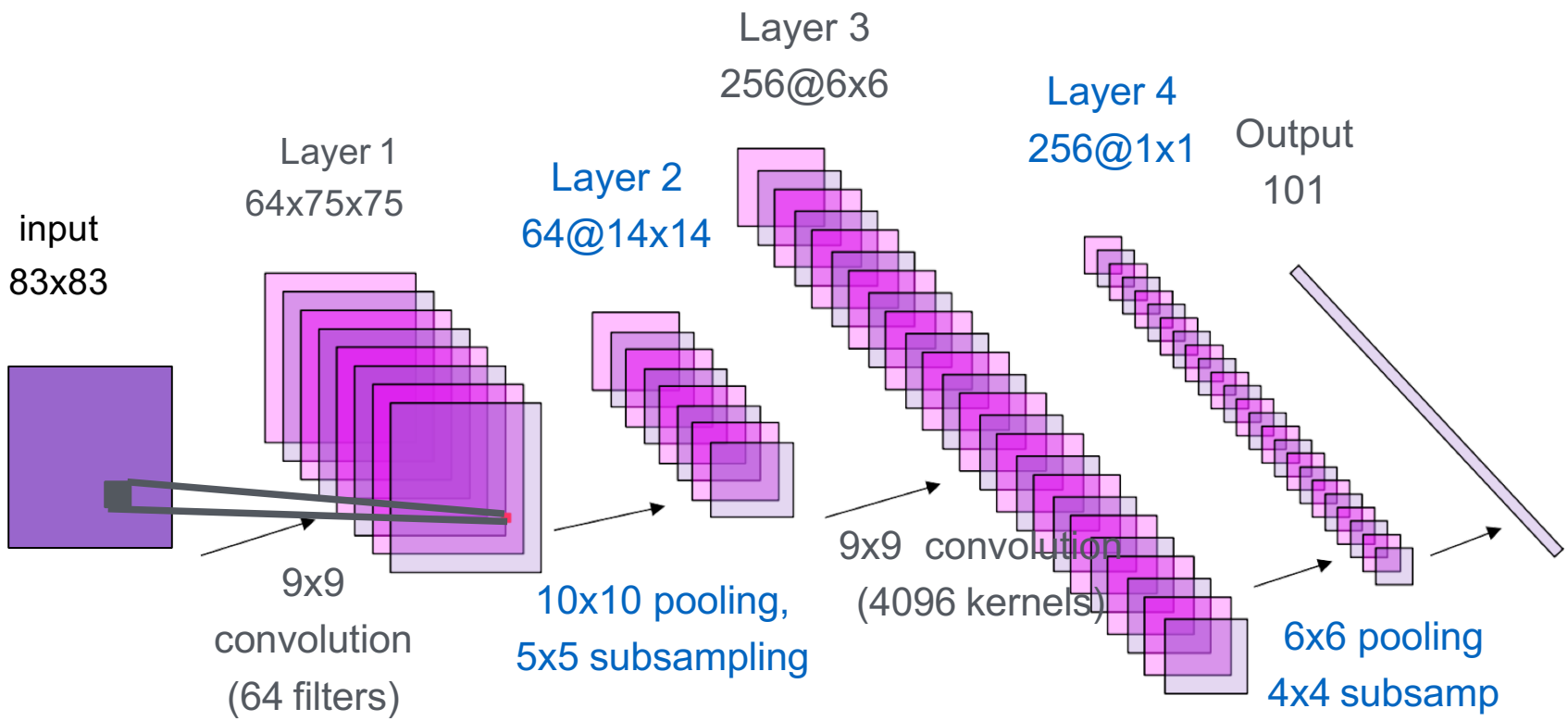
How to represent the image with different colors?

Convolutional Neural Network (CNN)

- We can use other filters
- For instance, one for each color
- We can also use more than one filter for each color



Convolutional Network (ConvNet)

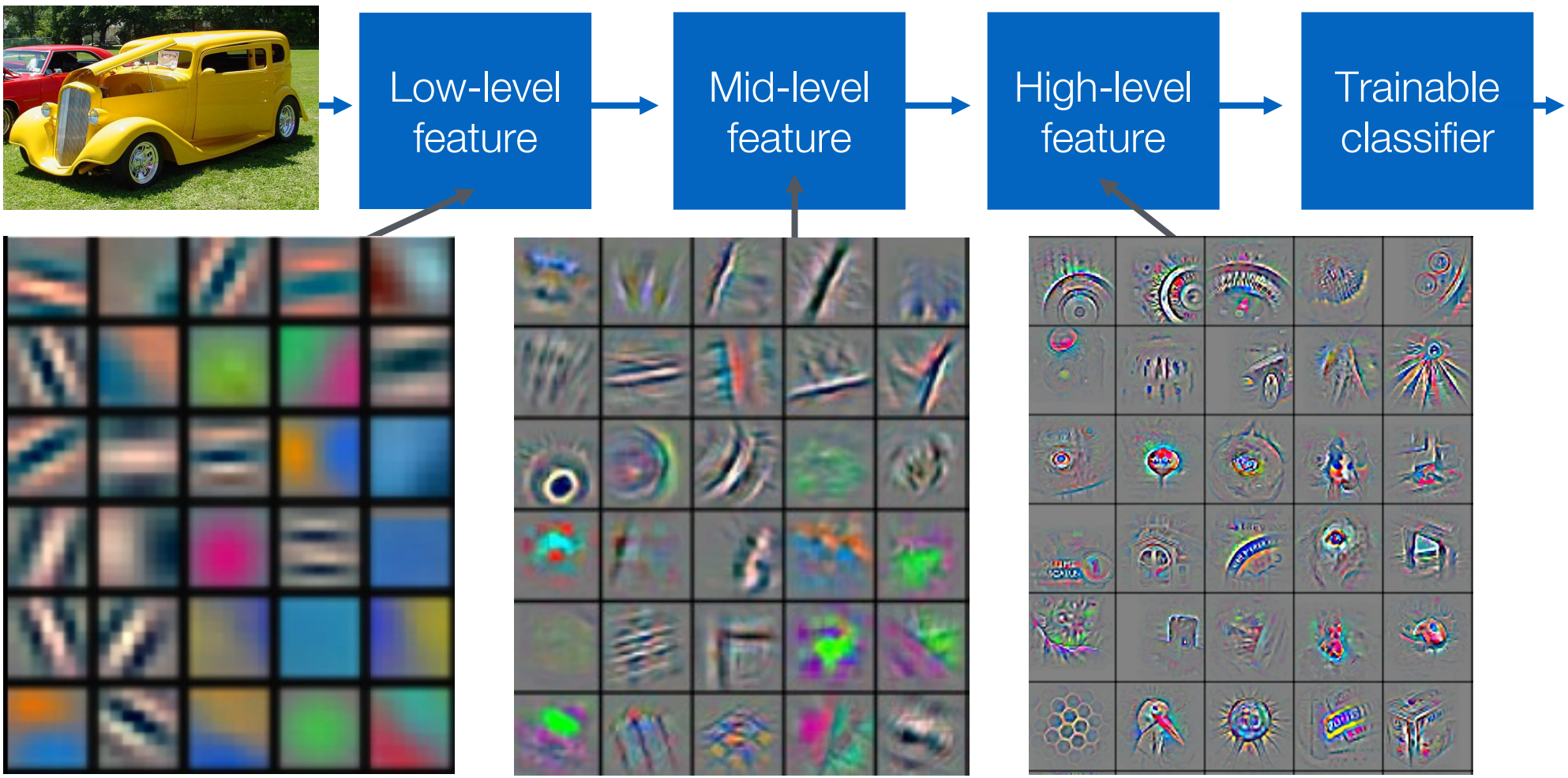


- Non-Linearity: sigmoid, rectified linear units (ReLU)
- Pooling: max, average,…
- Training: Image labels

Original: Y. LeCun, Facebook, modified by Ribeiro

Deep learning = learning hierarchical representations

It's deep if it has more than one stage of non-linear feature transformation



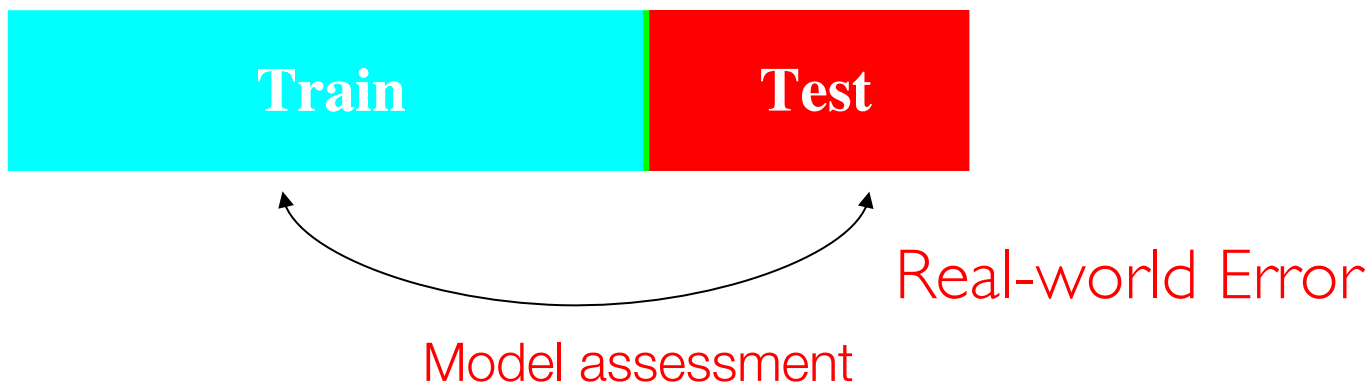
Feature visualization of convolutional net trained on ImageNet from [Zeiler & Fergus 2013]

