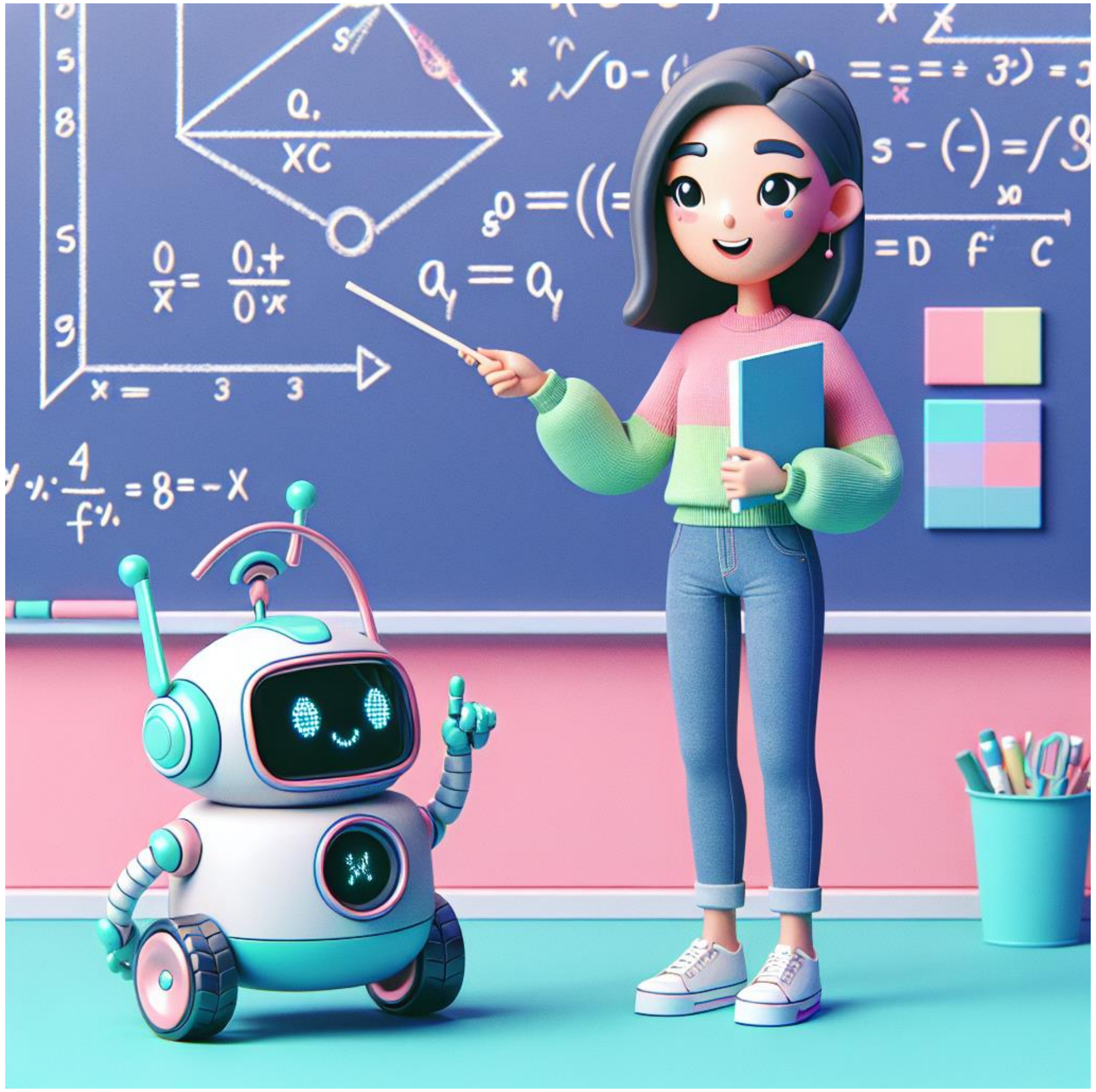
Supervised Learning



Supervised Learning

- Main tasks of supervised learning:
  - **Classification**: The expected output is one of several categories. Example: Apple variety classification, gender classification.
  - **Regression**: The expected output is a numerical value. Example: temperature in weather forecast, age estimation.
  - Some applications fall between these two. Examples: Age group classification, disease stage classification.
- Each involves the learning of some input/output mapping: $f: \mathbf{x} \rightarrow \mathbf{y}$. Note: Both $\mathbf{x}$ and $\mathbf{y}$ are vectors in general.

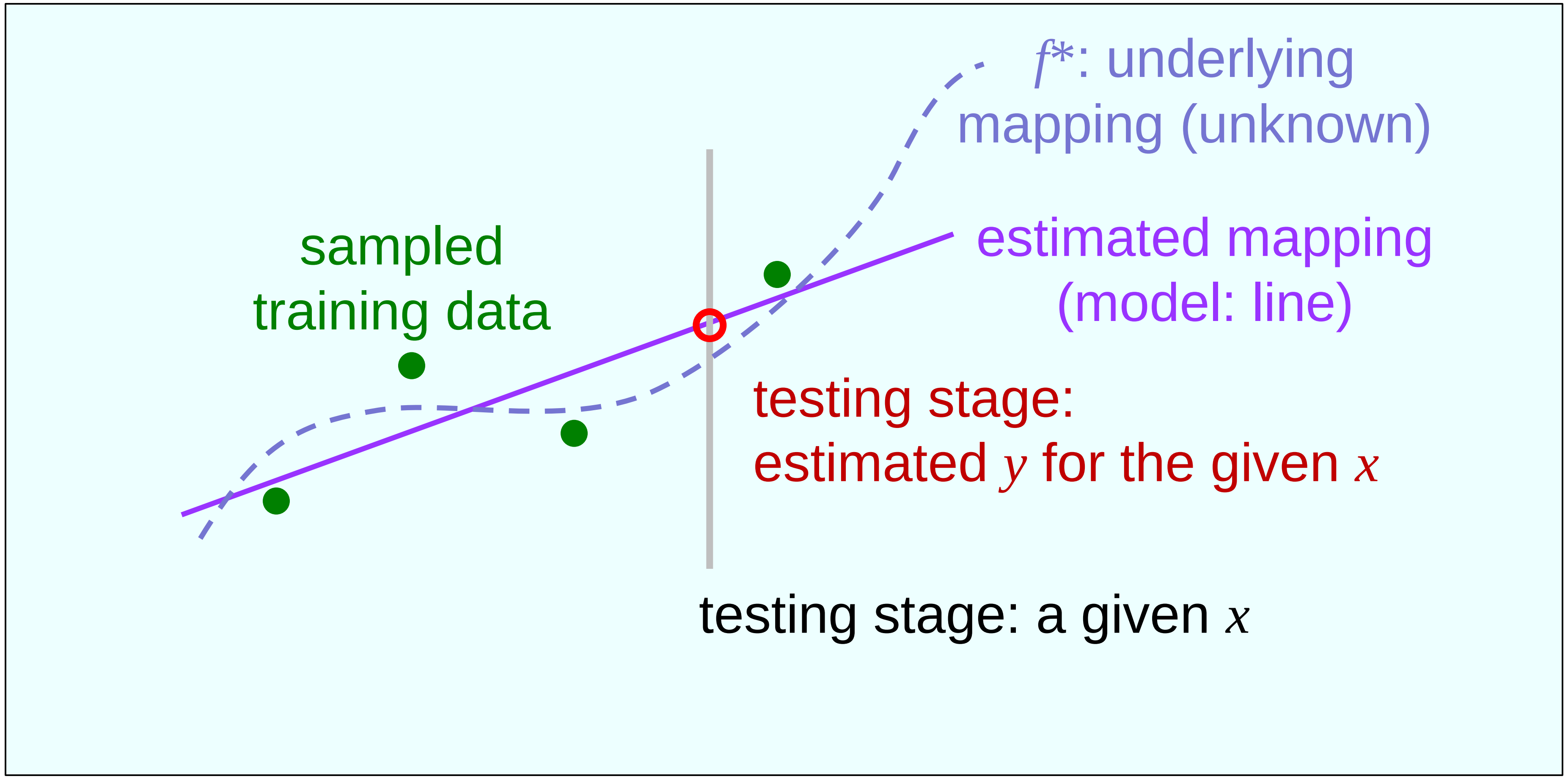
Supervised Learning Overview

Training and testing:

- Training data: Collect samples in the form of ($\boldsymbol{x} \rightarrow \boldsymbol{y}$).
  - For such a pair of ($\boldsymbol{x} \rightarrow \boldsymbol{y}$), $\boldsymbol{y}$ is often called the ground-truth or target output of $\boldsymbol{x}$.
  - If $\boldsymbol{y}$ represents a category (class), then it is also called the class label (or simply the label) of $\boldsymbol{x}$.
- Training: Derive an estimated model of the mapping f using the training data.
- Testing: Using the derived model, determine $\boldsymbol{y}$ for some given $\boldsymbol{x}$ (normally not in the training data).

Example: Regression

Here we will use regression as an example.


f*: underlying mapping (unknown)
estimated mapping (model: line)
sampled training data
testing stage: estimated y for the given x
testing stage: a given x

Supervised Learning Overview

- Stages of training a model for $f: x \rightarrow y$:

Model Type	Polynomial? Decision tree? Neural network? ...
Model Design Hyper-parameters	Polynomial: degree? Decision tree: maximum depth? Neural network: #neurons? ...
Parameter Estimation	Polynomial: coefficients Decision tree: threshold at each node Neural network: network weights ...

Classification Models

Two approaches to build classification models:

- **Generative models**: Build models of the individual classes, and classify a new sample by comparing its “similarity” or “compatibility” to the individual models.
  - Examples: Bayesian classifiers and their derivatives like naïve Bayes, k-nearest-neighbor, etc..
- **Discriminative models**: Models that do classification by identifying the differences or boundaries between classes.
  - Examples: Logistic regression, SVM, decision trees, etc..

Bias-Variance Dilemma

Desired properties of the model:

- Low **Bias** (goodness of fitting):
  - The estimated mapping should fit to the training data as well as possible.
  - Goal: To capture the behaviors of the underlying mapping as well as possible.
- Low **Variance** (high confidence of prediction):
  - Assume that we have several estimated mappings, each derived from a different set of training samples.
  - For a given x, each mapping gives an estimation for y.
  - The estimations for y should by as similar to each other as possible.
  - Goal: To be confident of the estimation of y.

