CSci 8980

Basics of Machine Learning and Deep Learning (Part I)

Machine Learning

- Tom Mitchell:
  - An algorithm that is able to learn from data

- Learning?
  - A computer program is said to learn from experience E with respects to some class of tasks T and performance measure P, if its performance at tasks T, as measured by P, improves with experience E.

Machine Learning

- Task types
  - Classification: k categories
  - Regression: predict a value
  - Structured outputs: decompose/annotate output
  - Anomaly detection

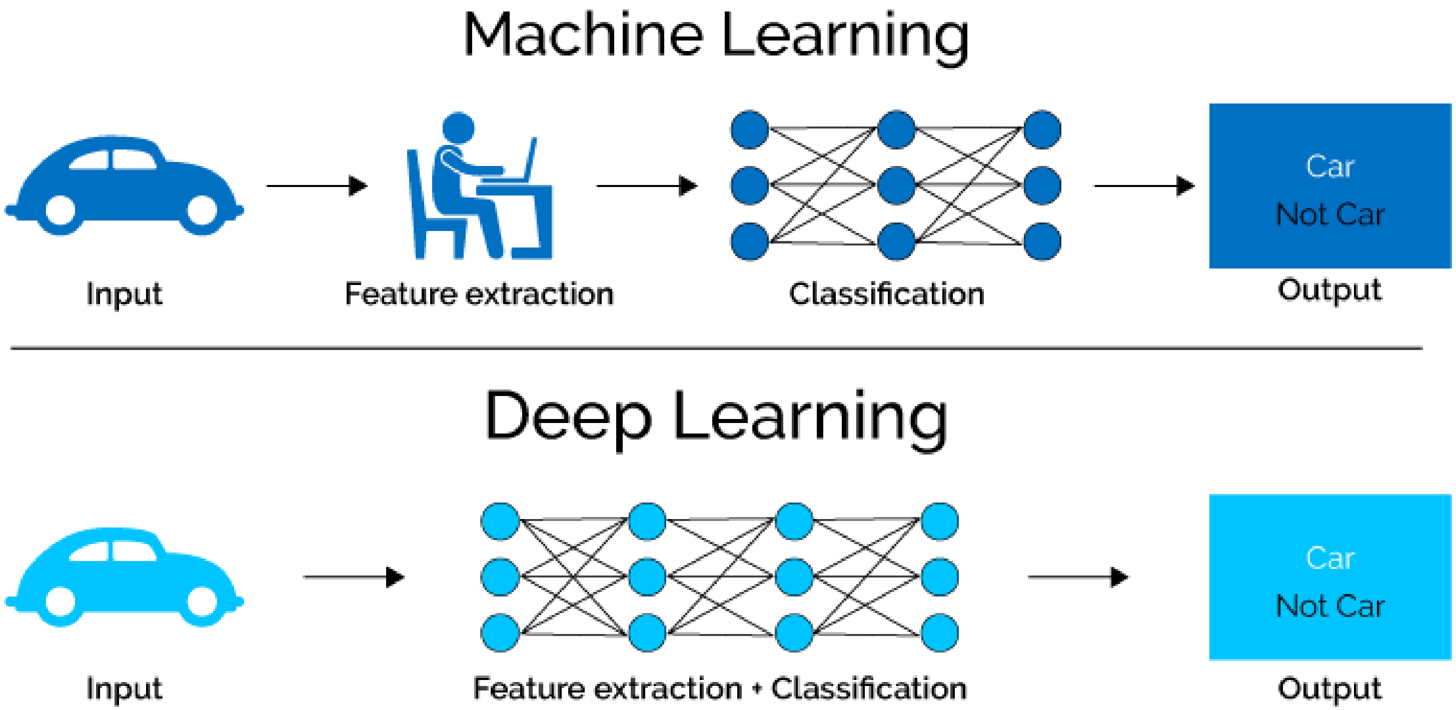
- Experience E; samples x
  - Supervised: labelled outputs => p (y|x)
  - Unsupervised: non-labelled outputs => p(x)
  - Reinformement learning: seq. experience x_1 x_2 …

Machine Learning

- Input is represented by features
  - image: pixels, color, …
  - game: move right
- Extract features from inputs to solve a task
  - Classic ML: human provides features
  - DL: system learns representation (i.e. features)
    - From simpler to complex (layers of simpler)

DL vs. ML

- Learning representations and patterns of data
- Generalization (failure of classic AI/ML)
- Learn (multiple levels of) representation by using a hierarchy of multiple layers



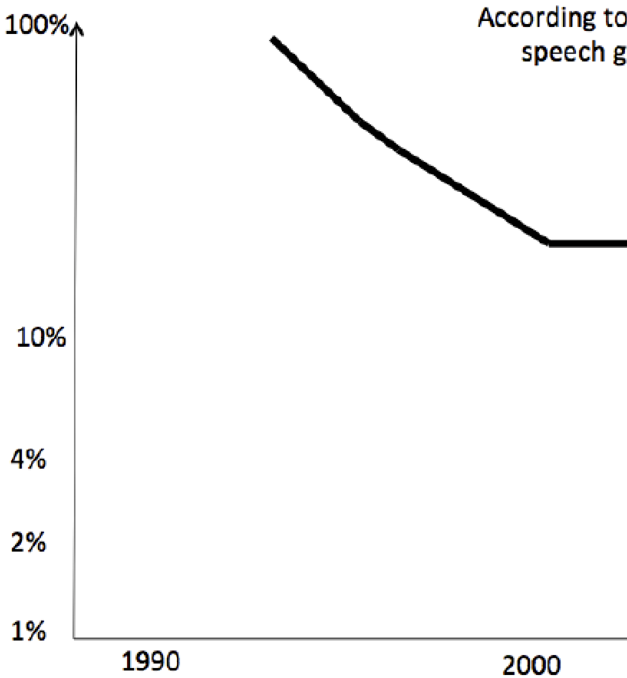
https://www.xenonstack.com/blog/static/public/uploads/media/machine-learning-vs-deep-learning.png

Why is DL useful?