Credit: Y. LeCun, Facebook

Today's Goal

- Predictive Model Assessment

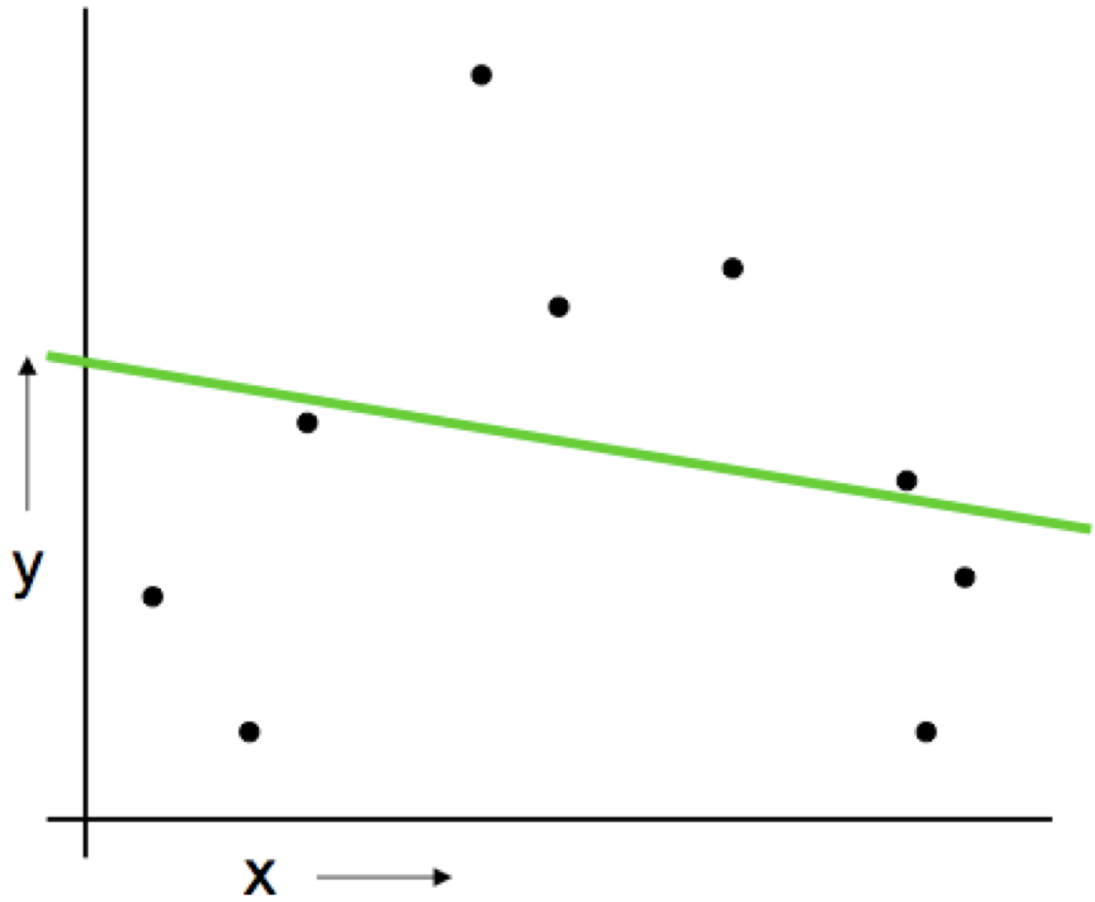
In Training Phase There is Often a Better Model