Bias-Variance Dilemma

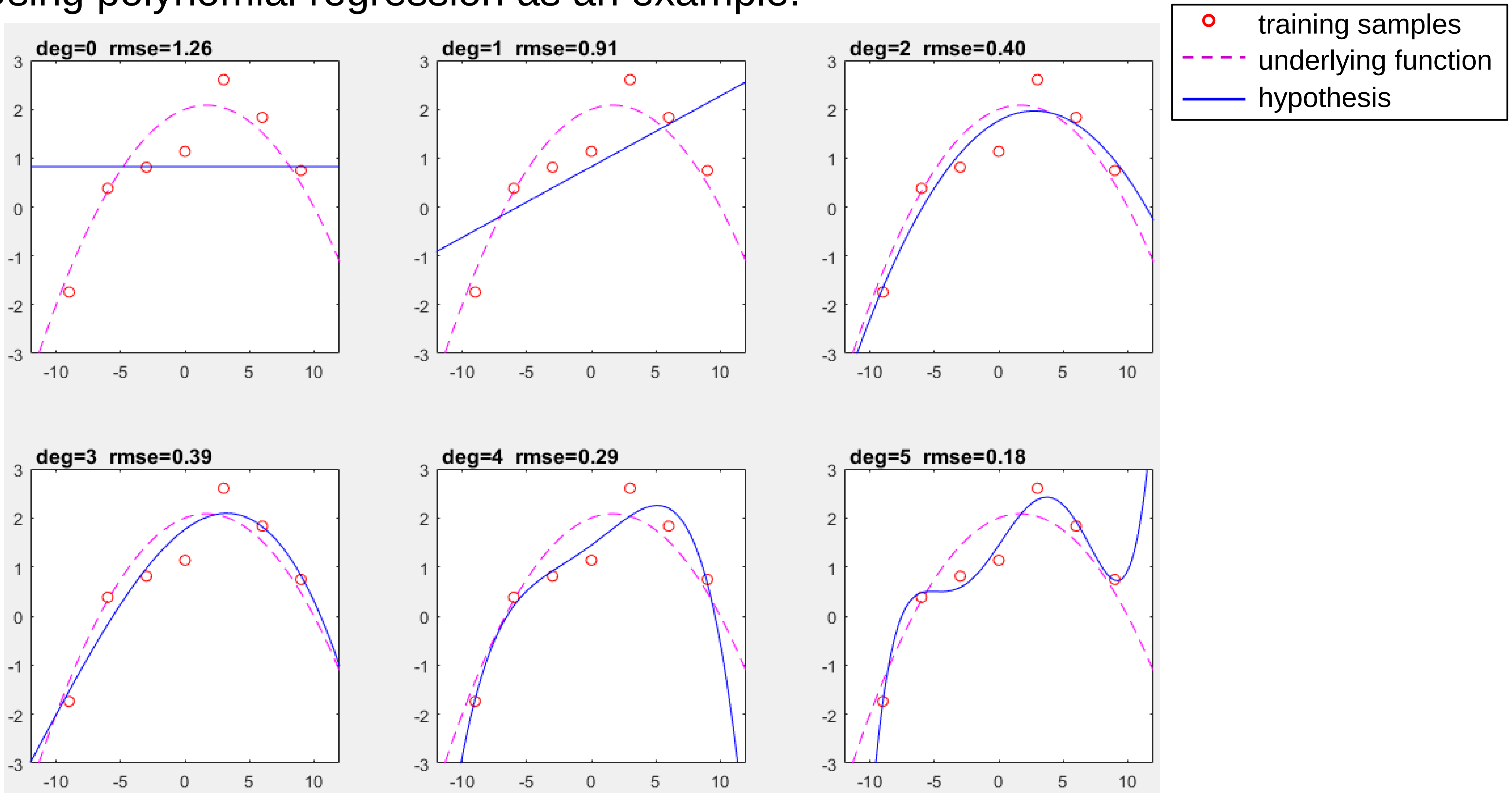
Now, difficulties for supervised learning:

- To reduce **Bias**
  - ➔ Use more complex models
  - ➔ BOTH signal and noise are learned
  - ➔ Overfitting
- To reduce **Variance**
  - ➔ Try to avoid learning noise (variance is caused by noise)
  - ➔ Use less complex models
  - ➔ Signal is not sufficiently learned
  - ➔ Underfitting

Here we use the term "noise" to represent variations in data that are not relevant to the prediction.

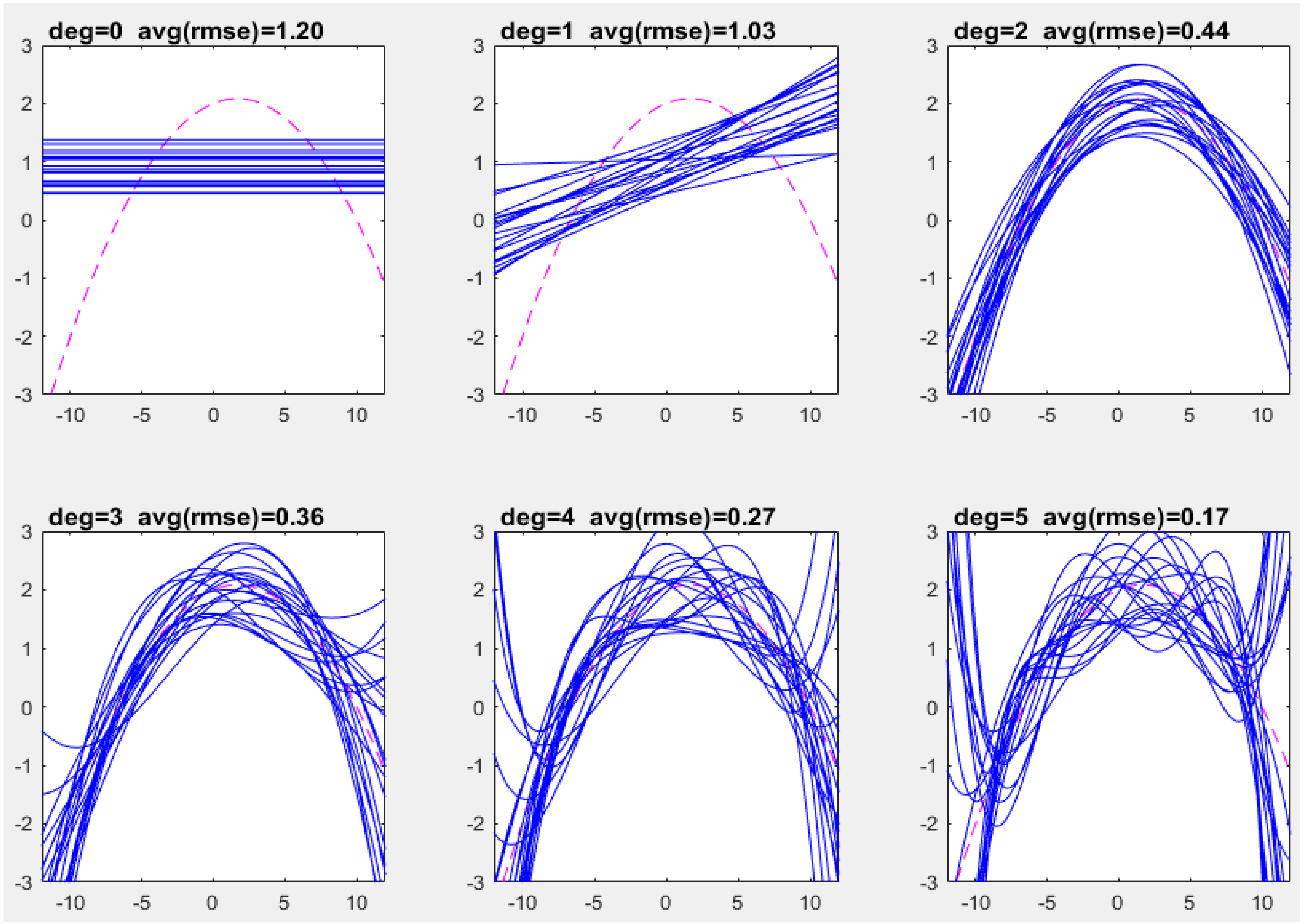
Fitting and Model Complexity

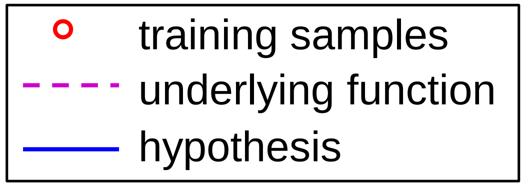
Using polynomial regression as an example:



Fitting and Model Complexity

Illustration of corresponding variances:





Each degree is sampled 20 times.

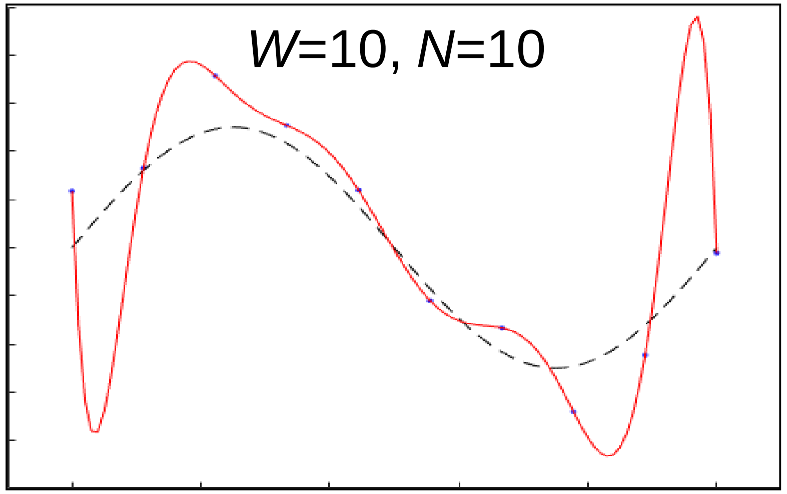
What to Do Now?