- Manual features
  - over-specified, incomplete and take a long time to design and validate
- Learned Features are easy to adapt, fast to learn
- Deep learning provides a universal, learnable framework for representing world information

In ~2010 DL started outperforming other ML techniques: e.g. speech, NLP, …

Big Win in Vision

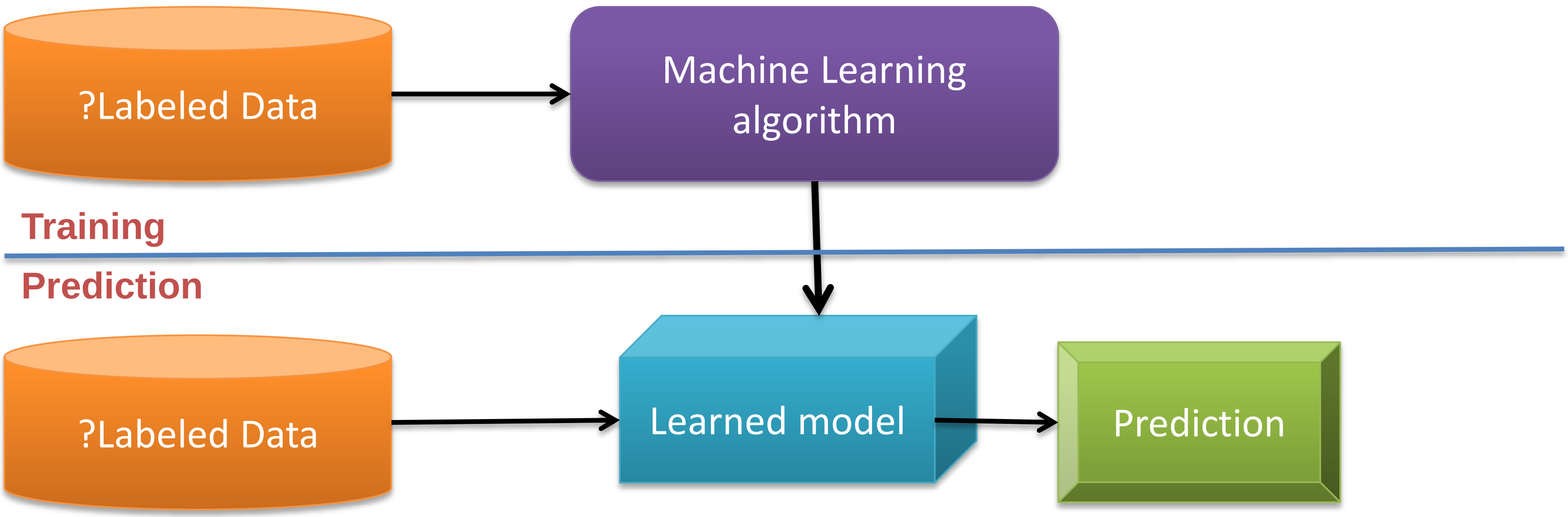


Deep Learning in **Speech Recognition**

ImageNet: The "computer vision World Cup"

Machine Learning Basics

Machine learning is a field of computer science that gives computers the ability to **learn without being explicitly programmed**



Methods that can learn from and make predictions on data

ML in a Nutshell

- Every machine learning algorithm has three components:
  - **Representation**
  - **Evaluation**
  - **Optimization**

(Model) Representation

- Decision trees
- Sets of rules / Logic programs
- Instances
- Graphical models (Bayes/Markov nets)
- Neural networks
- Support vector machines
- Model ensembles
- Logistic regression
- Randomized Forests
- Boosted Decision Trees
- K-nearest neighbor
- Etc.

Evaluation

- Differ between supervised and unsupervised learning
  - Accuracy
  - Precision and recall
  - Mean squared error
  - Max Likelihood
  - Posterior probability
  - Cost / Utility
  - Entropy
  - Etc.

Optimization

- Combinatorial optimization
  - E.g.: Greedy search
- Convex optimization
  - E.g.: Gradient descent
- Constrained optimization
  - E.g.: Linear programming

Types of Learning

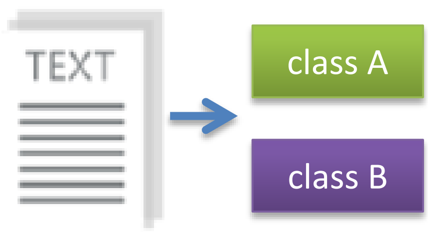
- **Supervised learning**
  - Training data includes desired outputs
  - Prediction, Classification, Regression
- **Unsupervised learning**
  - Training data does not include desired outputs
  - Clustering, Probability distribution estimation
  - Finding association (in features), Dimension reduction
  - Best representation of data
- **Reinforcement learning**
  - Rewards from sequence of actions
  - Seq. decision making (robot, chess, games)

Types of Learning: examples

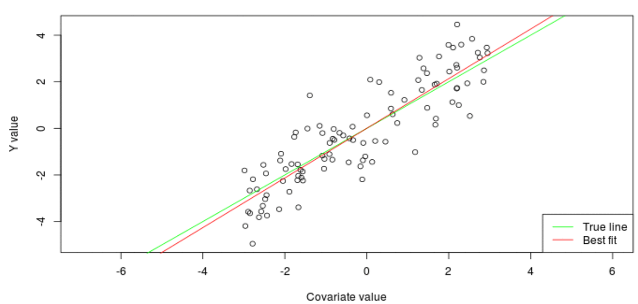
Supervised: Learning with a **labeled training** set
Example: email *classification* with already labeled emails

Unsupervised: Discover **patterns** in **unlabeled** data
Example: *cluster* similar documents based on text

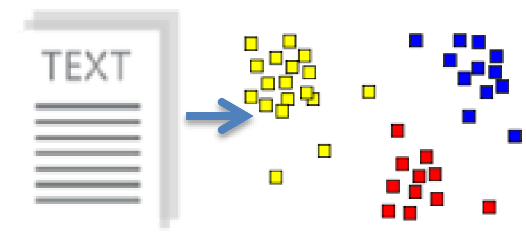
Reinforcement learning: learn to **act** based on **feedback/reward**
Example: learn to play Go, reward: *win or lose*