Data-rich Approach

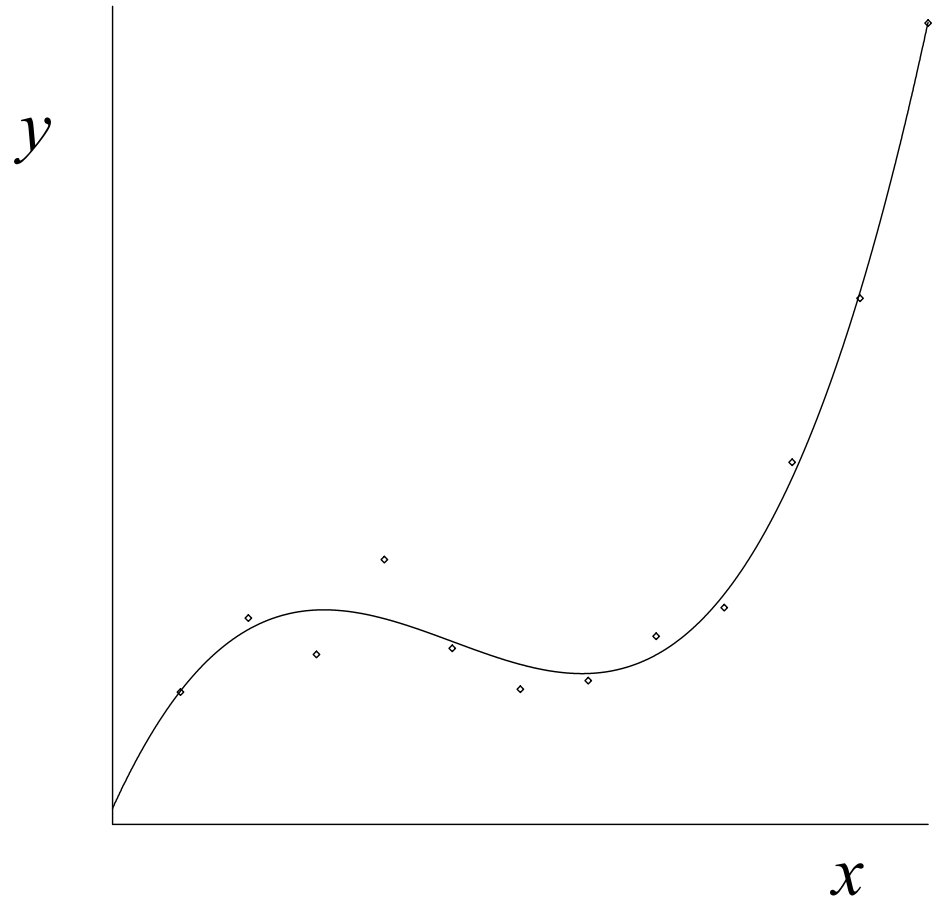
Linear Regression

n observations



$$y = ax + b$$

Cubic Regression



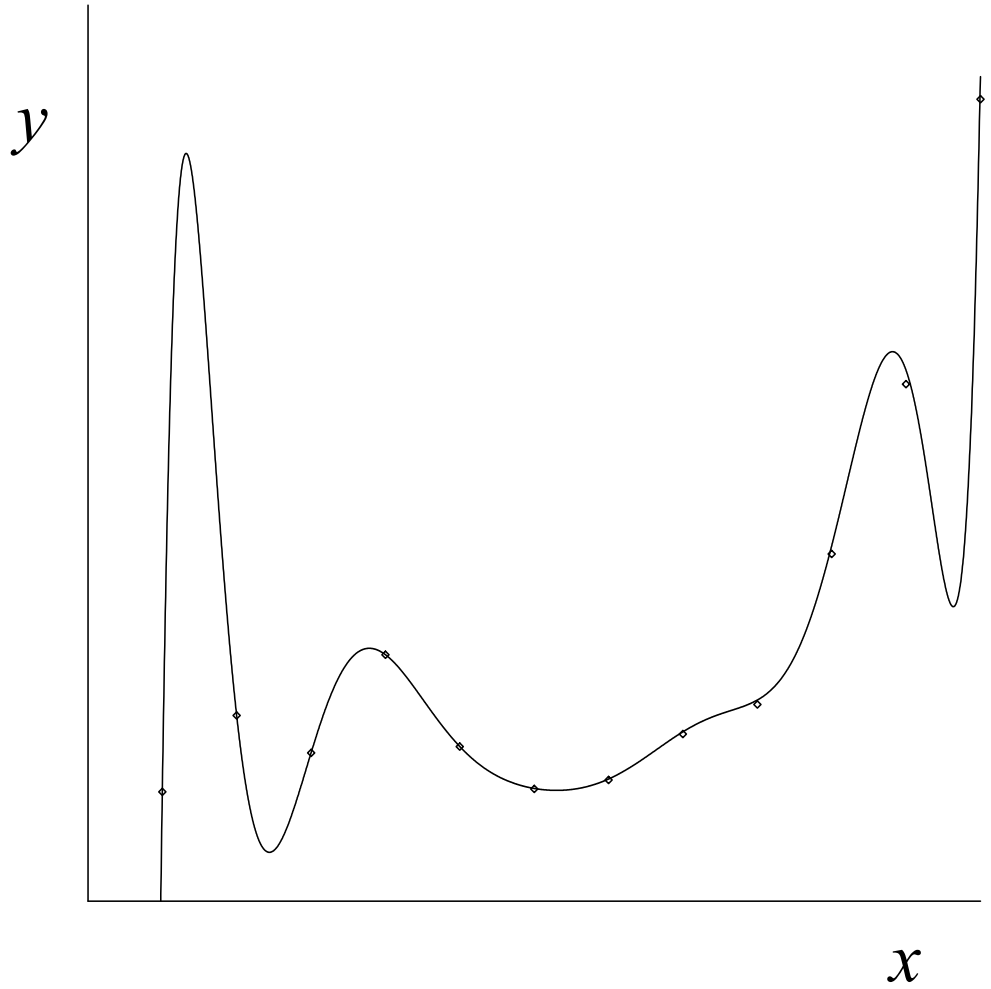
Better

n observations

$$y = ax^3 + bx^2 + cx + d$$

Degree *n*-1 Polynomial

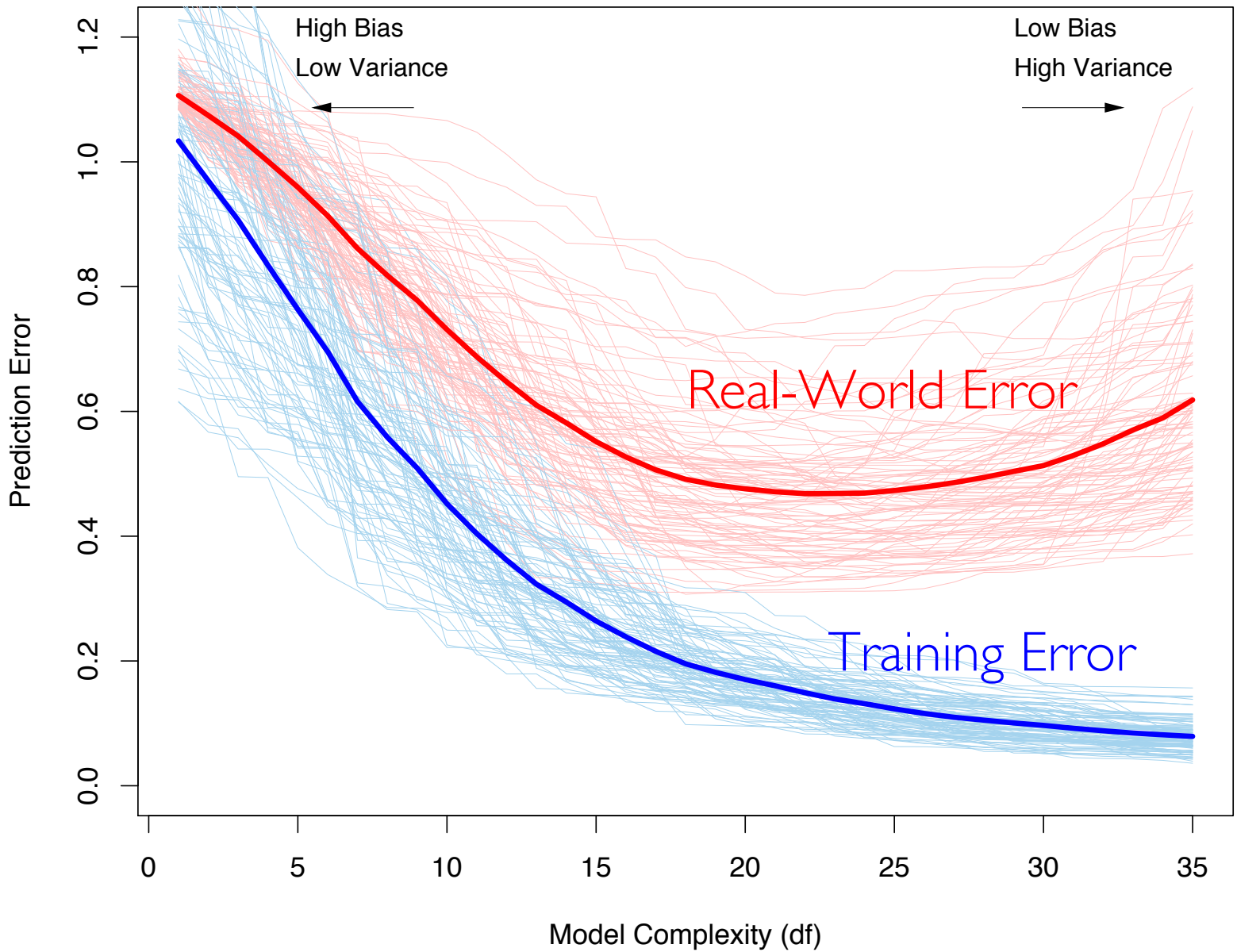
n observations



Even better?

$$y = ax^{n-1} + bx^{n-2} + \ldots$$

The Problem of Overfitting



The Elements of Statistical Learning
Jerome H. Friedman, Robert Tibshirani, and Trevor Hastie

Model Selection and Assessment

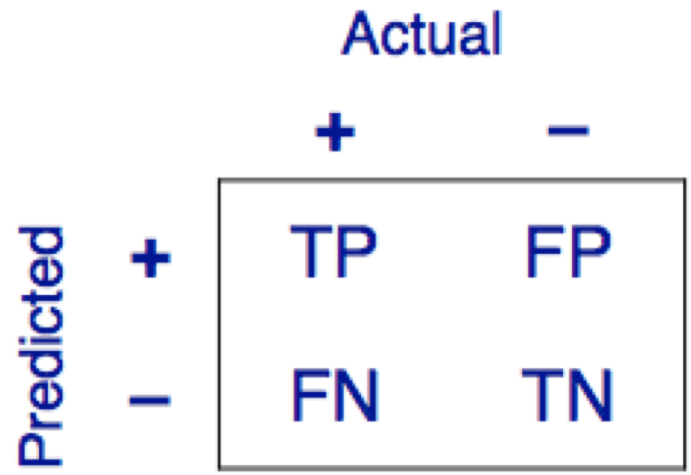
- Model Selection: Estimating performances of different models to select the best one

- Model Assessment: Having chosen a model, estimating the prediction error on new data

Testing Classification Accuracy

Classification Error

- True positive (**TP**):
  positive prediction that is correct
- True negative (**TN**):
  negative prediction that is correct
- False positive (**FP**):
  positive prediction that is incorrect
- False negative (**FN**):
  negative prediction that is incorrect

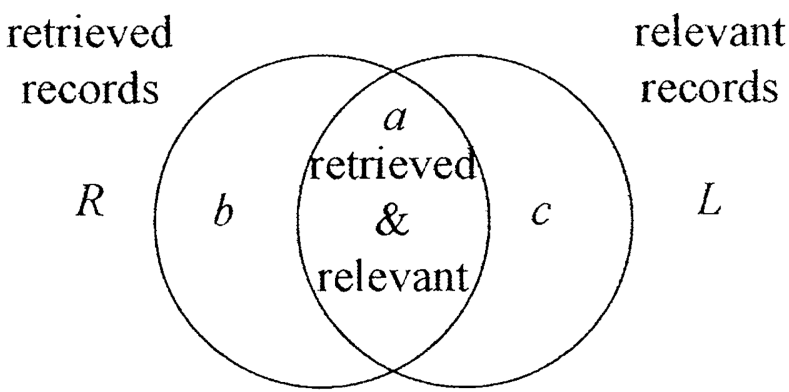


- Accuracy = (TP + TN) / (TP + TN + FP + FN) *% predictions that are correct*
- Misclassification = (FP+FN) / (TP+TN+FP+FN) *% predictions that are incorrect* i.e., zero-one loss
- Recall (Sensitivity) = TP / (TP + FN) *% positive instances that are predicted positive*
- Precision = TP / (TP + FP) *% positive predictions that are correct*
- Specificity = TN / (TN + FP) *% negative instances that are predicted negative*
- F1 = 2 (P·R) / (P+R) *harmonic mean of precision and recall*

Relationship to Type I and Type II errors

	Condition Positive	Condition Negative
Test Positive	True Positive	False Positive (Type I error)
Test Negative	False Negative (Type II error)	True Negative

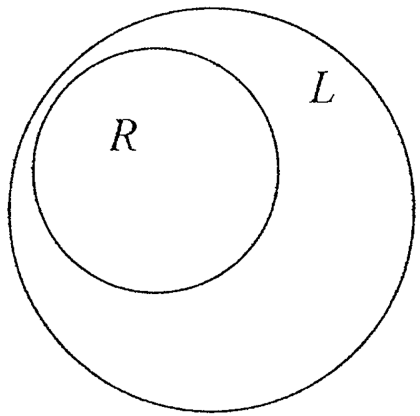
Precision and Recall



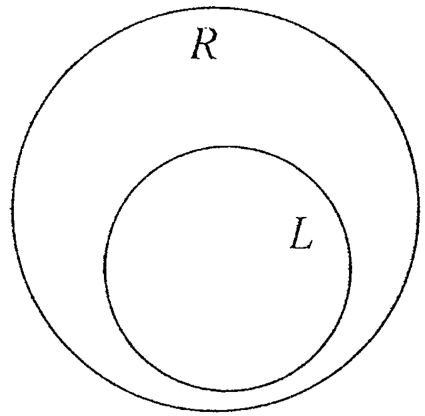
$$\text{Precision: } \frac{a}{a + b}$$

$$\text{Recall: } \frac{a}{a + c}$$

(a) Precision and recall



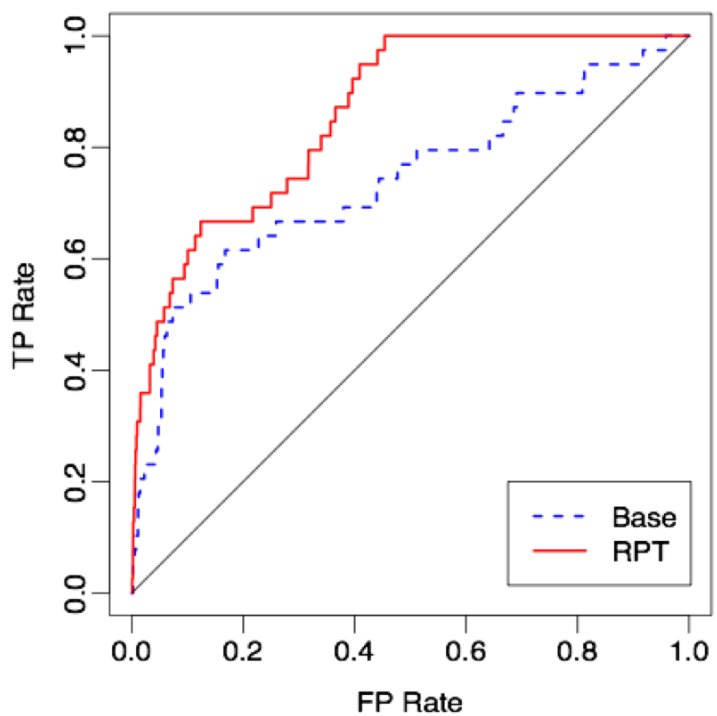
(b) Precision = 1



(c) Recall = 1

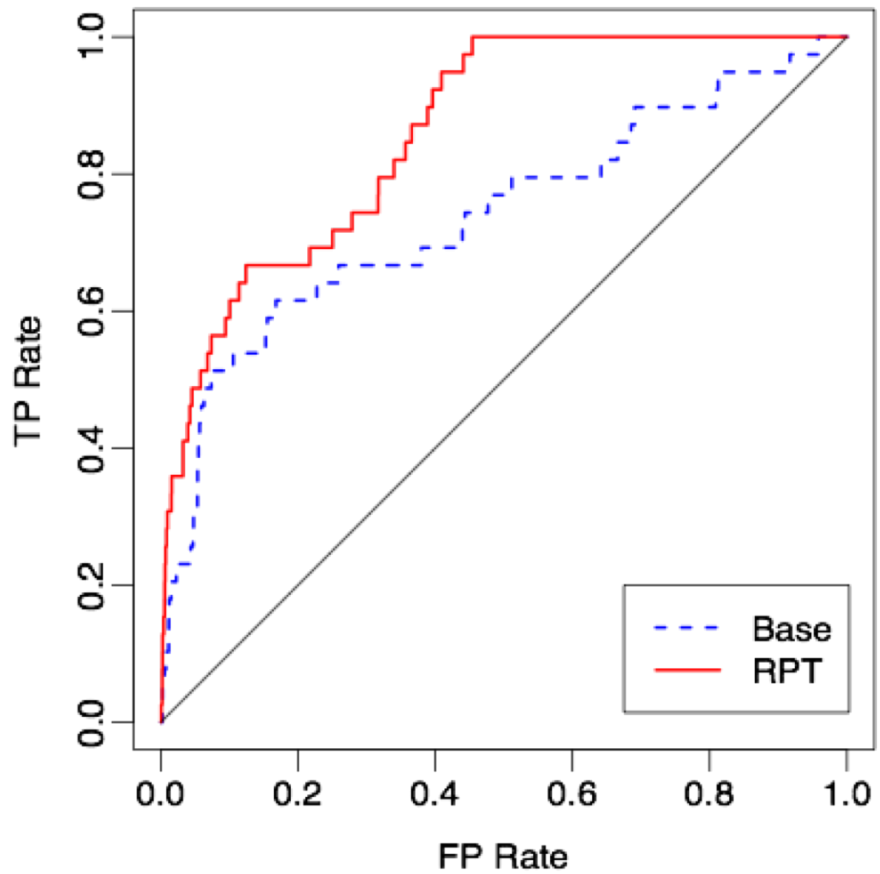
More score functions

- Absolute loss: $\frac{1}{n}\sum_{i=1}^{n}|p(y_i=t_i)-1.0|$ where t is true label
- Squared loss: $\frac{1}{n}\sum_{i=1}^{n}[p(y_i=t_i)-1.0]^2$ where t is true label
- Likelihood/conditional likelihood: $\prod_{i=1}^{n}p(y_i=t_i)$ where t is true label
- **Area under the ROC curve**
  - *Receiver Operating Characteristic* (ROC) curve
  - For classifier that a classification threshold can be defined (e.g. SVMs [vary hyperplane offset b],..)
  - Plots the true positive rate against the false positive rate for different classification thresholds