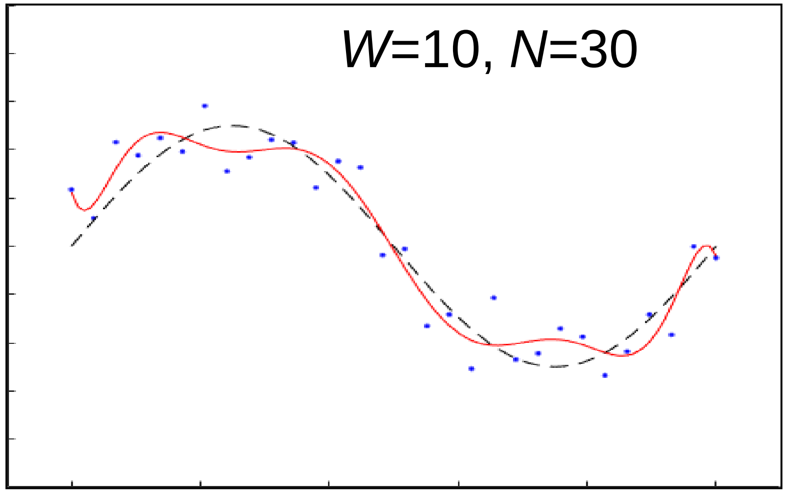
Several representative approaches:

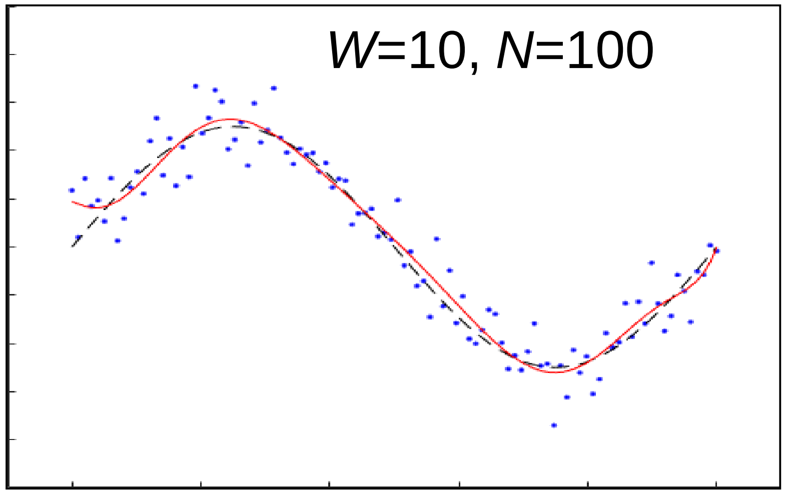
- Increasing the number of training samples.
- Trying to determine the suitable level of model complexity.
- Regularization (the "suppression" of model complexity during training)
- Consensus-based methods (ensembles)

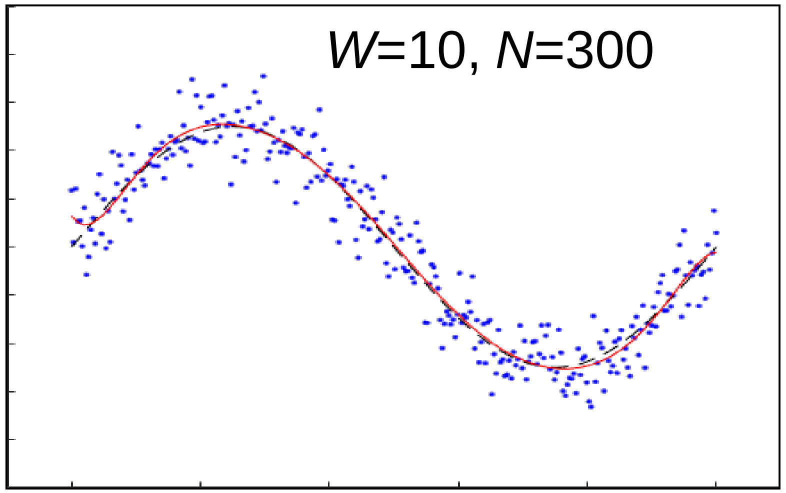
Amount of Training Data

- The usual case: The more training data, the better.
- Example (polynomial regression):









N: # samples
W: # parameters

Amount of Training Data

- The usual case: The more training data, the better.
- Possible difficulties:
  - Data availability
  - Increased computational cost
  - Data imbalance
- When there are few training samples, some classifier types (e.g., support vector machines) generally work better than others.
- **Data augmentation** and **resampling**: Produce new training data through the perturbation or combination of one or multiple samples of the same class.
- **Transfer learning**: Train the model first on a similar and larger dataset, then just adjust its parameters on the dataset being analyzed.

Regularization

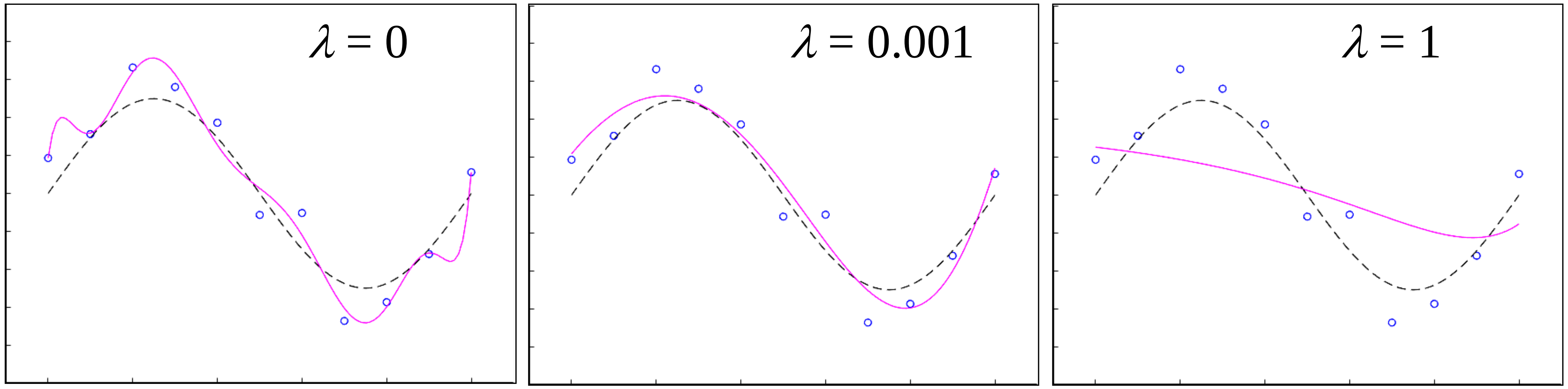
- Include the **bias-variance** trade-off during the learning process:
  - **Regularization**: Modify the learning objective to minimize BOTH the bias and the model complexity. (Normally, the learning process only attempts to minimize bias.)
  - Example **cost function** (to be minimized) for polynomial regression:

To minimize:

$$E = \underbrace{\sum_{i} \left| y_i - \sum_{k=0}^{d} a_k x_i^k \right|^2}_{\substack{\text{Fitting Error} \\ \text{(reducing bias)}}} + \lambda \underbrace{\sum_{k=0}^{d} a_k^2}_{\substack{\text{Regularization} \\ \text{(reducing complexity)}}}$$

Regularization

- Example (polynomial regression):



N=11, degree=9

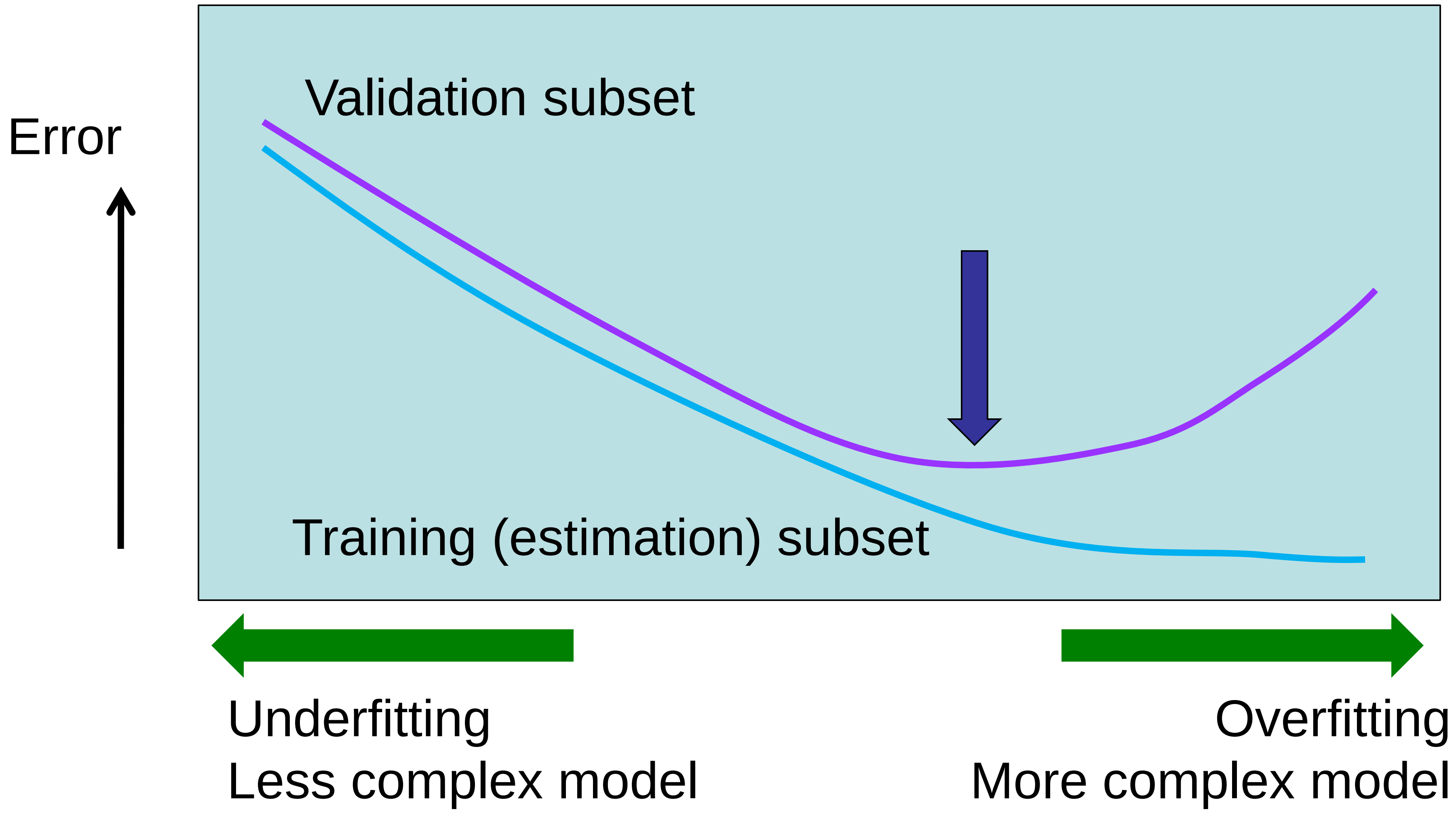
Consensus Based Methods

- Build many somewhat different models (an **ensemble**) for the same underlying mapping.
- The consensus should better represent the "signal" part, and the "noise" part is more likely to be averaged out. This leads to the reduction of variance with minimal effect on bias.
- The models in an ensemble need to be diverse:
  - Random subset of attributes/features
  - Resampling (bootstrapping) of training samples

How Much Model Complexity?