Classification



Regression

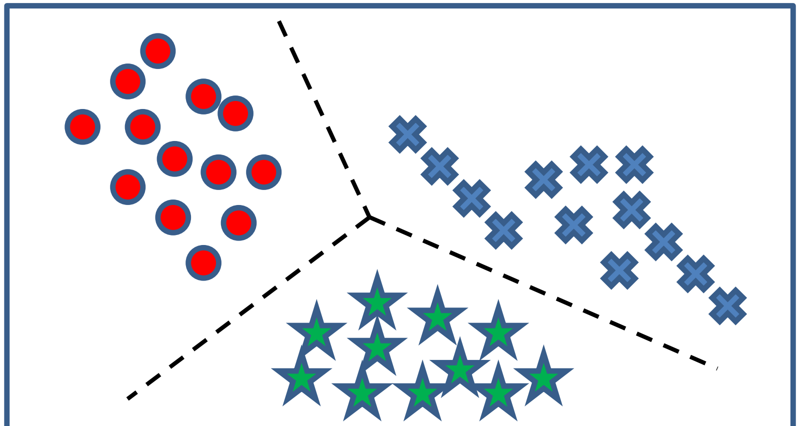


Clustering

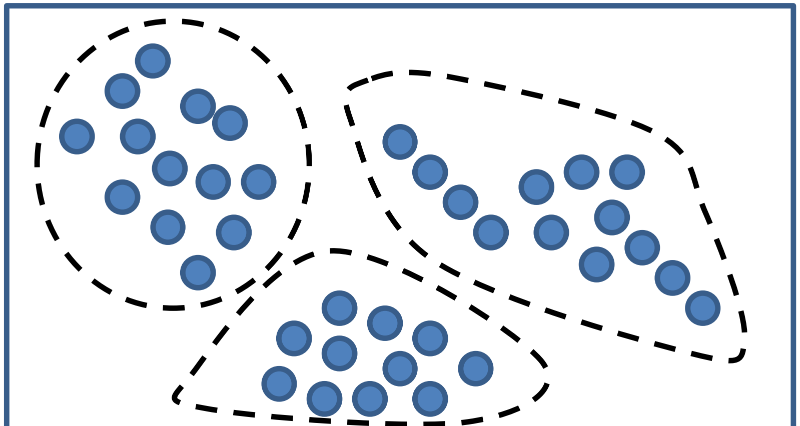
Anomaly Detection
Sequence labeling
…

http://mbjoseph.github.io/2013/11/27/measure.html

Comparison



Supervised learning



Unsupervised learning

Learning techniques

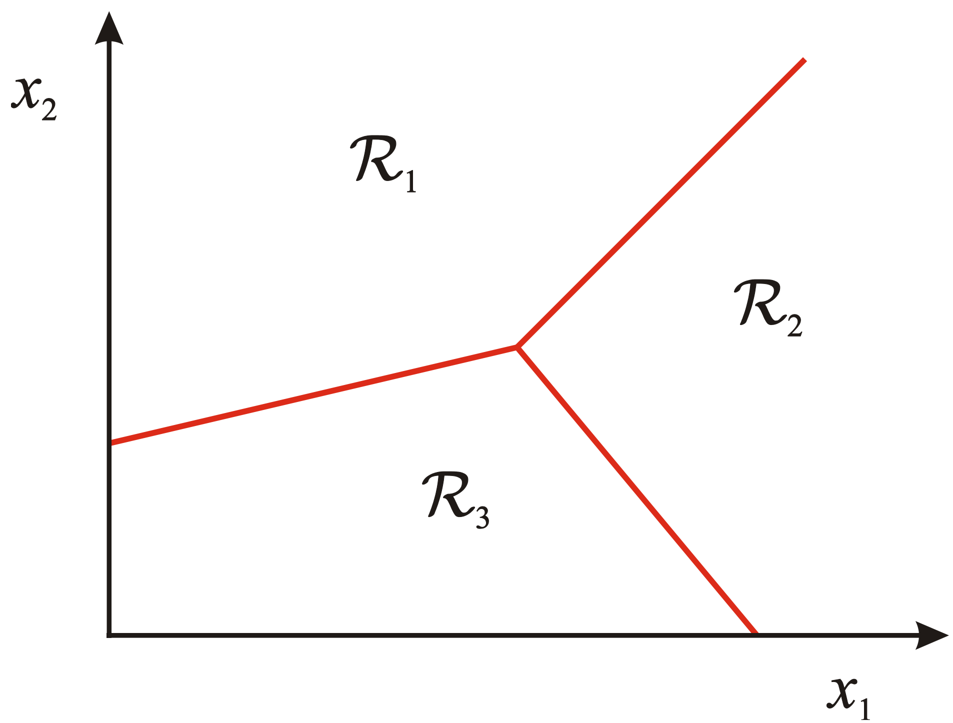
- Supervised learning categories and techniques
  - **Linear classifier** (numerical functions)
    - Works well: output depends on many features
  - **Parametric** (probabilistic functions)
    - Work wells: limited data, but with assumptions about function
    - Naïve Bayes, Gaussian discriminant analysis (GDA), Hidden Markov models (HMM), …
  - **Non-parametric** (Instance-based functions)
    - Works well: Lot of data, no prior knowledge
    - *K*-nearest neighbors, Kernel regression, Kernel density estimation, …

Learning techniques

- Unsupervised learning categories and techniques
  - **Clustering**
    - K-means clustering
    - Spectral clustering
  - **Density Estimation**
    - Gaussian mixture model (GMM)
    - Graphical models
  - **Dimensionality reduction**
    - Principal component analysis (PCA)
    - Factor analysis

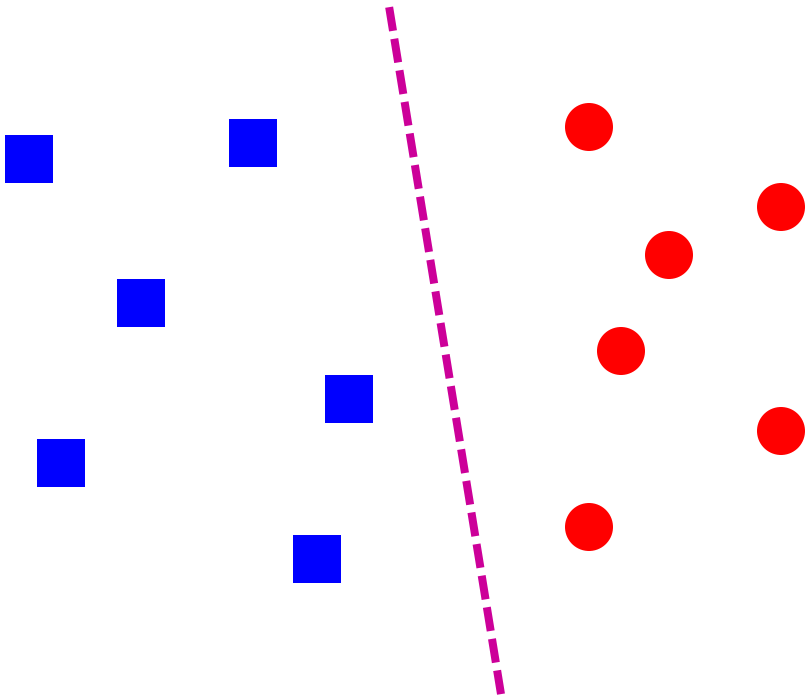
Classification

- Assign input vector to one of two or more classes
- Any decision rule divides input space into *decision regions* separated by *decision boundaries*