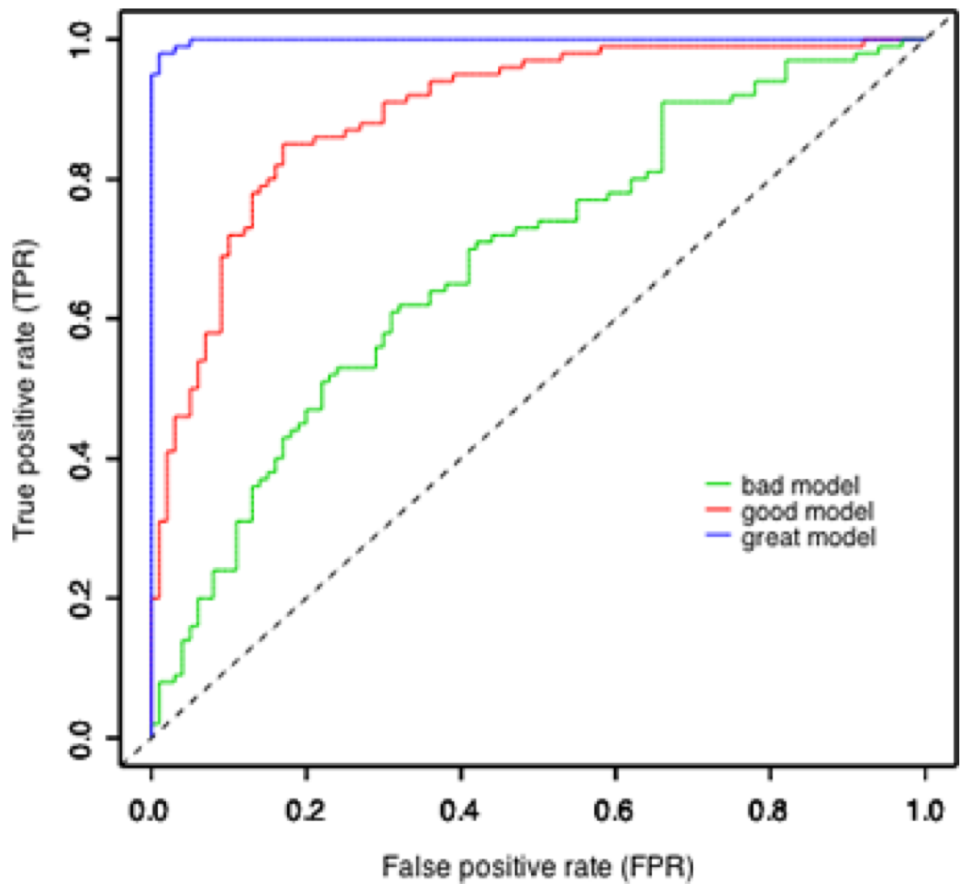
ROC curves

- Evaluates performance over varying costs and class distributions
  - Can summarize with area under the curve (AUC)
  - AUC of 0.5 is random
  - AUC of 1.0 is perfect



ROC: Good and Bad Models

- From bad to great (depends on application)



How to compute ROC curve

P(Y)	True class
0.94	+
0.84	-
0.67	+
0.58	+
0.51	+
0.42	+
0.16	-
0.1	+
0.07	-

P(Y)	True class	Predict class
0.94	+	+
0.84	-	-
0.67	+	-
0.58	-	-
0.51	+	-
0.42	+	-
0.16	-	-
0.1	-	-
0.07	-	-

FPR = 0/5 TPR = 1/4

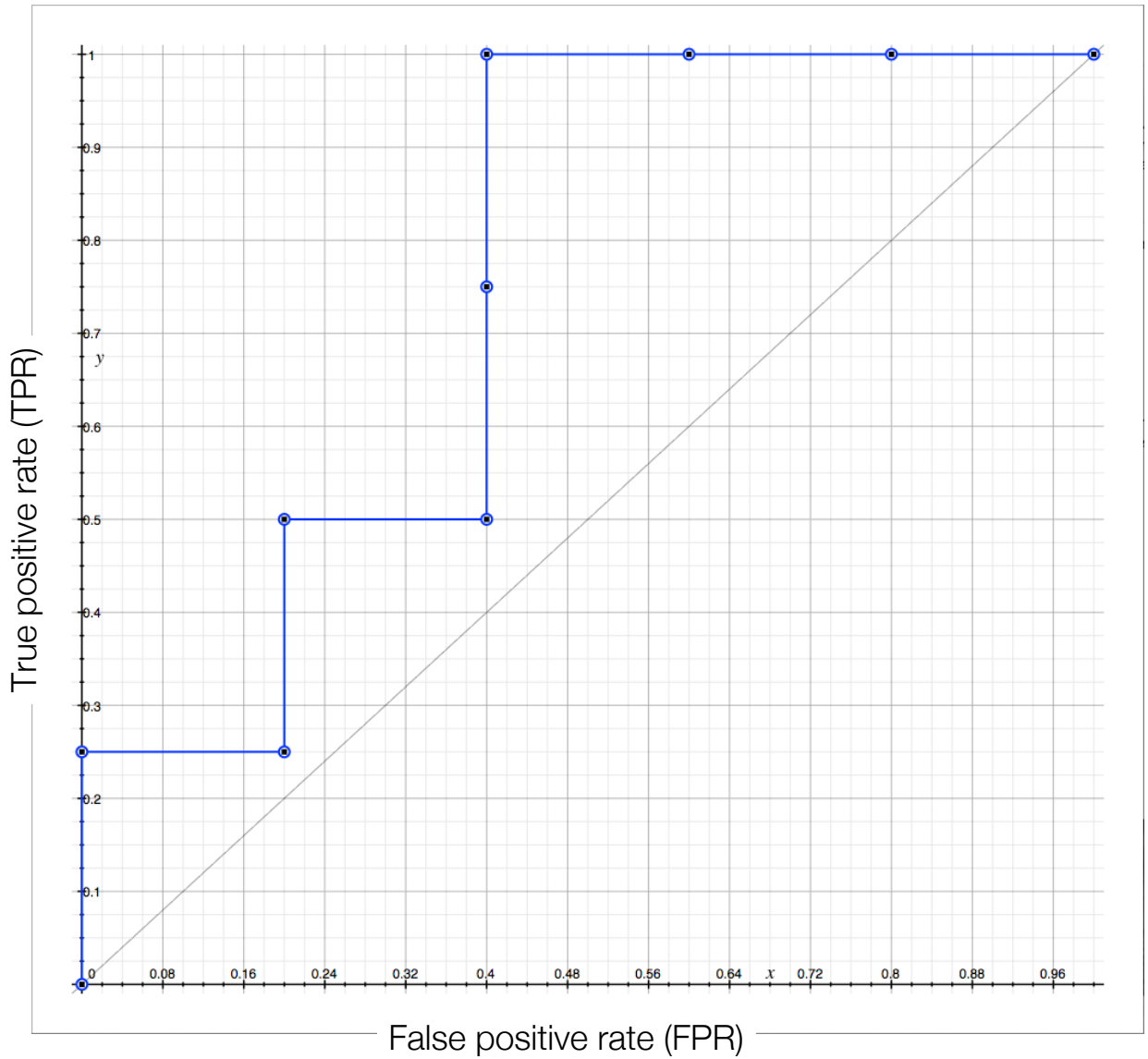
P(Y)	True class	Predict class
0.94	+	+
0.84	-	+
0.67	+	-
0.58	-	-
0.51	+	-
0.42	+	-
0.16	-	-
0.1	-	-
0.07	-	-

FPR = 1/5 TPR = 1/4

P(Y)	True class	Predict class
0.94	+	+
0.84	-	+
0.67	+	+
0.58	-	-
0.51	+	-
0.42	+	-
0.16	-	-
0.1	-	-
0.07	-	-

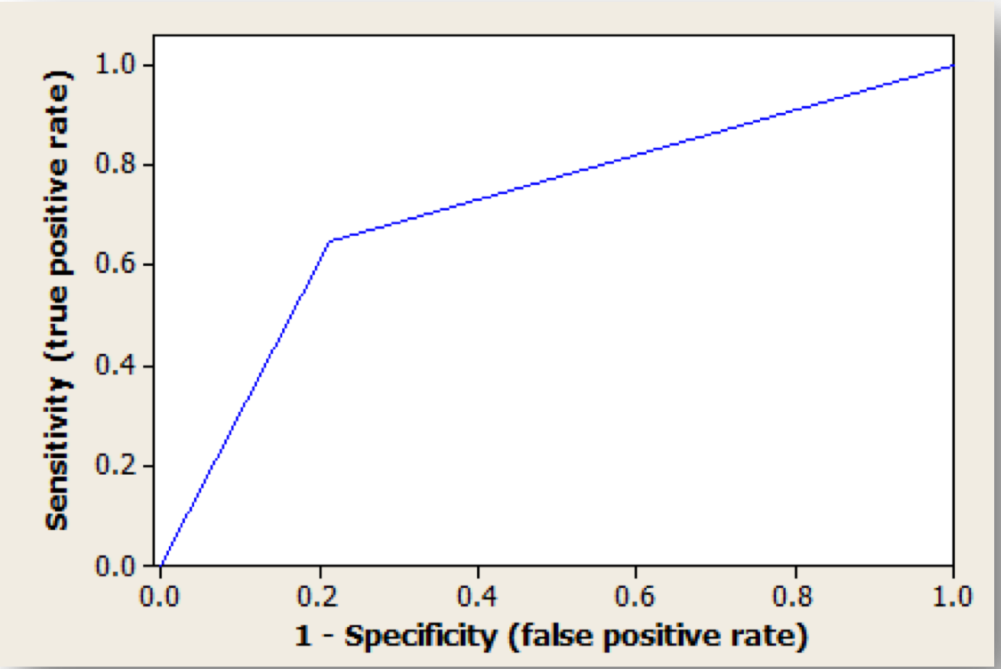
FPR = 1/5 TPR = 2/4

ROC curve


True positive rate (TPR)
False positive rate (FPR)
y
x
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0.08
0.16
0.24
0.32
0.4
0.48
0.56
0.64
0.72
0.8
0.88
0.96