- **Ockham's Razor**: Among different ways to explain the data, choose the simplest one.
- Model complexity is usually controlled / adjusted using some **hyperparameters**.
- Example methods: **cross-validation**, post-processing model pruning.
- Many of these methods utilize **validation data** (training samples not used for model parameter estimation) to check the **generalization** ability of the built model.

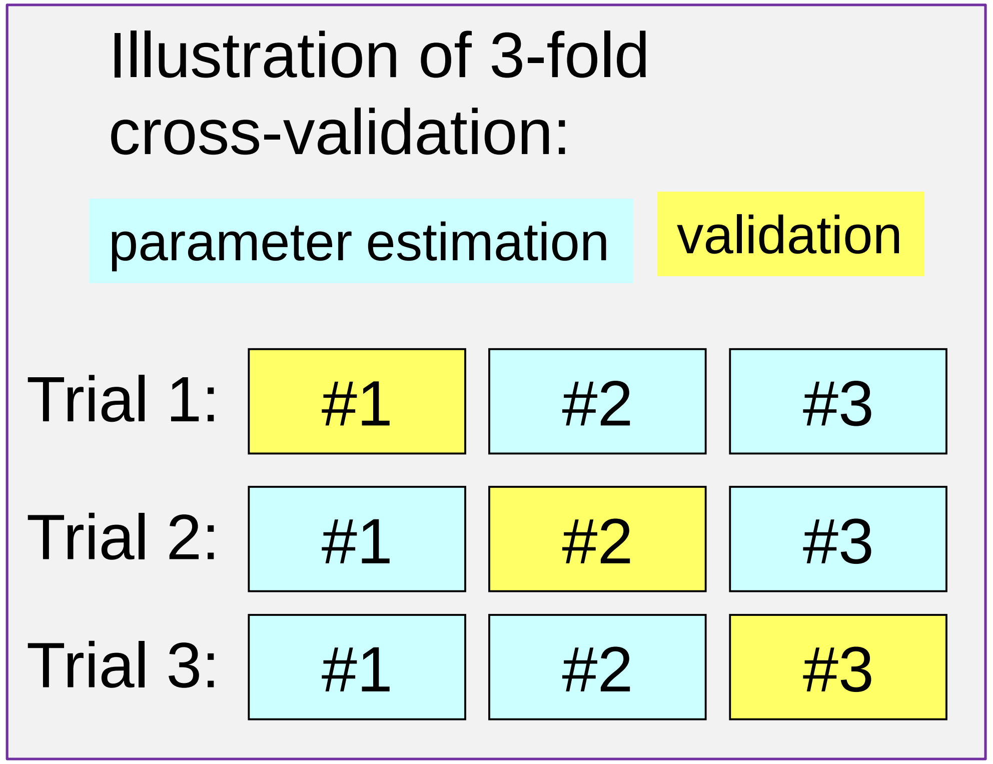
Validation



Error
Validation subset
Training (estimation) subset
Underfitting
Less complex model
Overfitting
More complex model

Cross-Validation

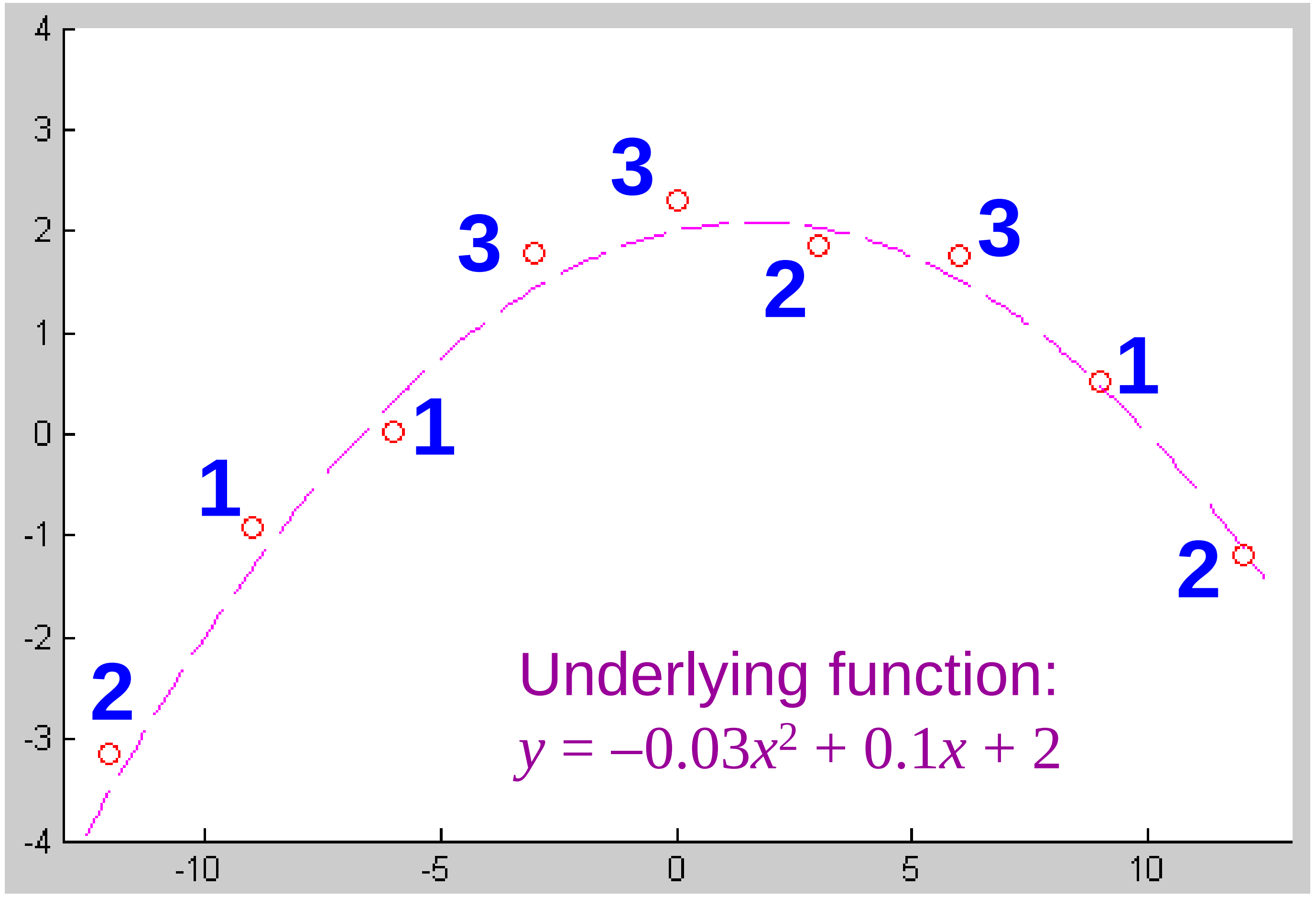
- The N labeled samples are divided into K subsets of approximately equal sizes ($K>1$). They should have similar distributions.
- The training process (model parameter estimation) is run K times (K trials).
- In the K^{th} trial, the K^{th} subset is used for validation, and the other subsets are used for training.
- The overall performance is the combination of the performance on all the validation subsets.
- Extreme case: The leave-one-out method ($K=N$).



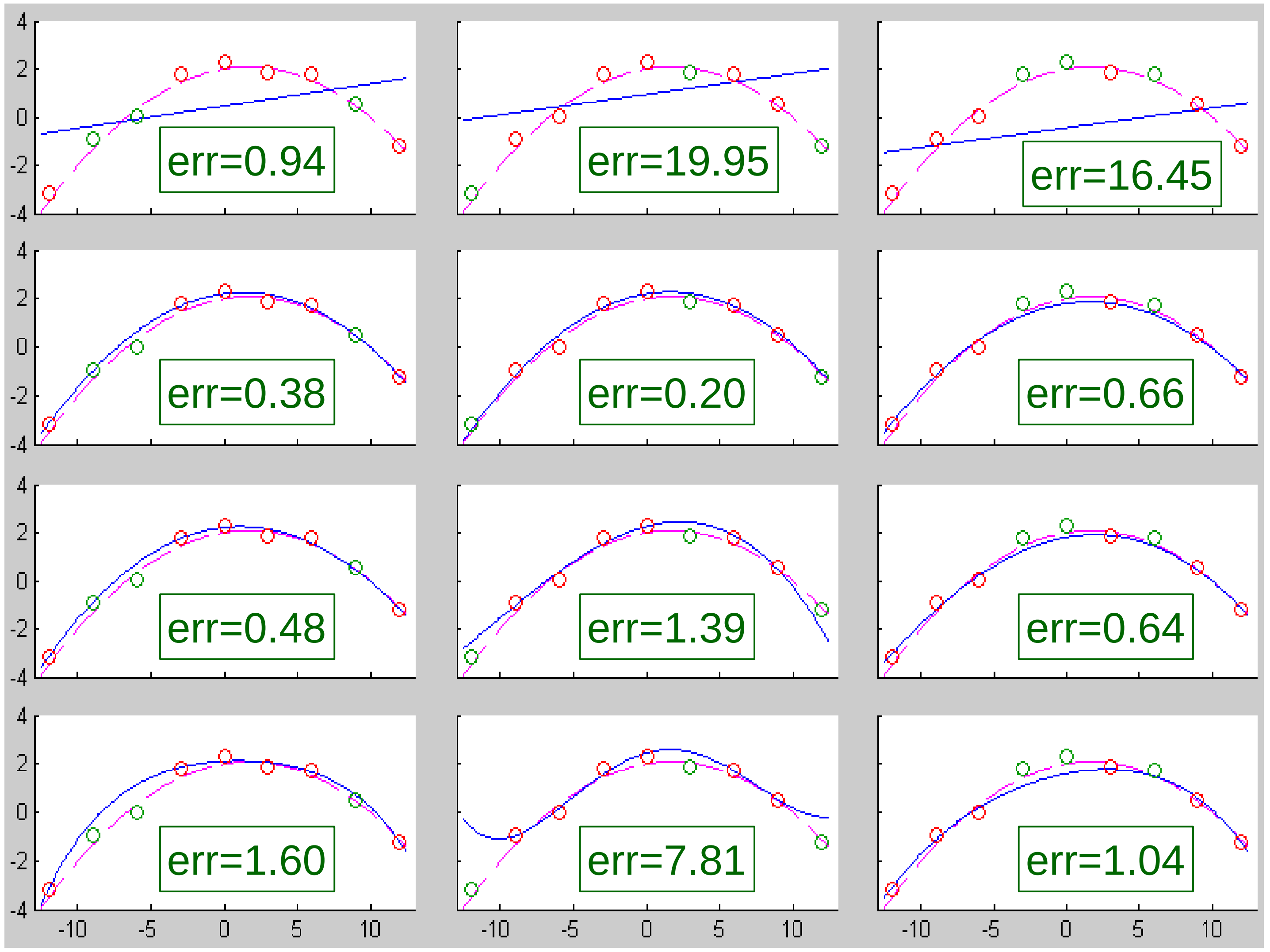
Model Hyperparameter Selection by Cross-Validation