Slide credit: L. Lazebnik

Linear Classifier

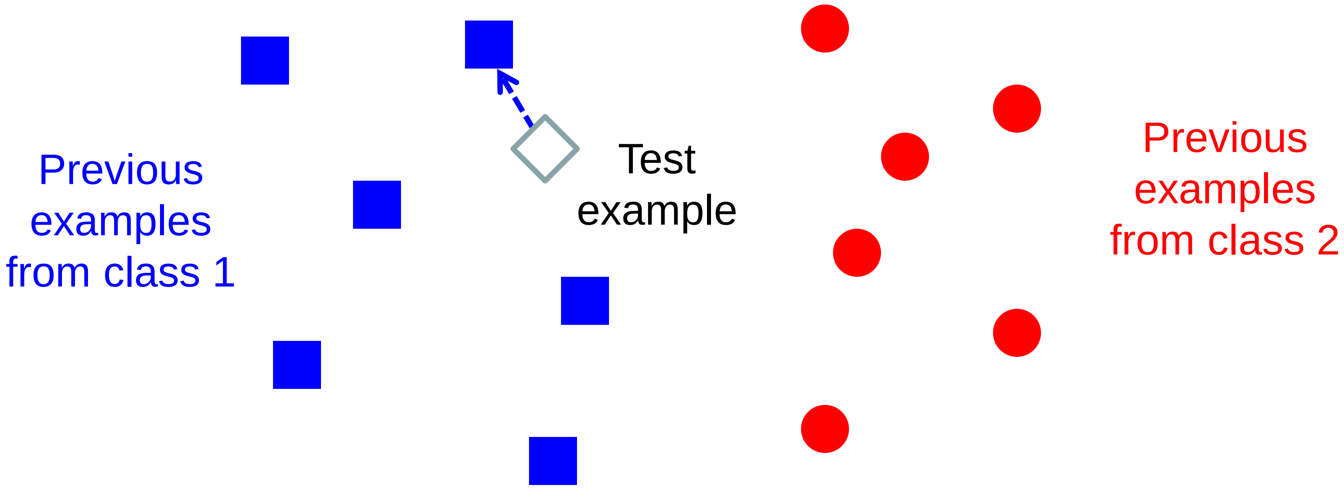


- Find a *linear function* to separate the classes:

$$f(\mathbf{x}) = \mathrm{sgn}(\mathbf{w} \cdot \mathbf{x} + b)$$

Slide credit: L. Lazebnik

Classifiers: Nearest neighbor

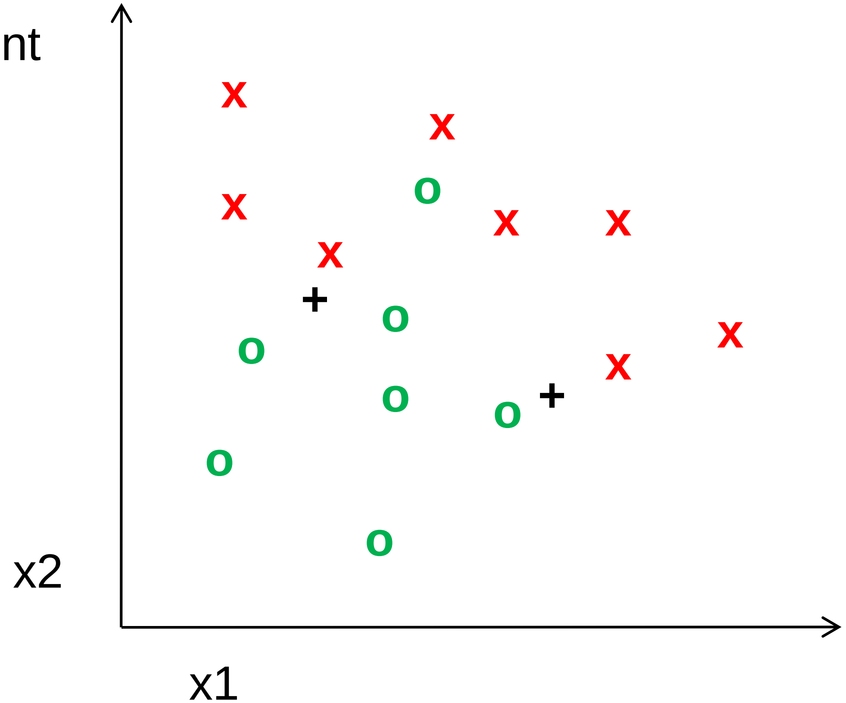


f(**x**) = label of the example nearest to **x**

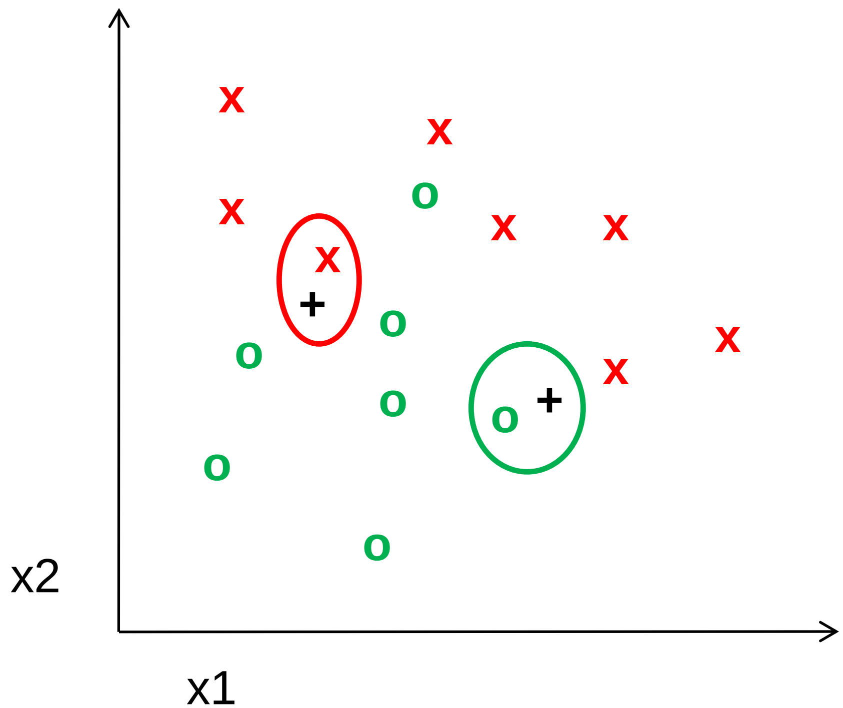
- All we need is a distance function for our inputs
- No training required!