ROC Curves Should **not** Look Like This

- If your ROC curve looks like this, you are likely doing something wrong
  - Either you have too few discrete features
  - Or you are predicting labels “0” and “1” but gave software library the predictions ∈ {“0”, “1”} instead of the probability ∈ [0,1] the label is “1”



Precision-Recall Curve

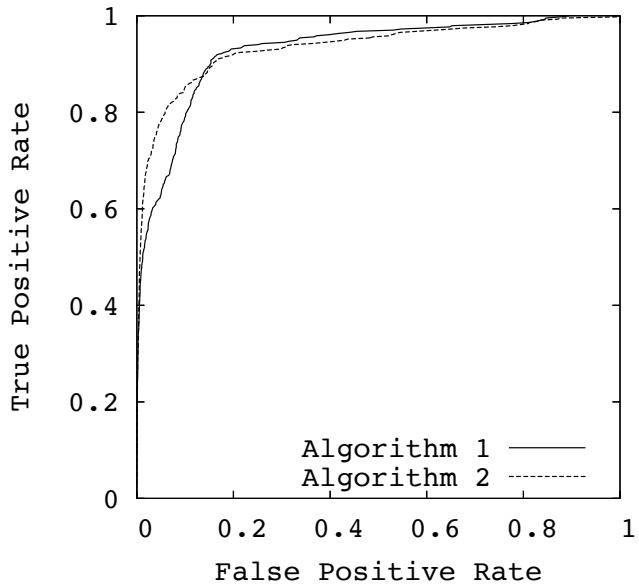
- Similarly, we can plot the Precision and Recall curve

  $$\text{Recall (Sensitivity)} = TP / (TP + FN)$$

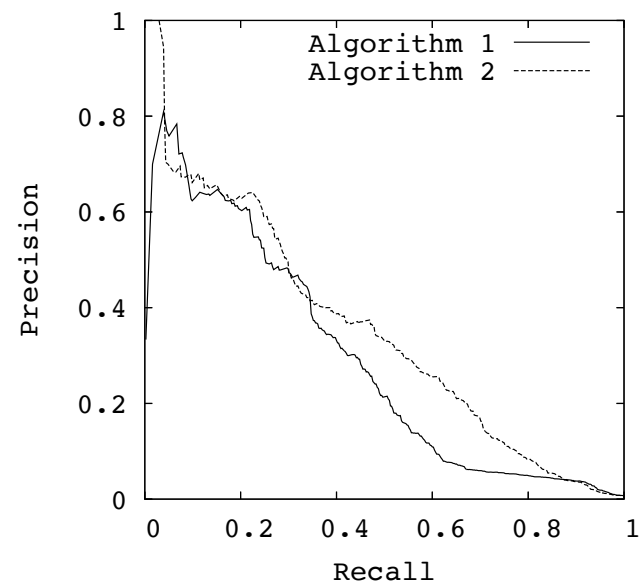
  $$\text{Precision} = TP / (TP + FP)$$

- The PR curves tend to show more fine-grained differences

- Looks only at true positives. Better than ROC if positives are rare



(a) Comparison in ROC space



(b) Comparison in PR space

Matthews correlation coefficient

- For binary classification problems, the Matthews Correlation Coefficient is another reasonable accuracy metric:

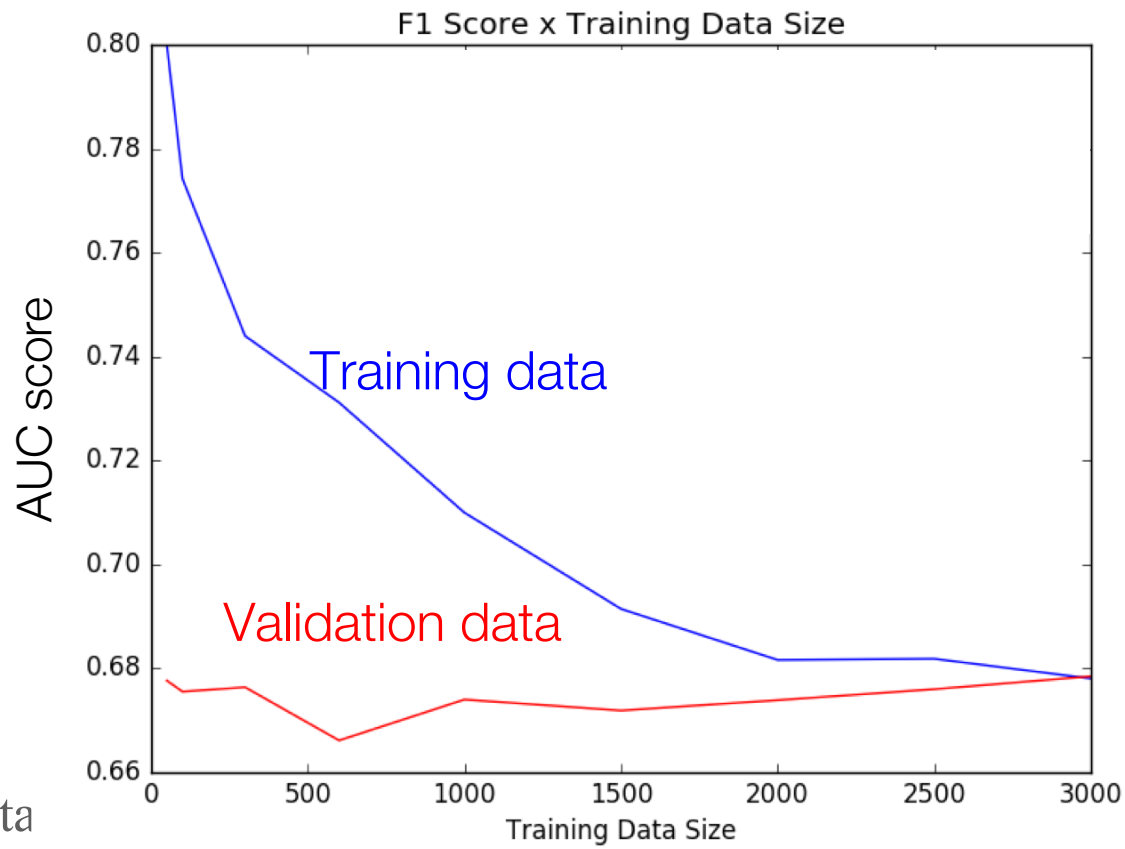
$$\mathrm{MCC} = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}}$$

- Works OK with imbalanced classes

Learning Curves (both data-rich and data-poor)

- Learning curve shows how method performs with **more training data**:
  (a) Over validation data
  (b) Over training data

- One of the most useful methods to "debug" classifiers.
  - Check for "overfitting"?
    - See gap between training and validation
  - Check for code bugs or bad local optimum?
    - Training data accuracy must decrease with more data



Issues with conventional score functions

- Assumes errors for all individuals are equally weighted
  - May want to weight recent instances more heavily
  - May want to include information about reliability of sets of measurements
- Assumes errors for all contexts are equally weighted
  - May want to weigh false positive and false negatives differently
  - May be costs associated with acting on particular classifications (e.g., marketing to individuals)

Examples

- Loan decisions
  - Cost of lending to a defaulter is much greater than the lost business of refusing loan to a non-defaulter
- Oil-slick detection
  - Cost of failing to detect an environmental disaster is far less than the cost of a false alarm
- Promotional mailing
  - Cost of sending junk mail to a household that doesn't respond is far less that the lost business of not sending it to a household that would have responded

Cost-sensitive models

- Define a score function based on a cost matrix
- If ~*y* is the predicted class and *y* is the true class, then need to define a matrix of costs *C(~y,y)*
- Reflects the severity of classifying an instance with true class *y* to class ~*y*

		Actual +	Actual −
Predicted	+	TP	FP
Predicted	−	FN	TN

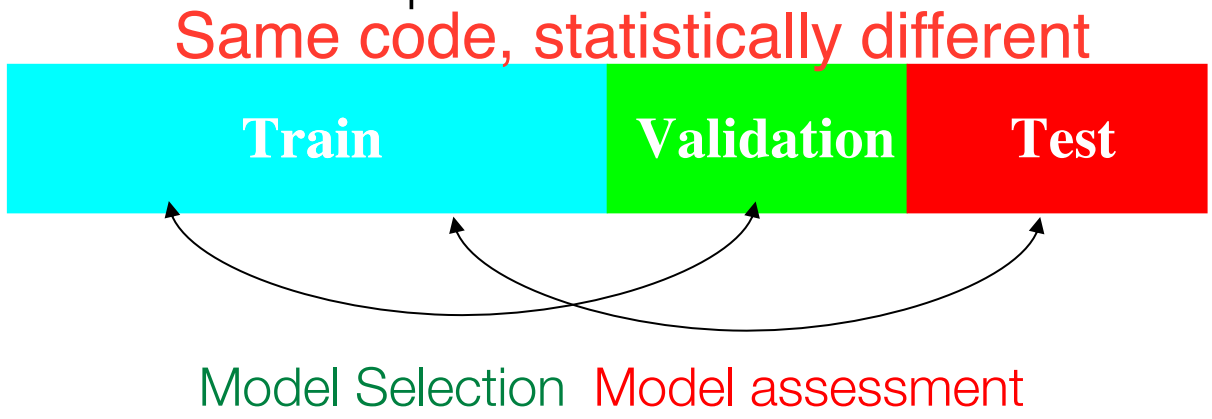
		Actual +	Actual −
Predicted	+	+$10	-$1
Predicted	−	-$5	0