Example: polynomial regression:

9 training samples → 3 subsets



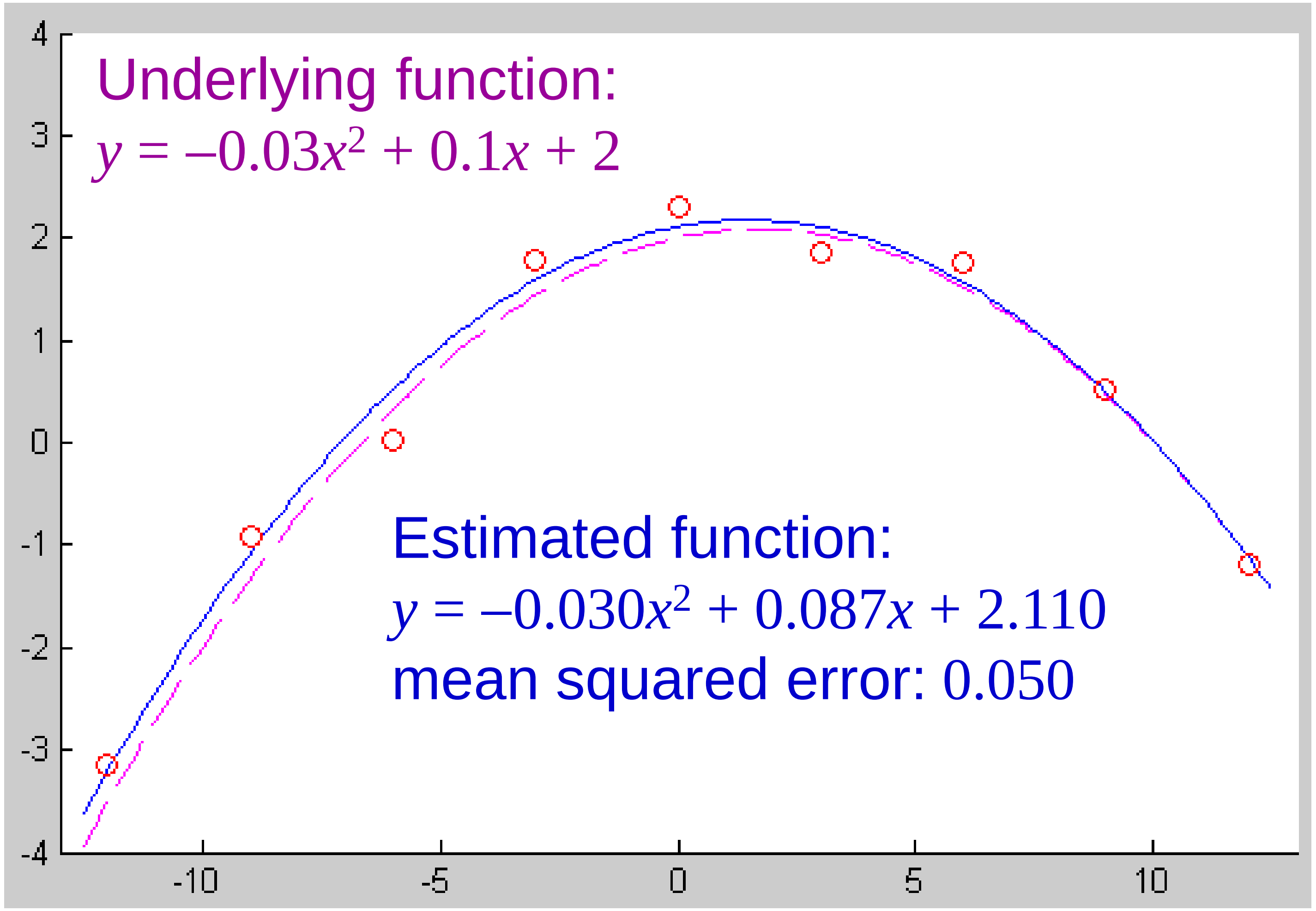
Model Hyperparameter Selection by Cross-Validation



order	training:2/3 validation:1	training:1/3 validation:2	training:1/2 validation:3	Mean Squared Error
1	err=0.94	err=19.95	err=16.45	4.15
2	err=0.38	err=0.20	err=0.66	0.14 ★
3	err=0.48	err=1.39	err=0.64	0.28
4	err=1.60	err=7.81	err=1.04	1.16

Model Selection by Cross-Validation

Finally, use the selected model and ALL the training samples for parameter optimization:

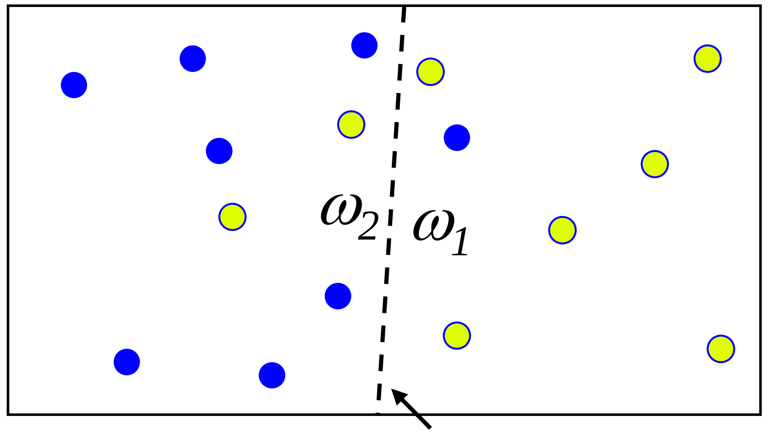


Classifier Evaluation

The most simple way to evaluate a classifier is to see the number of correct and incorrect classifications.

For a M-class problem with N samples to be classified, the **confusion matrix** is a MxM matrix, whose (i,j) element is the number of vectors that are actually in class ω_i and classified to class ω_j.

Example (2-class): ω_1 ● ω_2 ●



decision boundary

Confusion Matrix: $\begin{pmatrix} 6 & 2 \\ 1 & 7 \end{pmatrix}$

Correct classification rate
= trace(Confusion Matrix)$/N$

Confusion Matrix Examples

A typical confusion matrix for the Iris dataset:

$$\begin{array}{ccc} 50 & 0 & 0 \\ 0 & 46 & 4 \\ 0 & 0 & 50 \end{array}$$

Clothing color classification:

Pred / Real	Red	Orange	Yellow	Green	Blue	Pink	Purple	Brown	Gray	Black	White
Red	167	17	1	0	4	23	8	4	3	9	2
Orange	4	37	13	0	2	0	0	2	0	1	0
Yellow	3	1	87	5	0	3	0	5	3	1	3
Green	0	0	9	100	7	2	0	3	8	8	3
Blue	0	0	0	13	450	10	6	0	42	114	21
Pink	16	2	2	0	2	124	6	3	5	2	9
Purple	9	0	1	1	23	21	70	1	7	15	2
Brown	3	2	8	12	0	7	0	66	14	22	7
Gray	4	0	1	23	21	15	1	14	289	38	38
Black	10	1	0	15	44	15	15	5	49	903	9
White	1	0	2	7	29	26	2	4	52	9	322

Two-Class Confusion Matrix

Many two-class problems can be considered as "detection" problems where the classifier is expected to answer a "Yes/No" question for each sample, such as in a medical screening test.

Let class#1 be "No", class#2 be "Yes", confusion matrix be $\begin{pmatrix} TN & FP \\ FN & TP \end{pmatrix}$

Common metrics (in pairs) derived from the confusion matrix:

PD (probability of correct detection) = $TP / (TP + FN)$
FA (probability of false positive/alarm) = $FP / (TN + FP)$
Recall = **PD**
Precision = $TP / (TP + FP)$
Sensitivity = **PD**
Specificity = $TN / (TN + FP) = 1 -$ **FA**
PPV (positive predictive value) = **Precision**
NPV (negative predictive value) = $TN / (TN + FN)$

Two-Class Confusion Matrix

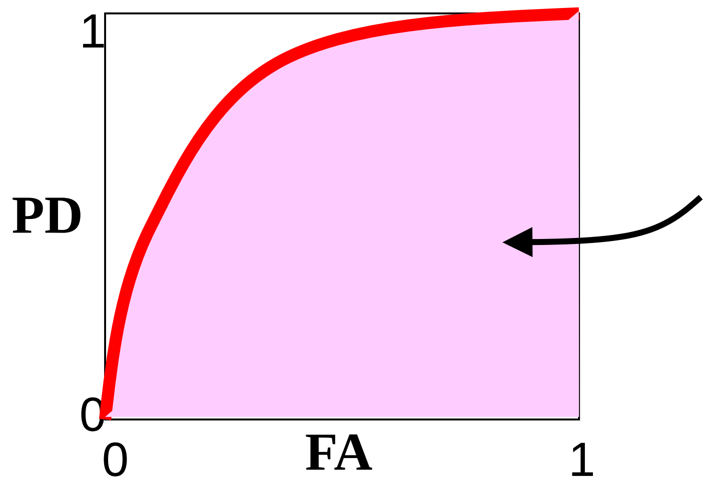
- A threshold / bias can be used to adjust how sensitive the classifier is to identify positive cases.
- This adjustment, while increasing one metric in a pair (e.g., **Recall**), is likely to decrease the other (e.g., **Precision**).
- **F1 measure** is one combined metric to allow for easier comparison between such paired classification results. It can also be used to select a “proper” threshold / bias.

$$F1 = \frac{2 * precision * recall}{precision + recall}$$

ROC Curves

Receiver Operating Characteristics (ROC) Curve is the plot of **PD** vs. **FA** at different threshold (bias) values.