K-nearest neighbor

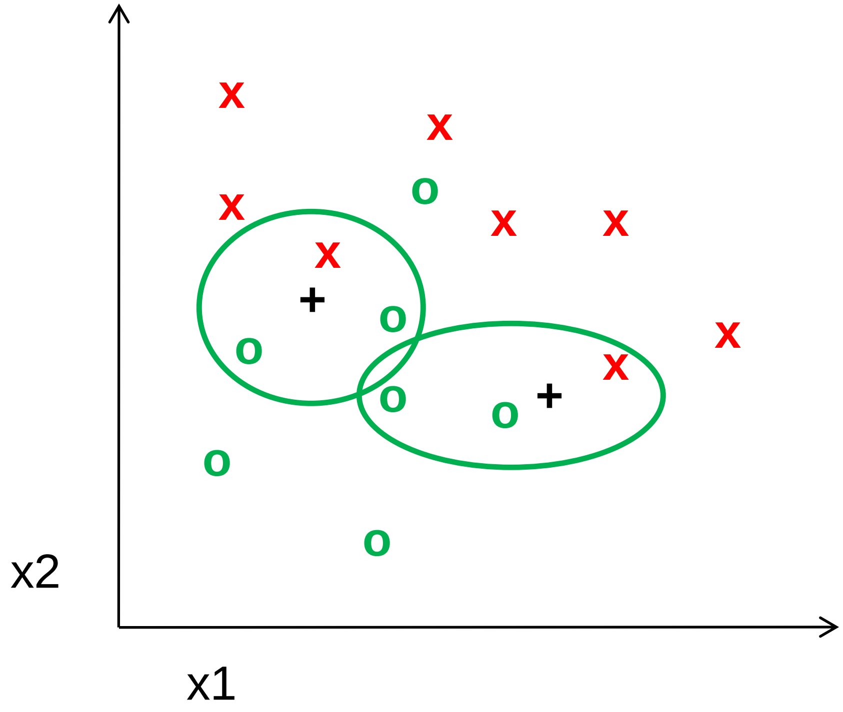
Assign label of nearest training data point to each test data point



1-nearest neighbor

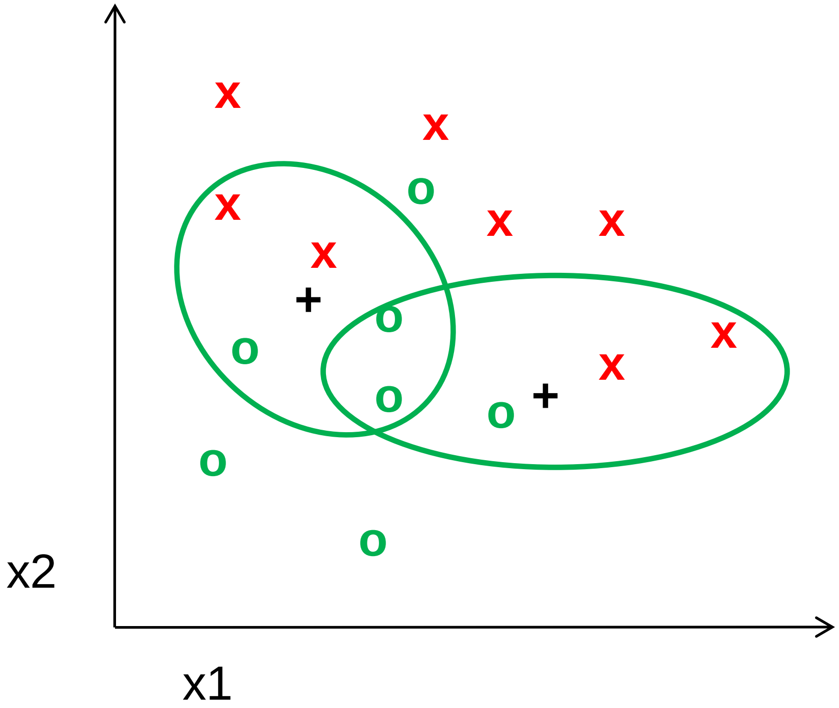


3-nearest neighbor



5-nearest neighbor

- Cannot discriminate between features
  - Poor generalization if small "training set"



Supervised Learning Goal

$$y = f(\mathbf{x})$$

output

prediction function

feature(s) or inputs

- **Training:** given a *training set* of labeled examples $\{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)\}$, estimate f by minimizing the prediction error on the training set
- **Testing:** apply f to a never before seen *test example* $\mathbf{x}$ and output the predicted value $y = f(\mathbf{x})$

Slide credit: L. Lazebnik

Example

- Apply a prediction function to a feature representation of the image to get the desired output:

f() = "apple"

f() = "tomato"

f() = "cow"