Predictive modeling: summary

- Classification is an extensively studied problem
- Probably the most widely used data mining technique
  - Clearly defined task
  - Well-defined evaluation functions to guide search
  - Framework to understand model performance
- Other issues:
  - Scalability, understandability, non-standard data representations

Model Selection and Assessment

- In data-rich scenarios split the data:



  - In data-poor scenarios: *approximate validation step*
    - analytically
      - AIC, BIC, MDL
    - via sample re-use
      - cross-validation
        - Leave-one-out
        - K-fold
      - bootstrap

Data-poor Assessment of Classification Error

Model vs Hypotheses

- Simple hypothesis: single probability distributions (or functions)
  - Model with specific parameters Given weights
- Composite hypothesis: family of probability distributions (or functions)
  - A Model Looking for weights

Decomposing Test Error

True target / Predicted target

Model: $t = f(x_0; \theta^\star) + \epsilon; \quad E[\epsilon] = 0 \quad \mathrm{Var}(\epsilon) = \sigma_\epsilon^2$ (random noise)

For squared-error loss with additive noise:

Deviation of average estimate From true function's mean

$$\mathrm{Err}(x_0) = E[(t - f(x_0; \hat{\theta}))^2 | X = x_0]$$

$$= \sigma_\epsilon^2 + \left(Ef(x_0; \hat{\theta}) - f(x_0; \theta^\star)\right)^2 + E\left[f(x_0; \hat{\theta}) - Ef(x_0; \hat{\theta})\right]^2$$

$$= \sigma_\epsilon^2 + \mathrm{Bias}^2(f(x_0; \hat{\theta})) + \mathrm{Var}(f(x_0; \hat{\theta}))$$

True noise

Expected squared deviation of estimate around its own mean

Findings

- Bias
  - Often related to size of model space
  - More complex models tend to have lower bias
- Variance
  - Often related to size of dataset and quality of information
  - When data is large enough to estimate true parameters (θ^*) well we get lower variance
- With big data simple models can perform surprisingly well due to lower variance

$$
\begin{aligned}
\mathrm{Err}(x_0) &= E[(t - f(x_0;\hat{\theta}))^2 | X = x_0] \\
&= \sigma_\epsilon^2 + \left(Ef(x_0;\hat{\theta}) - f(x_0;\theta^\star)\right)^2 + E\left[f(x_0;\hat{\theta}) - Ef(x_0;\hat{\theta})\right]^2 \\
&= \sigma_\epsilon^2 + \mathrm{Bias}^2(f(x_0;\hat{\theta})) + \mathrm{Var}(f(x_0;\hat{\theta}))
\end{aligned}
$$

Optimism of the Training Error Rate

- Typically: training error rate < true error

(same data is being used to fit the method and assess its error)

$$Err_{\text{train}} = \underbrace{\frac{1}{N}\sum_{i=1}^{N} L(t(i), f(x_0(i); \hat{\theta}))}_{\text{overly optimistic}} < Err(X) = E[L(t, f(X; \hat{\theta}))]$$

Estimating Test Error

- Can we estimate the discrepancy between Err(x_0) and Err(X)?
- D – old observations; D^{new} – new observations
- Suppose we observed new values of label $t(i)$ for same input $x(i)$

Err_{in} --- In-sample error:

Expectation over N new responses at each x_i

$$E_D[Err_{\text{in}}] = \frac{1}{N}\sum_{i=1}^{N} E_D E_{D^{\text{new}}} L(t^{\text{new}}(i), f(x_0(i); \hat{\theta}))$$

Trained with old observation D

$$= E_D[Err_{\text{train}}] + \frac{2}{N}\sum_{i=1}^{N} \text{Cov}(\hat{t}_i, t_i)$$

Adjustment for optimism of training error

Measuring Optimism

If model: $t = f(X;\theta) + \epsilon; \quad E[\epsilon] = 0 \quad \mathrm{Var}(\epsilon) = \sigma_\epsilon^2$
with $|\theta| = d$

Then, if f is linear, $f(x;\theta) = x^T\theta$ then

$$E_D[Err_{\text{in}}] = E_D[Err_{\text{train}}] + \frac{2}{N} d\, \sigma_\epsilon^2$$

- Optimism grows linearly with model dimension
- Optimism decreases as training sample size increases

Ways to Estimate Prediction Error

- In-sample error estimates:
  - AIC
  - BIC
  - MDL
- Extra-sample error estimates:
  - Cross-Validation
    - Leave-one-out
    - K-fold
  - Bootstrap

Estimates of In-Sample Prediction Error

- General form of the in-sample estimate:

$$Err_{\text{in}} = Err_{\text{train}} + \text{optimism error}$$

- For additive error:

$$C_p = \text{Err}_{\text{train}} + \frac{2d}{N}\hat{\sigma}_\epsilon^2$$

Known as the Marllow's C_p statistic

AIC & BIC

Akaike Information Criterion (AIC)

Linear models
log-likelihood

$$\mathrm{AIC} = -\frac{2}{N} \ln P[t = f(x_0; \hat{\theta})] + 2\frac{d}{N}$$

AIC does not assume model is correct
Better for small # samples
May not converge to "correct model"
(as $N \to \infty$, model bias may be $\neq 0$)

Linear models

Bayesian Information Criterion (BIC)

$$\mathrm{BIC} = -\frac{2}{N} \ln P[t = f(x_0; \hat{\theta})] + \frac{d}{N} \log N$$

converge to correct model

BIC assumes model is correct
Converges to correct parameterization (θ^*) of model
(as $N \to \infty$ model bias = 0)

MDL (Minimum Description Length)

- Find hypothesis (model) that minimizes
  - H(M) + H(D|M), where
    - H(M) – length in bits of description of model M
    - H(D|M) – length in bits of description of data encoded by model M

$$\text{length} = -\ln P[t = f(x_0; \hat{\theta})|M] - \ln P[\hat{\theta}|M]$$

Length of transmitting the discrepancy given the model + optimal coding under the given model

Description of the model under optimal coding

MDL principle: choose the model with the minimum description length

- If model probabilistic:

  Equivalent to maximizing the posterior: $P[t = f(x_0; \hat{\theta})|M]P[\hat{\theta}|M]$

BIC is one example of Bayesian Approach

- While we will not cover Bayesian methods in this course, they tend to ameliorate some of these point estimate issues
- Harder to compute

Bayesian Approach

- Not interested in a single ”Best Hypothesis”
  - Not using a point estimate (single parameter $\boldsymbol{\theta}$) of model parameters

- Seeks to take variance into account into model evaluation
- For instance, consider mean square error of data (x_0, t), where x_0 are the features and t is the class

$$\mathrm{Err}(x_0) = E[(t - f(x_0; \tilde{\theta}))^2 | X = x_0, \tilde{\theta}] P[\tilde{\theta} | Data]$$

- If testing too many hypotheses and not enough data, $P[\hat{\theta} | X = x_0]$ is less concentrated (higher entropy), thus leading to larger errors

Data-rich Assessment of Classification Error

Estimation of Extra-Sample Error

- Cross Validation

- Bootstrap

Estimation of Extra-Sample Error via Cross-Validation

- *Goal:* Estimate the test error for a supervised learning method

- *Algorithm:*

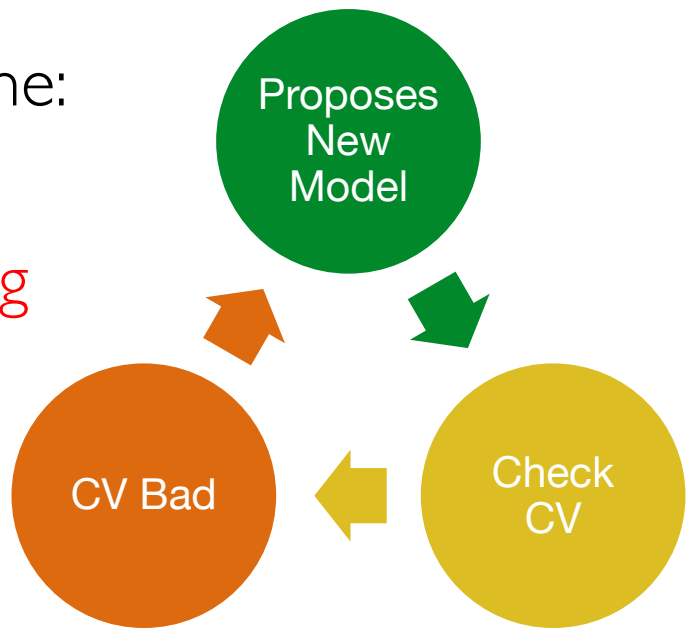
  1. Split the data in two parts (training and validation)
  2. Train the method in the *training* part
  3. Compute the error on the *validation* part

Wrong Way to Perform Cross-validation

- Use cross-validation to estimate too many model decisions or for selection of too many features

- Use cross-validation to hunt for too many new models
  - Same problem we have when hunting for p-values (too many hypotheses)

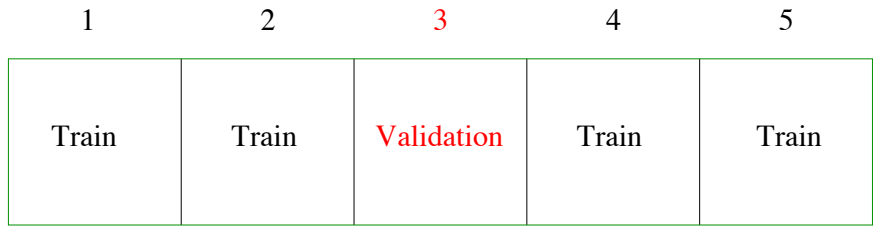
    The human regression machine:

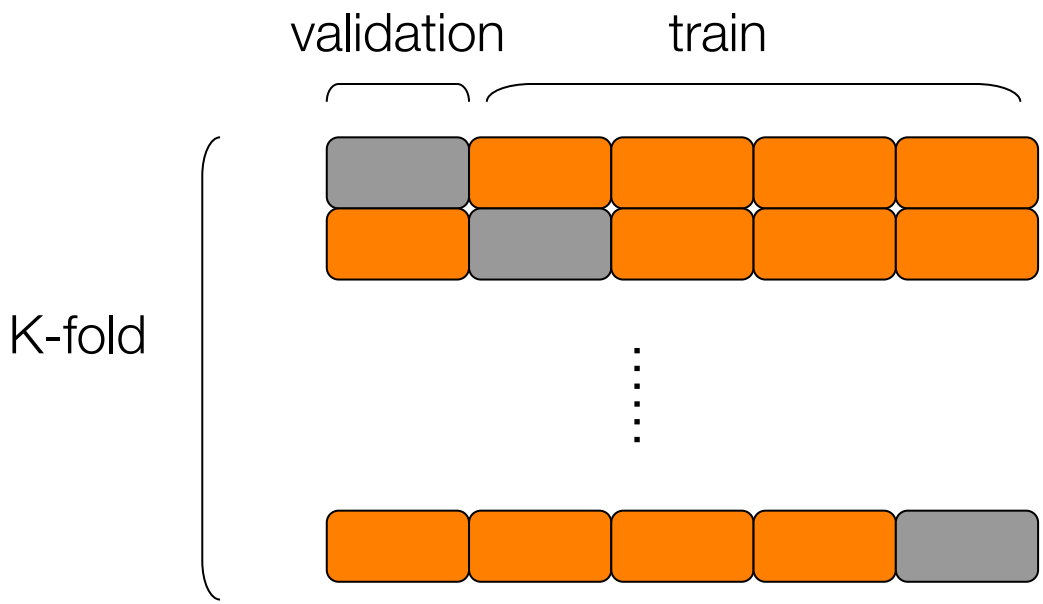


Same as multiple hypothesis testing

In Data-poor scenarios we should use
K-fold Cross-validation

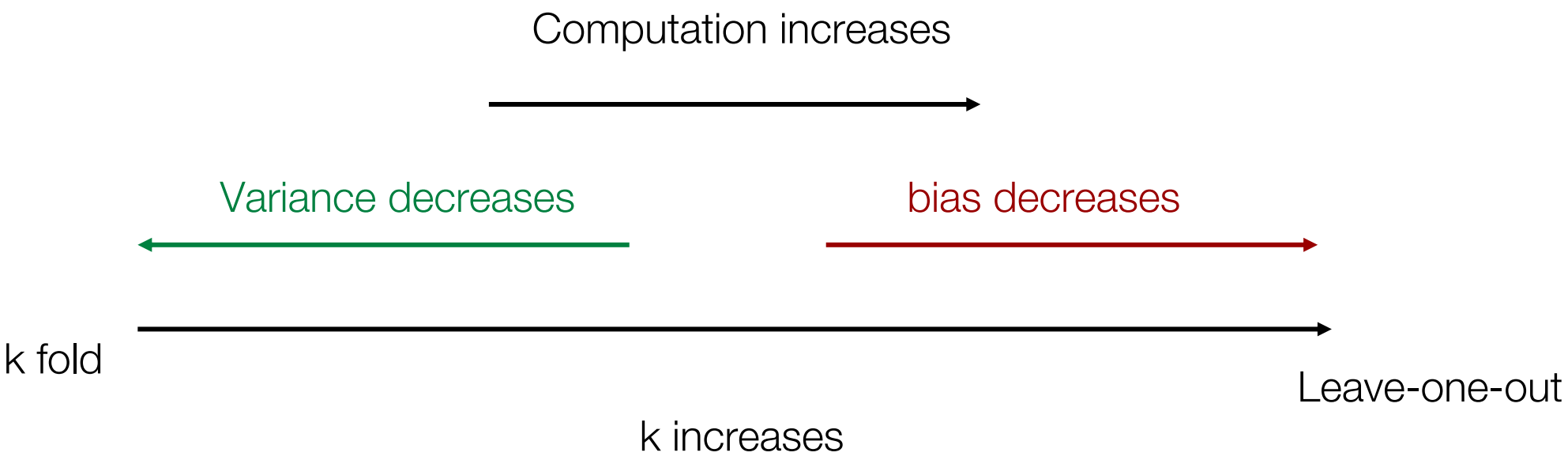
Cross-Validation



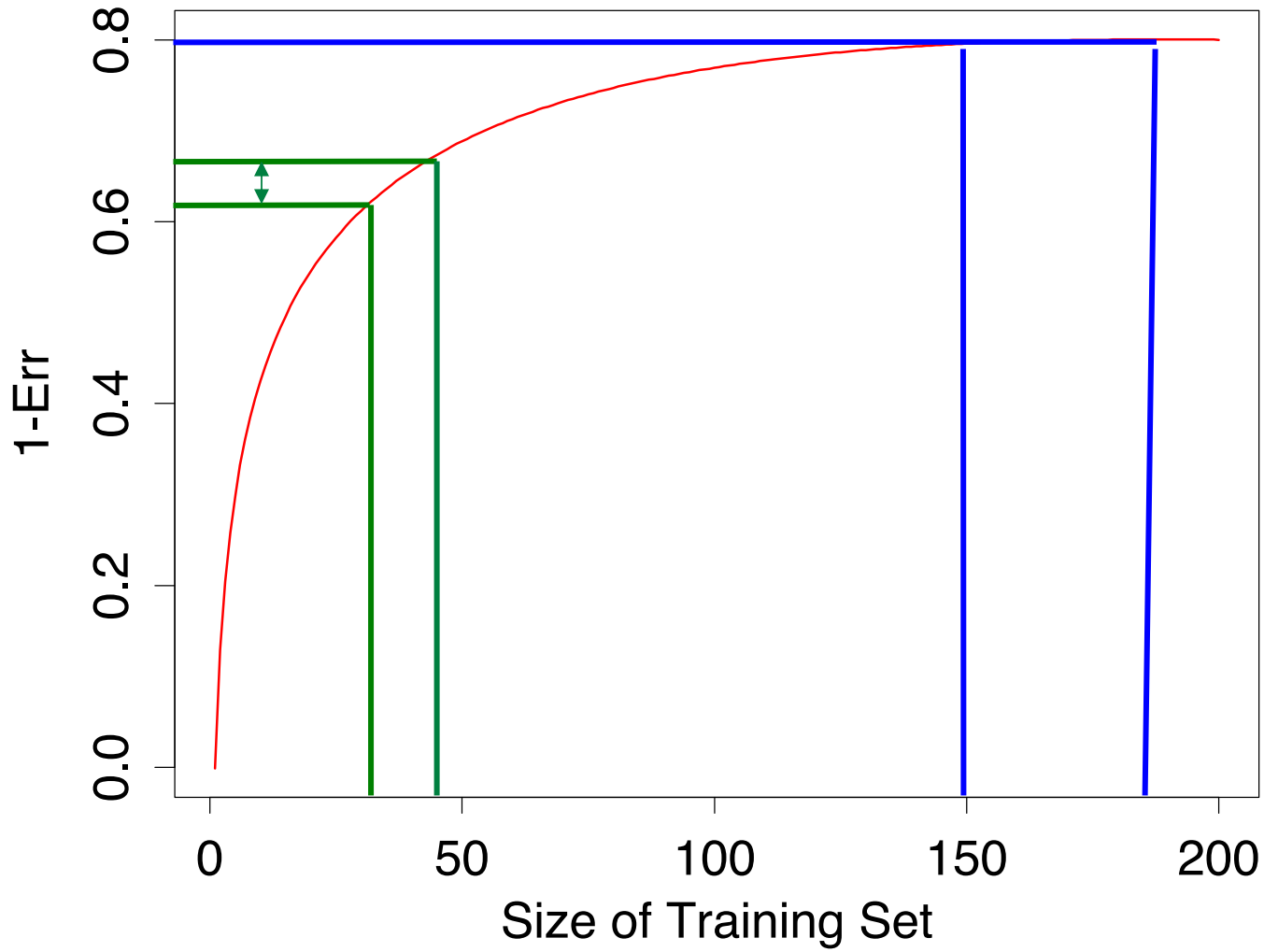


$$\mathrm{CV}_{\mathrm{err}}(\hat{\theta}) = \frac{1}{N}\sum_{i=1}^{N} L(y(i), f(x_0^{(-K)}(i); \hat{\theta}))$$

How many folds?