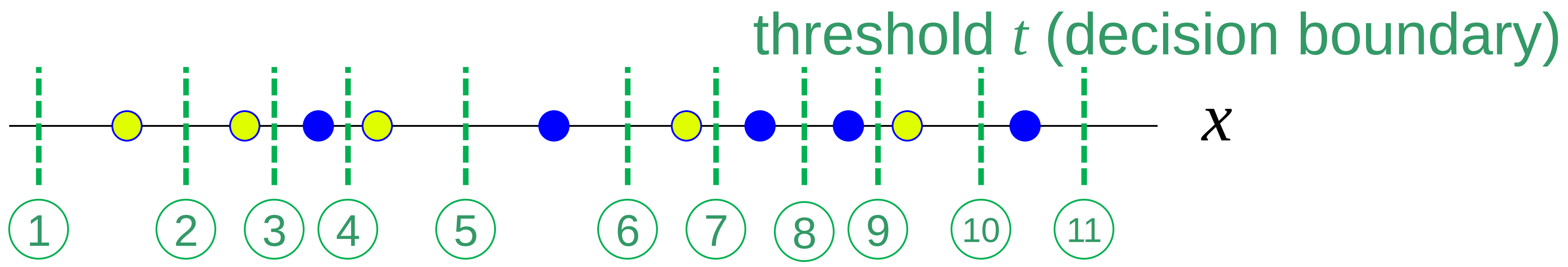
It allows the separation of the evaluation of different classification methods and/or settings from the choice of the threshold.

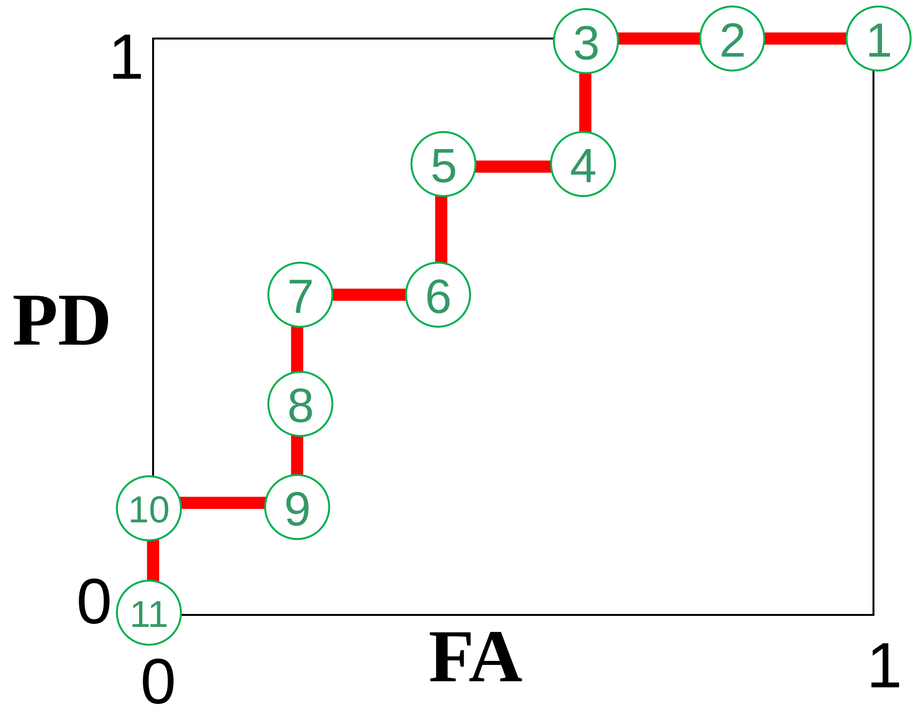


AUC or **AUROC** (area under the ROC curve): The larger, the better.

ROC Curves

Example (1-D): ○ negative ● positive





Classifier Evaluation w/ Cross-Validation

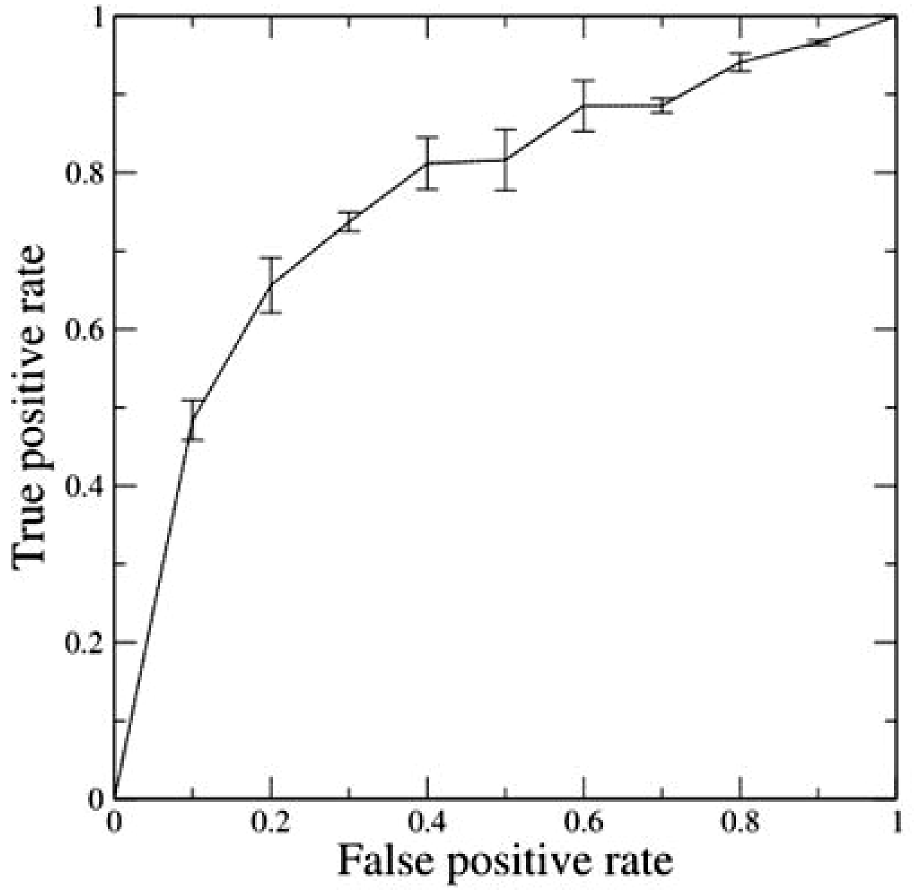
- Confusion matrix: Add the validation-subset confusion matrices from all the trials. Each sample is included exactly once. Metrics based on the confusion matrix, such as **PD** and **FA**, can then be computed.

- ROC curve (two-class problems):
  - Method 1: Just collect the outputs of the validation subsets from all the trials and draw a single ROC curve.
  - Method 2: Compute a ROC curve for each trial. We can average the curves at any given **FA** rate. The standard deviation gives us an estimation of the uncertainty of **PD** at any **FA** rate.
  - Method 3: Use a common set of thresholds to compute separate **PD** and **FA** values for all the validation subsets. This produces a single ROC curve with uncertainties for both **PD** and **FA**.

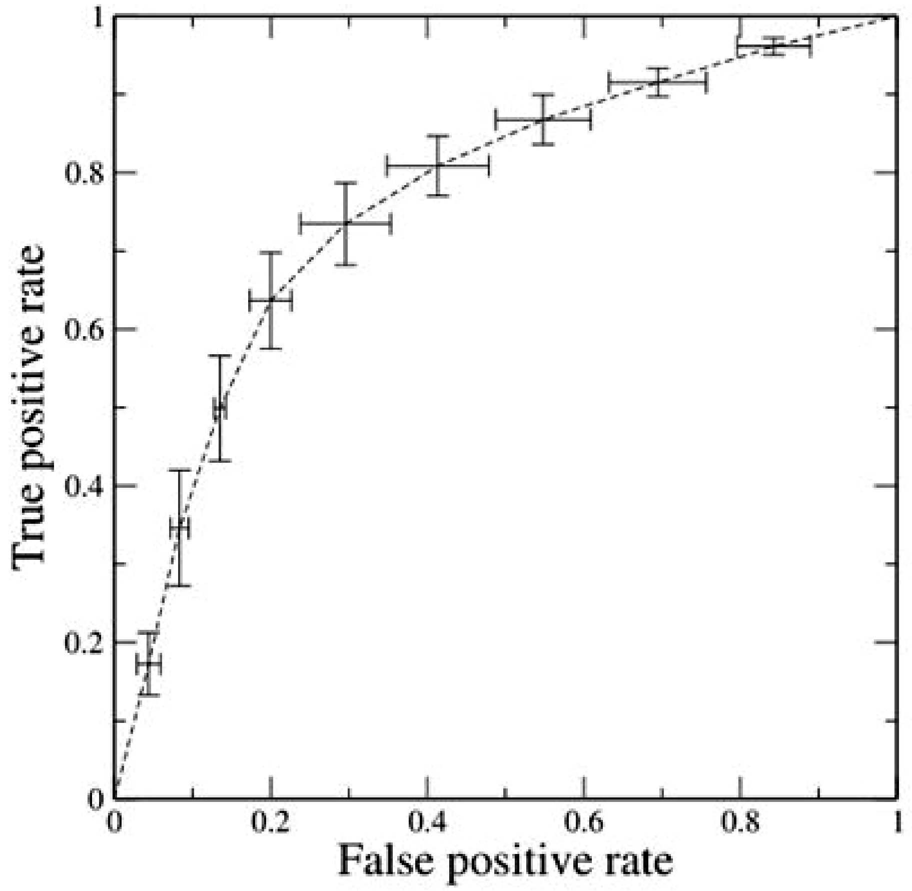
Classifier Evaluation w/ Cross-Validation

Examples of generating ROC curves with cross-validation:

(method 2)



(method 3)