Slide credit: L. Lazebnik

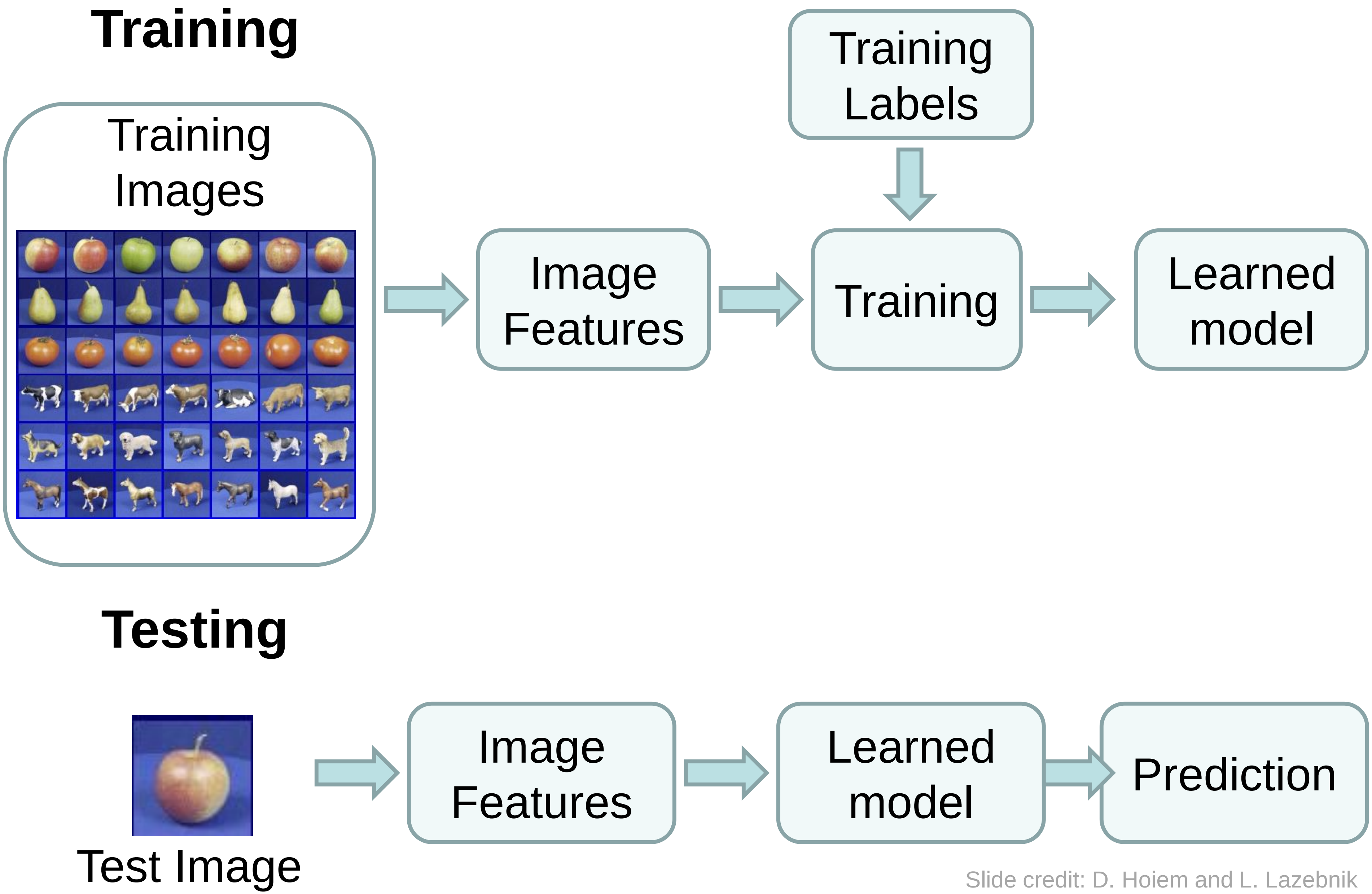
Generalization

Training set (labels known)

Test set (labels unknown)

- How well does a learned model generalize from the data it was trained on to a new test set?

Steps



Slide credit: D. Hoiem and L. Lazebnik

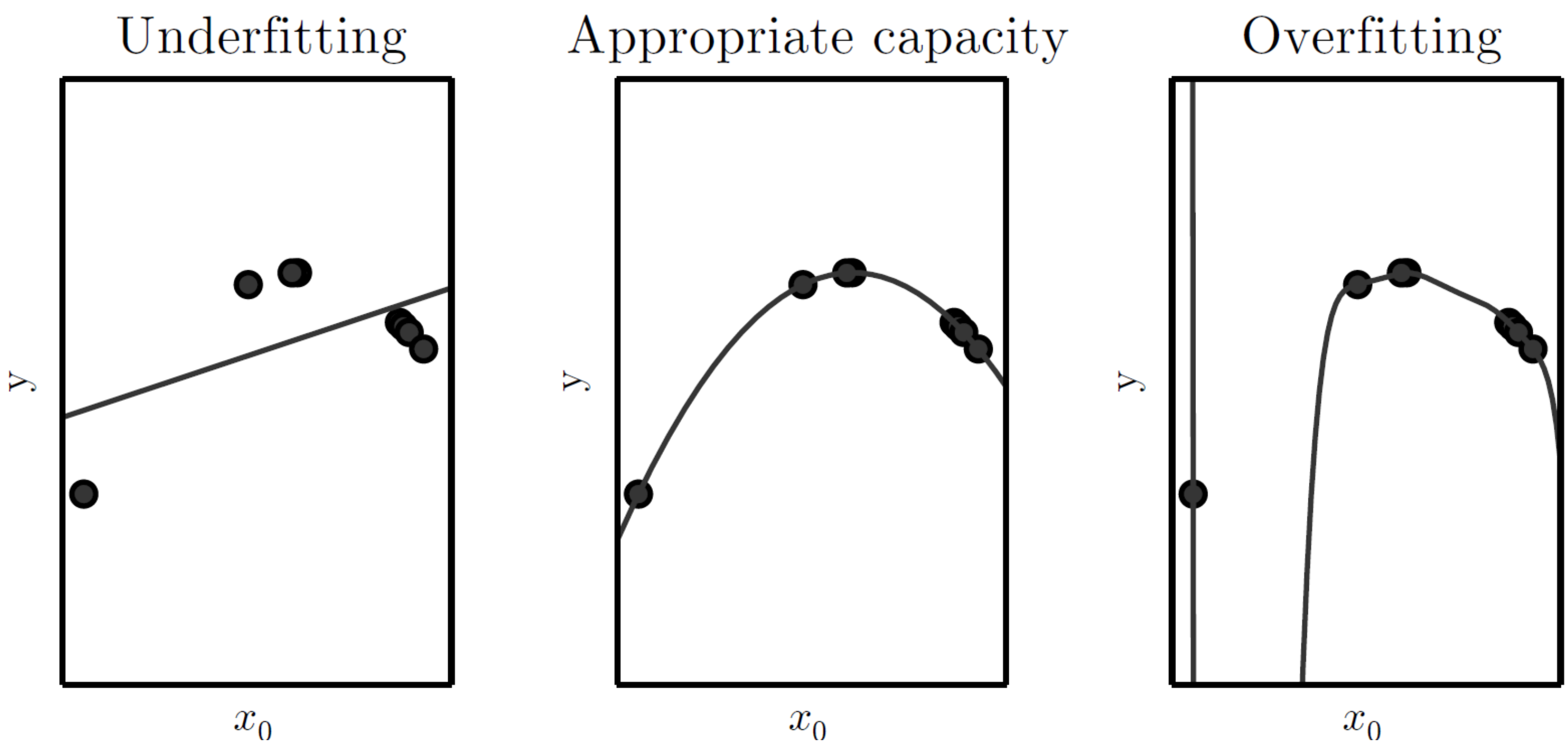
Training and testing

- Training is the process of making the system able to learn/generalize

- No free lunch rule:
  - Training set and testing set come from the same distribution
  - No universal ML algorithm!
  - Need to make some assumptions