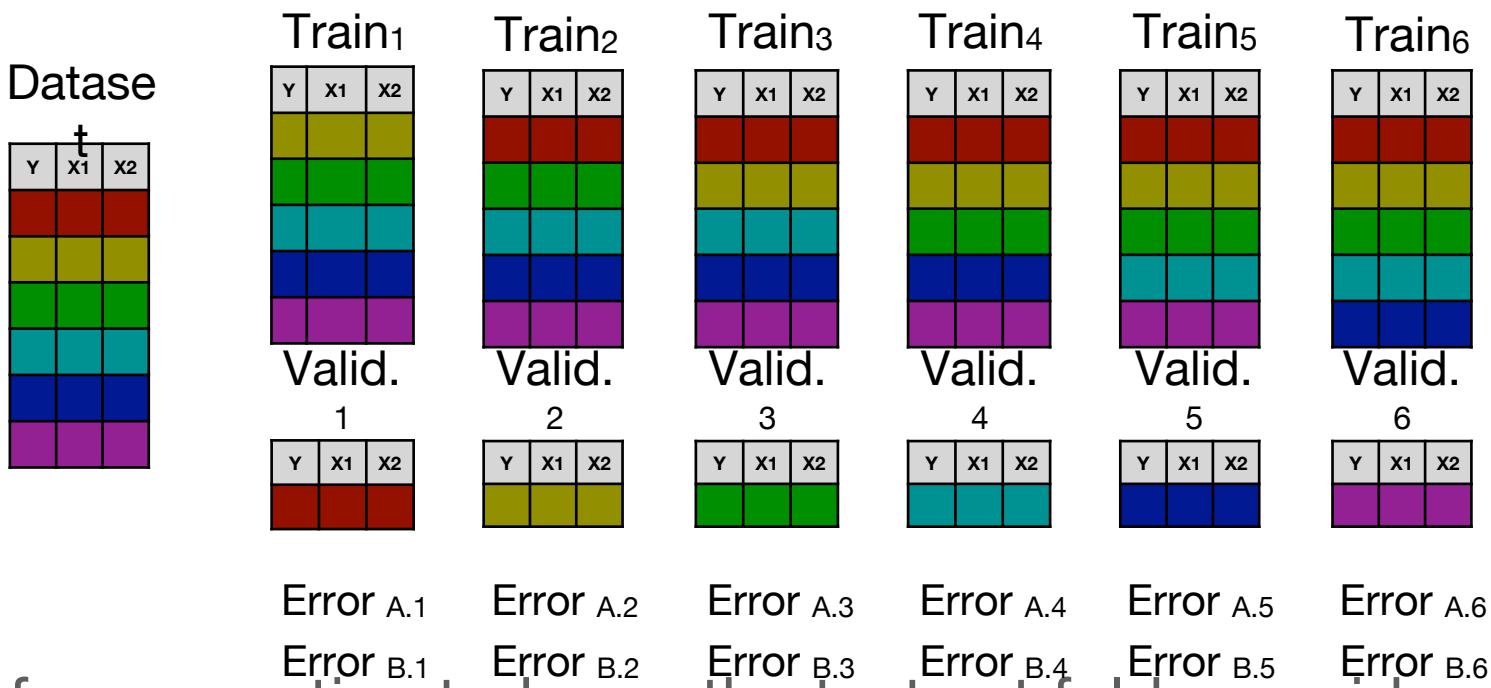
Cross-Validation: Choosing K



Popular choices for K: 5, 10, N=leave one out

Evaluating classification algorithms A and B

- Use k-fold cross-validation to get k estimates of error for algorithms A and B



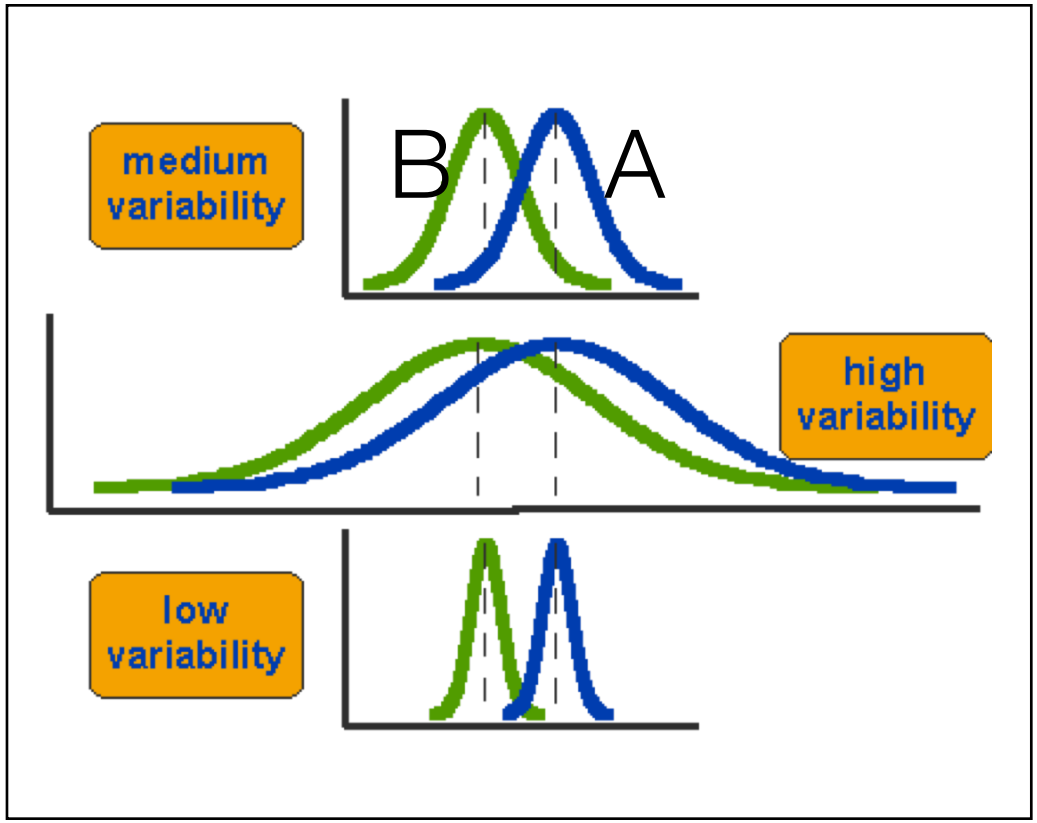
- Set of errors estimated over the test set folds provides empirical estimate of sampling distribution
- Mean is estimate of expected error
- Is the mean error significant between A and B?

Assessing significance

- Use **paired t-test** to assess whether the two distributions of errors are statistically different from each other

Error $_{A.1}$	Error $_{B.1}$
Error $_{A.2}$	Error $_{B.2}$
Error $_{A.3}$	Error $_{B.3}$
Error $_{A.4}$	Error $_{B.4}$
Error $_{A.5}$	Error $_{B.5}$

- Takes into account both the difference in means and the variability of the scores



Paired t-test

- Instead of comparing the difference in means between populations 1 and 2

$$\frac{1}{N}\sum_{i=1}^{N} X_i^{(1)} - \frac{1}{N}\sum_{i=1}^{N} X_i^{(2)}$$

- A paired t-test will compare the mean difference

$$\bar{d} = \frac{1}{N}\sum_{i=1}^{N}(X_i^{(1)} - X_i^{(2)})$$

- The real effect is when estimating the standard error of the mean difference

Var of d

$$\mathrm{SE}(\bar{d}) = \frac{s_d}{\sqrt{n}}, \qquad \text{where } s_d = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(X_i^{(1)} - X_i^{(2)} - \bar{d})^2}$$

- The *t* statistic has *N*-1 degrees of freedom and is given by

$$t = \frac{\bar{d}}{\mathrm{SE}(\bar{d})}$$

Bootstrapping

Bootstrapping

- Bootstrap is a powerful statistical tool to quantify the uncertainty associated with a given estimator or statistical learning method

- E.g.: It can provide an estimate of the standard error of a coefficient, or a confidence interval for that coefficient

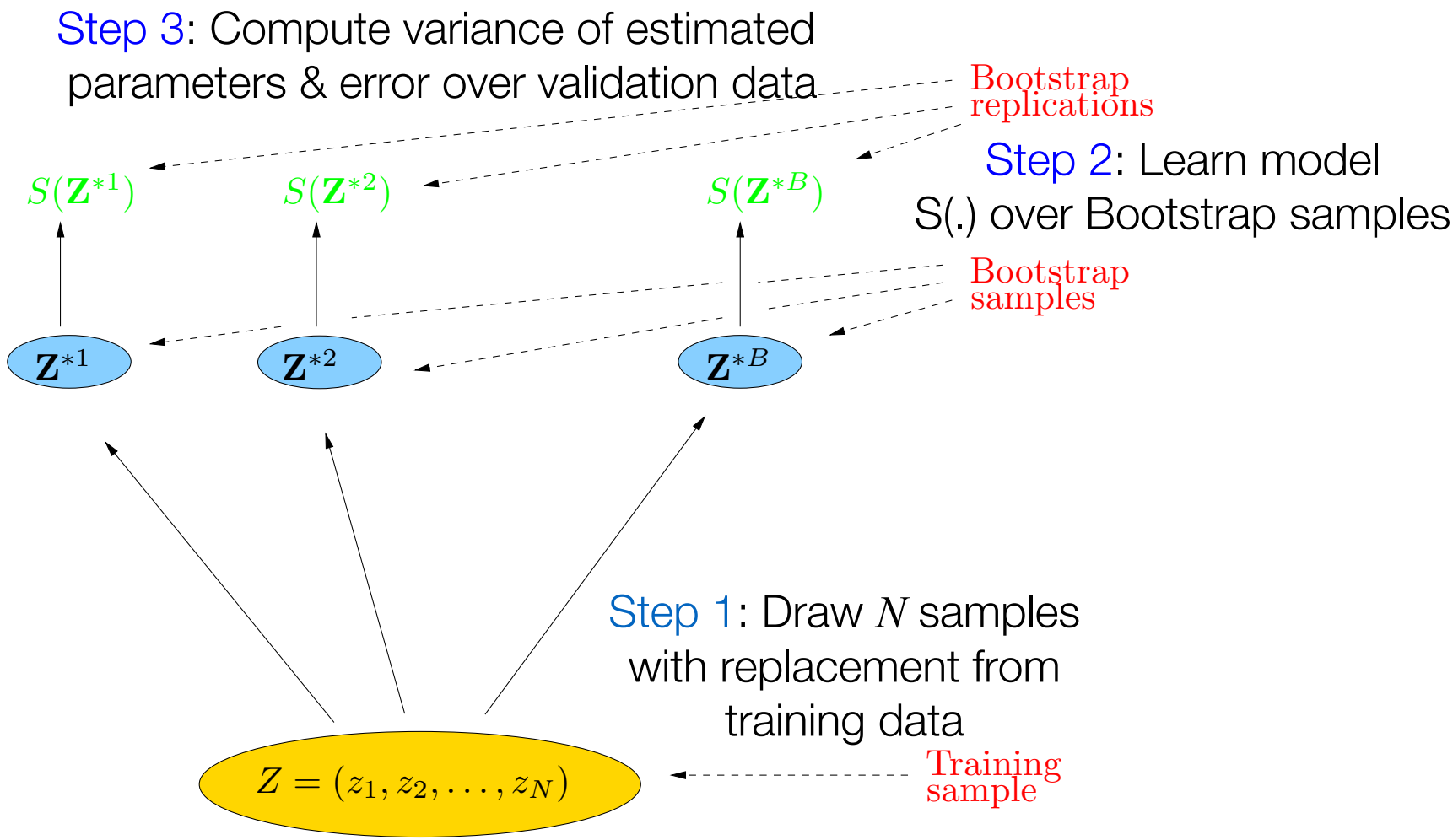
- Instead of sampling new datasets from the unknown population distribution, resample from the training data (empirical population distribution)

Why *Bootstrapping?*

- The term *bootstrap* derives from "*to pull oneself up by one's bootstraps*", likely based on one of the eighteenth century novel "The Surprising Adventures of Baron Munchausen" by Rudolph Erich Raspe:
    - The Baron had fallen to the bottom of a deep lake. Just when it looked like all was lost, he thought to pick himself up by his own bootstraps.

- It is not the same as the term "bootstrap" used in computer science meaning to "boot" a computer from a set of core instructions, though the meaning is similar.

Bootstrap: Main Concept



Bootstrap Validation Data

- Two ways to use validation data
  - Validation comes from whatever data points are not used in training
    - Issue:
      Validation data size not consistent across bootstrap replications →
      Validation error uncertainty not consistent
    - Advantage:
      Keeps all training data for potentially training model
  - Validation split from training before bootstrapping
    - Issue:
      Must split validation from training, reducing training data
    - Advantage:
      Validation data size kept consistent between bootstrap replications