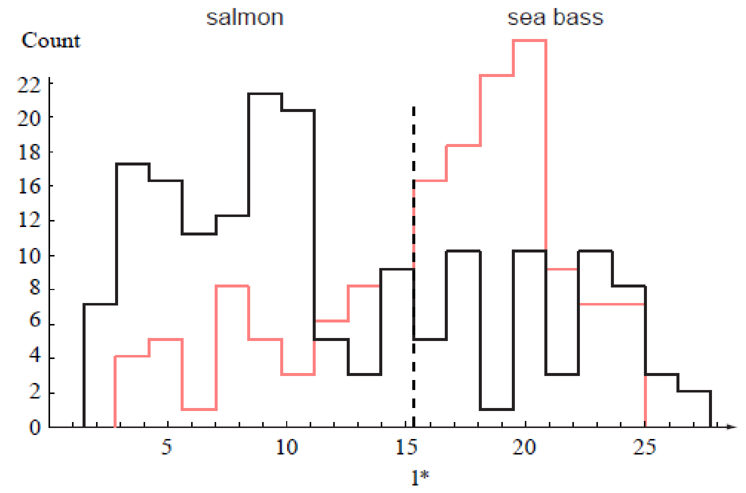


https://slideplayer.com/slide/7028153/

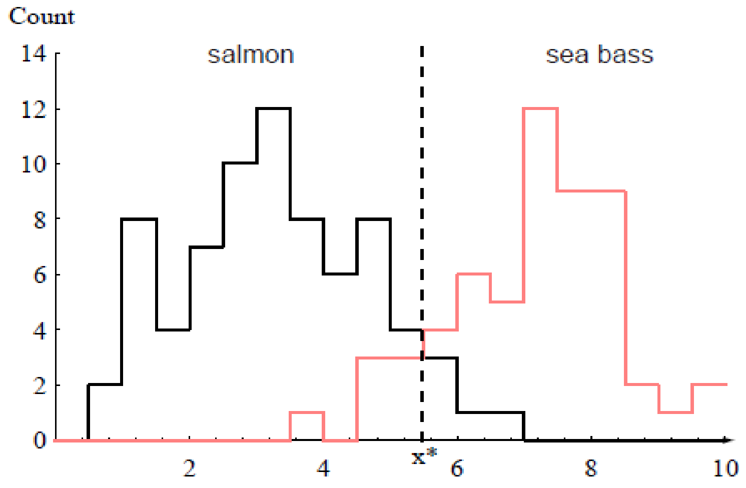
Example: Features for Classification

This is an example from DHS (a classic textbook on classification): To classify a fish as a sea bass or a salmon.

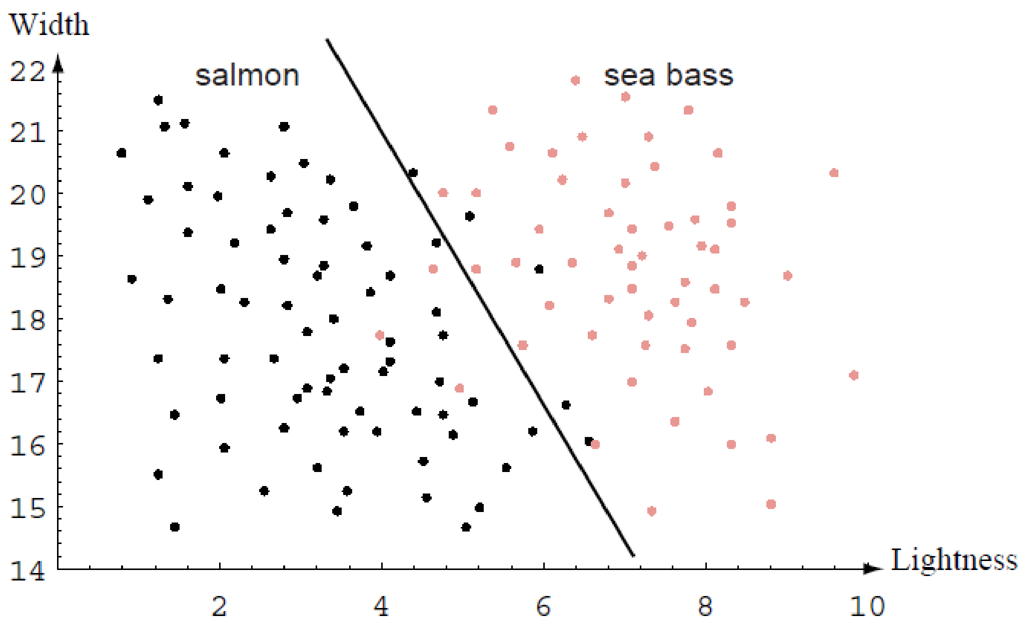
Feature: Width



Feature: Lightness



Feature Space



A Look at Features

- Features are the set of values we use to represent the samples.
- Features are domain specific.
- Using the very famous Iris dataset as an example:
  - Data: Each sample represents an iris flower.
  - Four features for each flower: sepal length, sepal width, petal length, petal width.

Iris sestosa				Iris versicolor				Iris virginica			
Sepal Leng.	Sepal Width	Petal Leng.	Petal Width	Sepal Leng.	Sepal Width	Petal Leng.	Petal Width	Sepal Leng.	Sepal Width	Petal Leng.	Petal Width
5.1	3.5	1.4	0.2	7.0	3.2	4.7	1.4	6.3	3.3	6.0	2.5
4.9	3.0	1.4	0.2	6.4	3.2	4.5	1.5	5.8	2.7	5.1	1.9
4.7	3.2	1.3	0.2	6.9	3.1	4.9	1.5	7.1	3.0	5.9	2.1

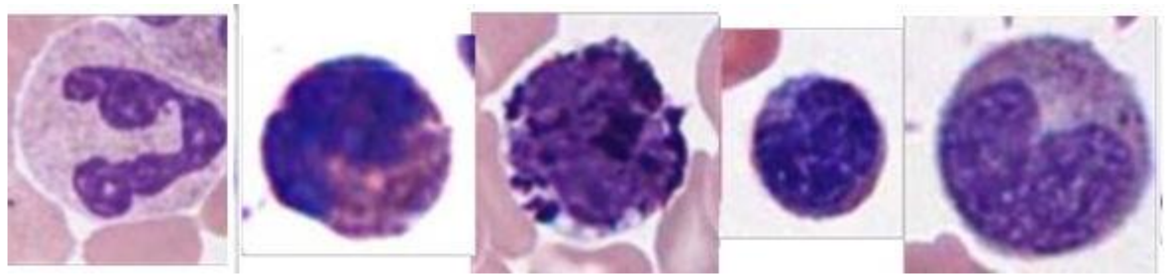
and more …

A Look at Features

- Some features are from direct measurements, or are otherwise straightforward:
  - Features for Iris
  - A person's age, gender, etc.
- For many problems, the "raw" features are difficult to use. Examples:
  - Images
  - Speech; audio signals
  - Trajectory, such as online handwriting recognition
  - Text
  - ...

A Look at Features

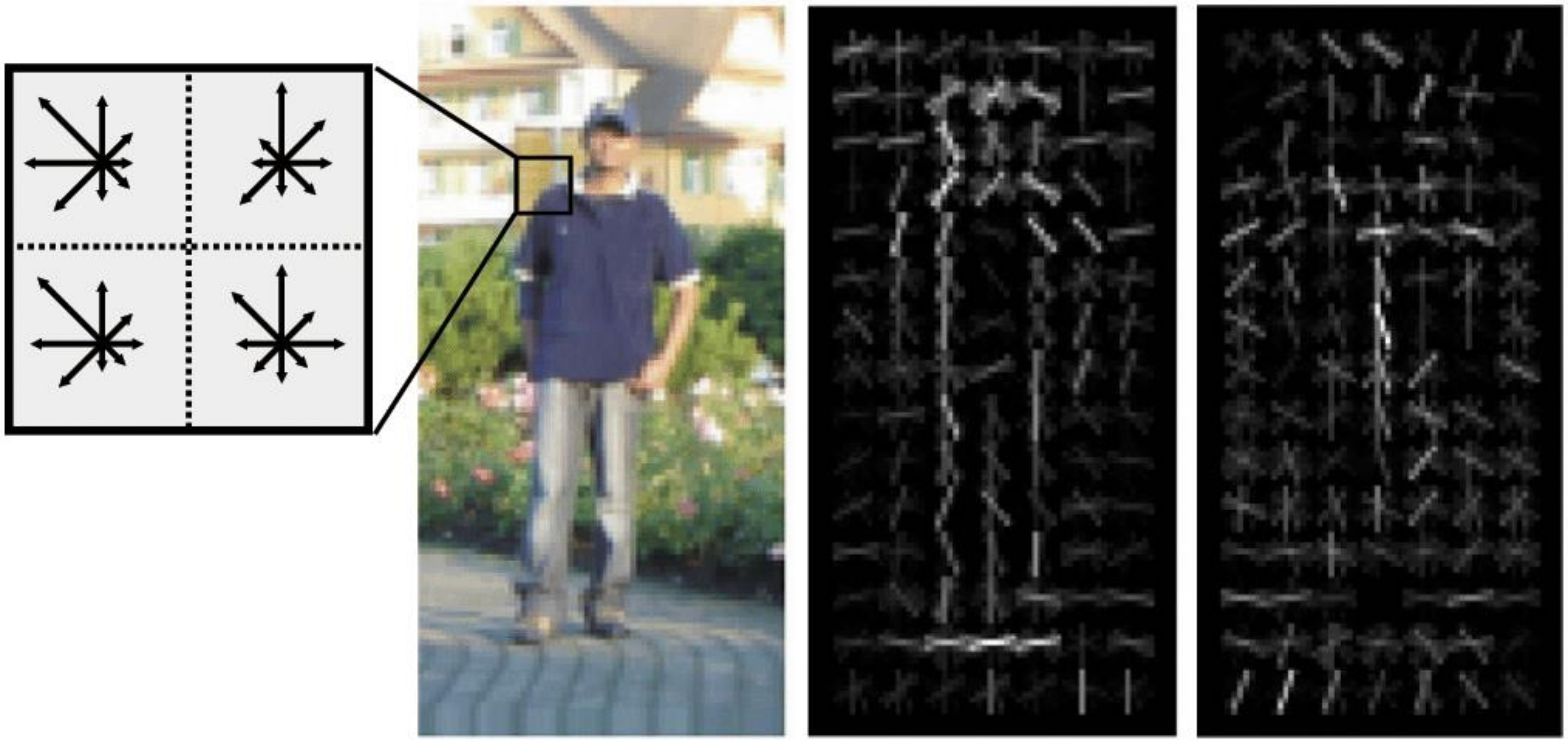
- For problems where it is impractical to use the "raw features", we need to find ways to extract meaningful and manageable "derived features" from the raw features, and then use these derived features for our classification or regression tasks.
- Example:



size (area)
ratio of nuclei
shape features:
    moments, etc.
    etc.
texture features:
    DCT, LBP, etc.
…