Under{Over} fitting

- ML algorithm must perform well on unseen inputs “generalization”
  - Training error – run training data back on model
  - Testing error – error on new data
- Underfit
  - High training error
- Overfit
  - Gap between training and testing error too large

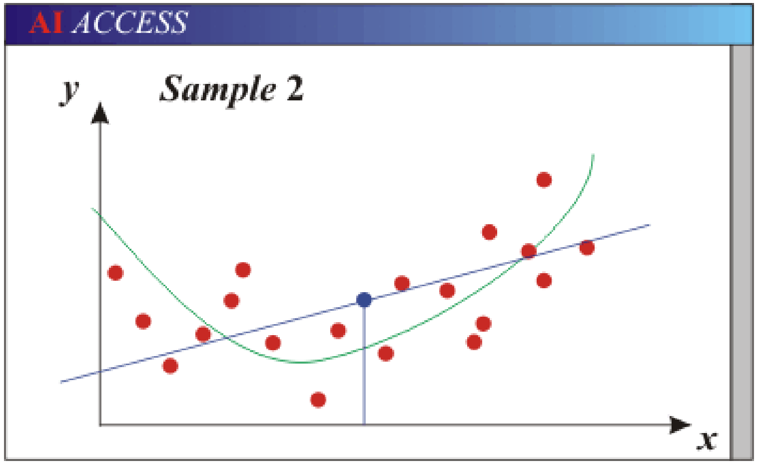

Underfitting
Appropriate capacity
Overfitting
y
x_0

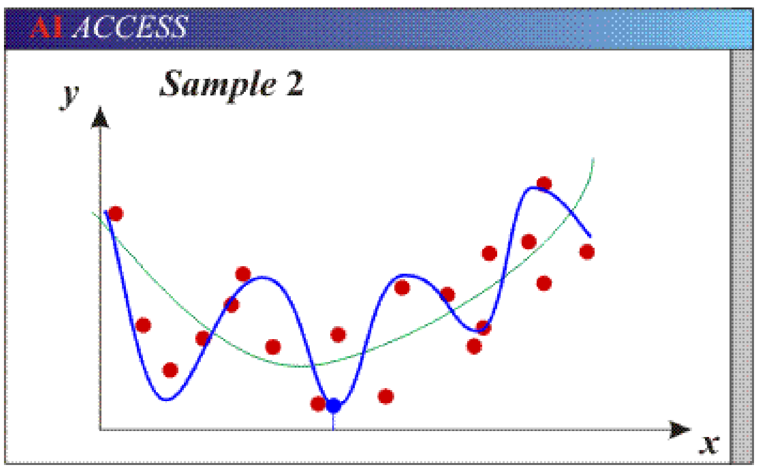
Generalization

- Components of generalization error
  - **Bias:** how much the average model over all training sets differ from the true model?
    - Error due to simplifications made by the model
  - **Variance:** how much models estimated from different training sets differ from each other
- **Underfitting:** model is too “simple” to represent all the relevant class characteristics
  - High bias and low variance
  - High training error and high test error
- **Overfitting:** model is too “complex” and fits irrelevant characteristics (noise) in the data
  - Low bias and high variance
  - Low training error and high test error

Slide credit: L. Lazebnik

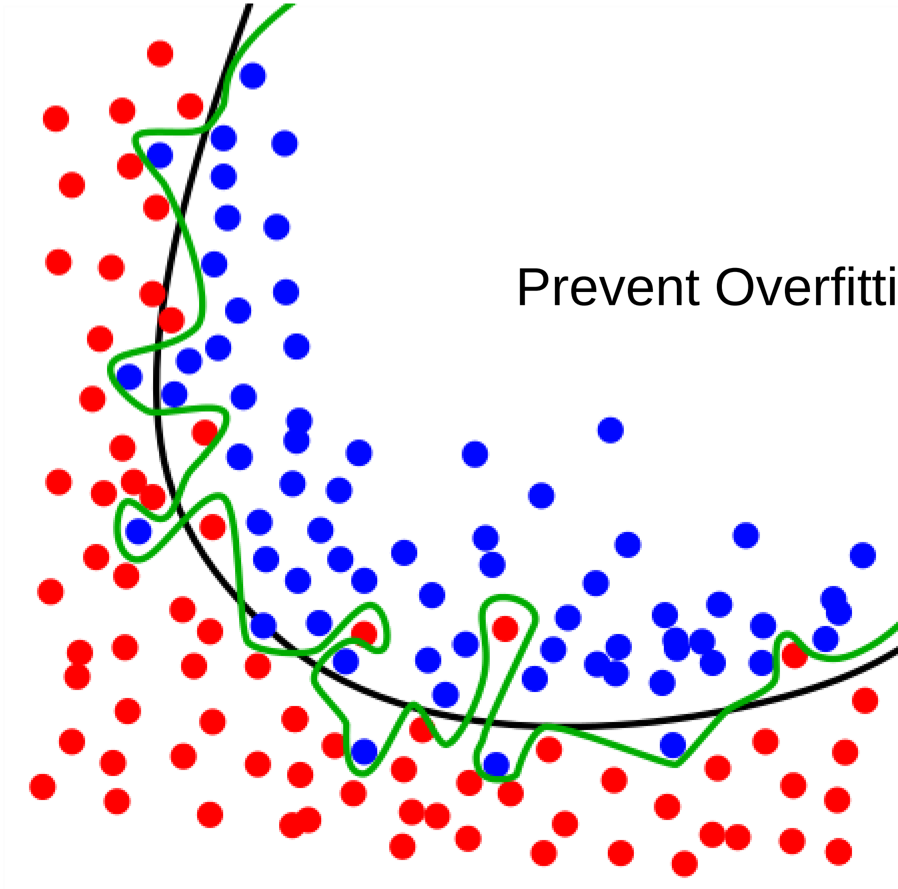
Bias-Variance Trade-off





- Models with too few parameters are inaccurate because of a large bias (not enough flexibility)
- Models with too many parameters are inaccurate because of a large variance (too much sensitivity to the sample)