A Look at Features

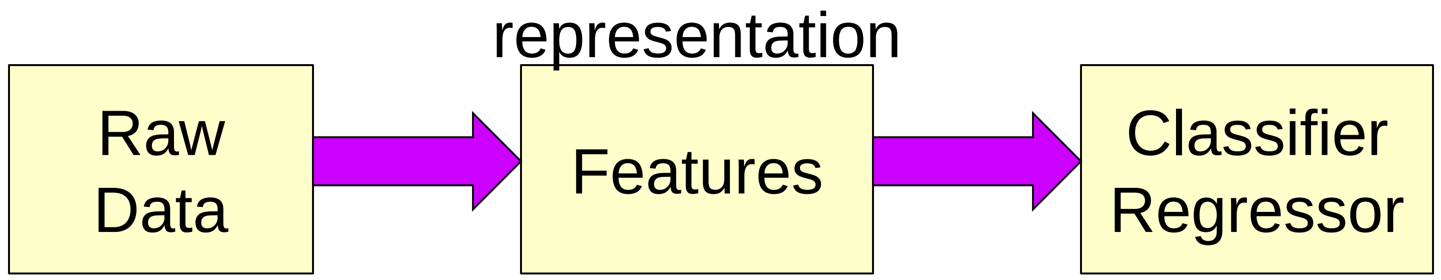
- Another classic example: The HOG (histogram of oriented gradients) features for pedestrian detection. This was the state-of-the-art before CNNs became practical.



http://doi.ieeecomputersociety.org/10.1109/TPAMI.2009.122

A Look at Features

- The basic steps:



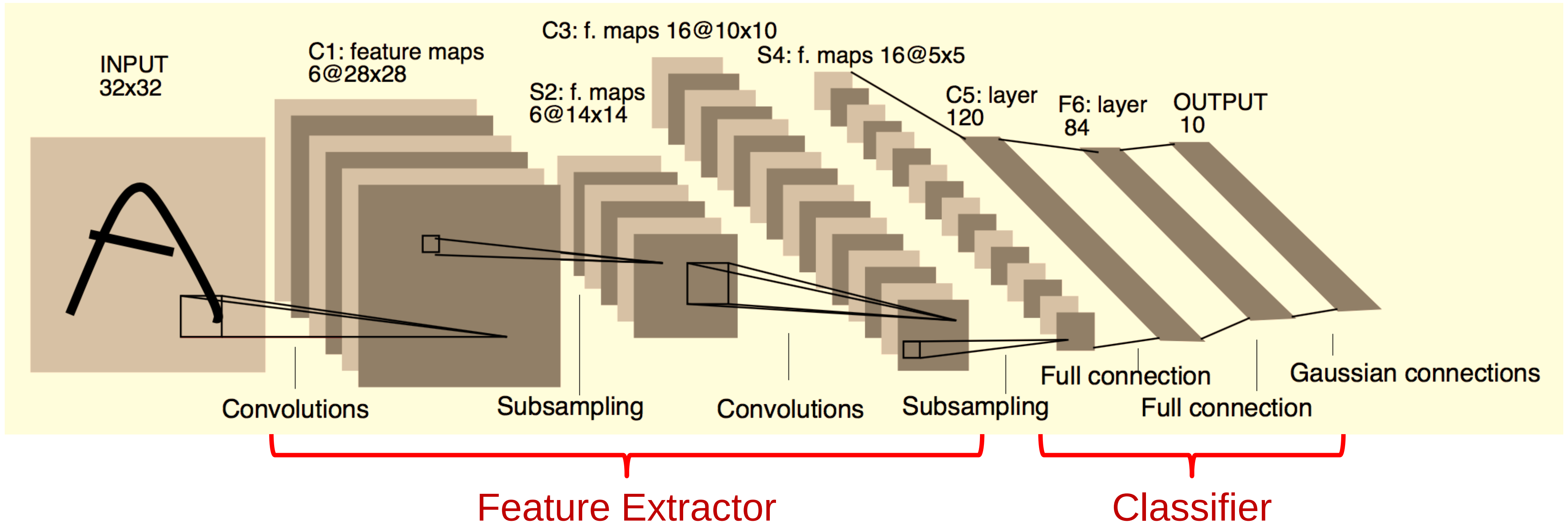
- For each type of raw data, there are many possible ways of designing "derived features".
- Good features are usually more important than good classifiers of regressors for solving a particular problem. Examples:
  - Haar features for face detection
  - HOG for pedestrian detection
  - TF-IDF for text document classification
  - MFCC for audio signals

A Look at Features

- However, even good hand-crafted features are never good enough.
  - They can not capture all the useful information.
  - They can only represent low-level information (easier to define and compute) well.
  - It is very difficult to design features for high-level and semantically rich information.
    - To understand this, consider the HOG features for pedestrian detection. They do not provide us with information regarding whether and where the head, arms, or legs are detected, or their relative positions.

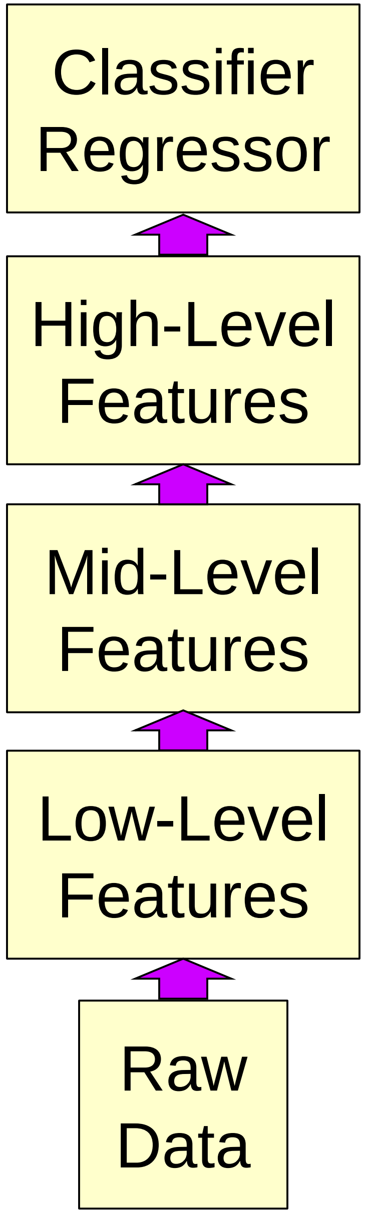
LeNet: A Basic Convolutional Neural Network

LeNet: A small CNN example for image classification:

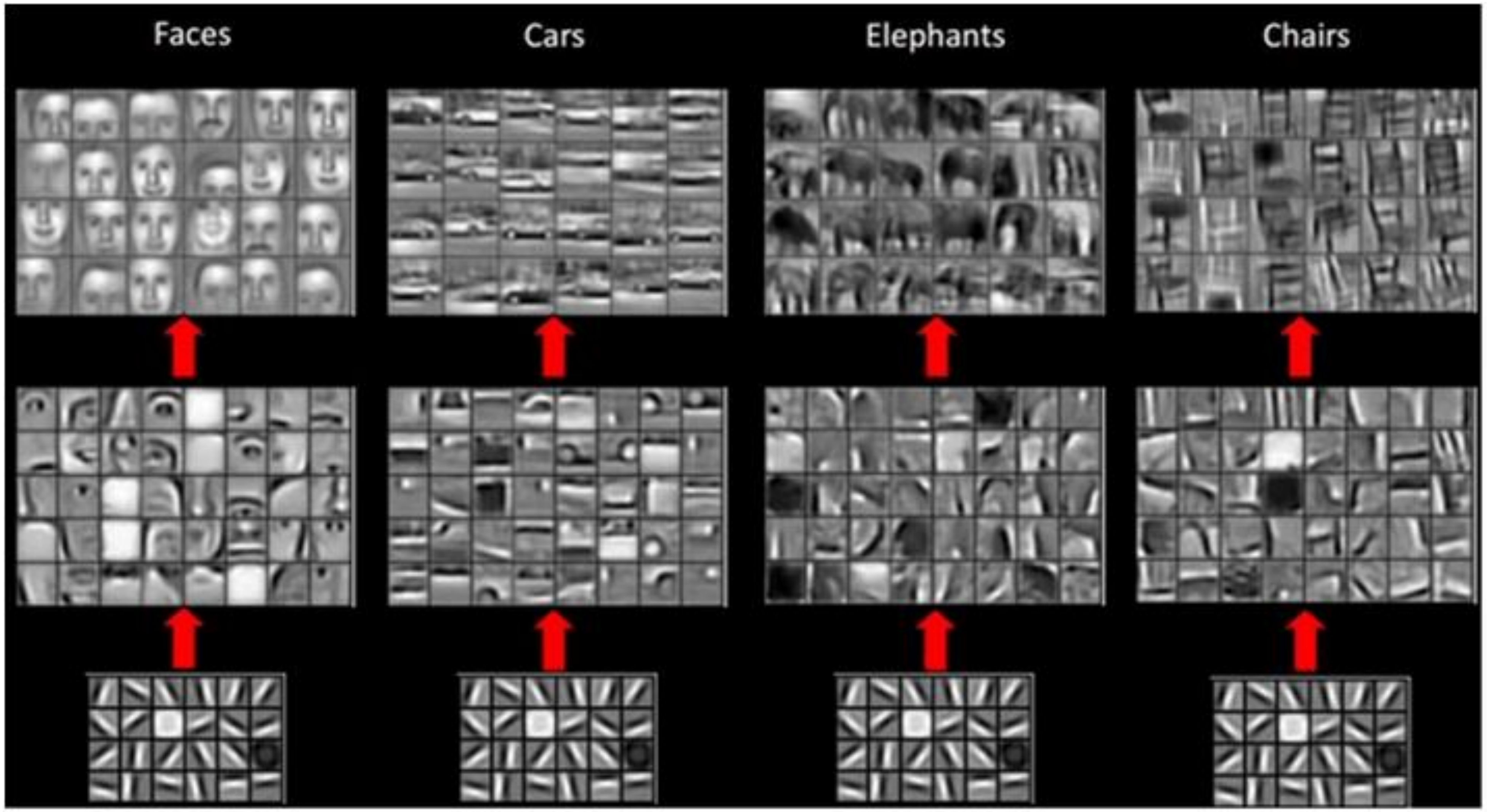


- The last part (fully connected layers) is actually the classifier. It is just like a traditional multi-layer perceptron.
- The CNN can also work with other classifiers (e.g., SVMs), but using FC layers is common as the training can be done end-to-end.

Convolutional Neural Networks as Feature Extractors



This is how our brain processes visual inputs.



https://www.researchgate.net/figure/Pictorial-representation-of-features-in-3-layers-of-a-CNN-34-Notice-the-increasing_fig2_308883811