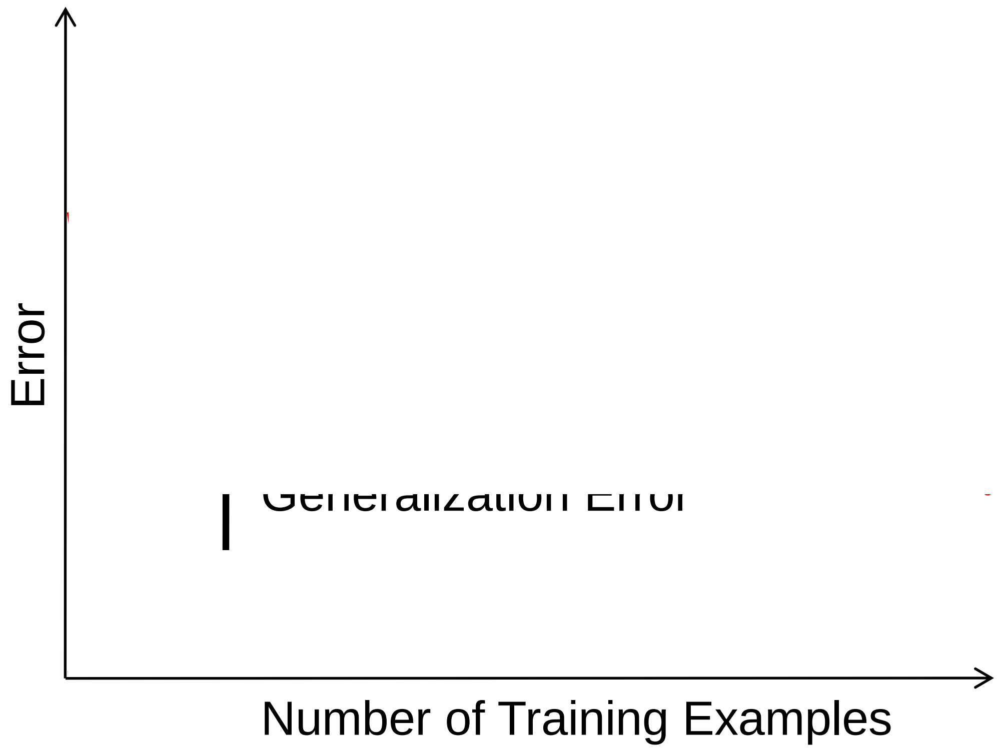
Regularization



Prevent Overfitting: bias green toward black

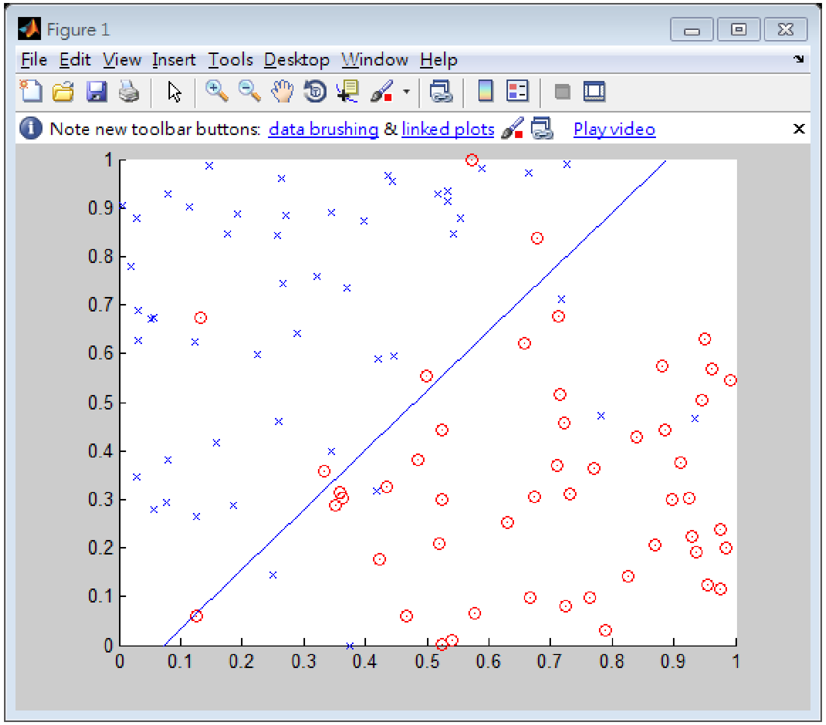
Effect of Training Size

Fixed prediction model



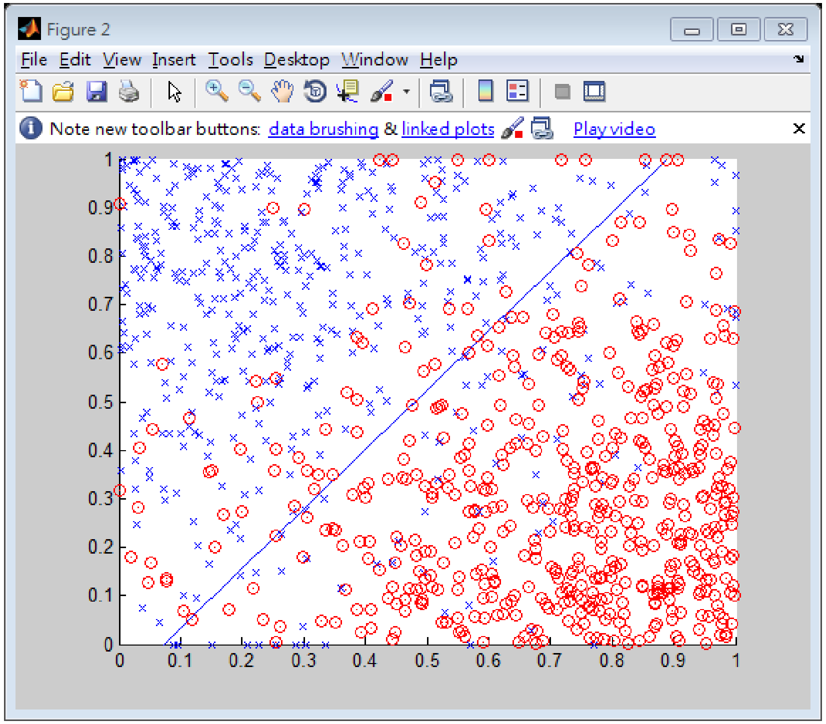
Comparison of errors

Using **logistic regression**



Training

Error rate:
0.11



Testing

Error rate:
0.145

Next Week

- More on deep learning
- Start research papers on Thursday