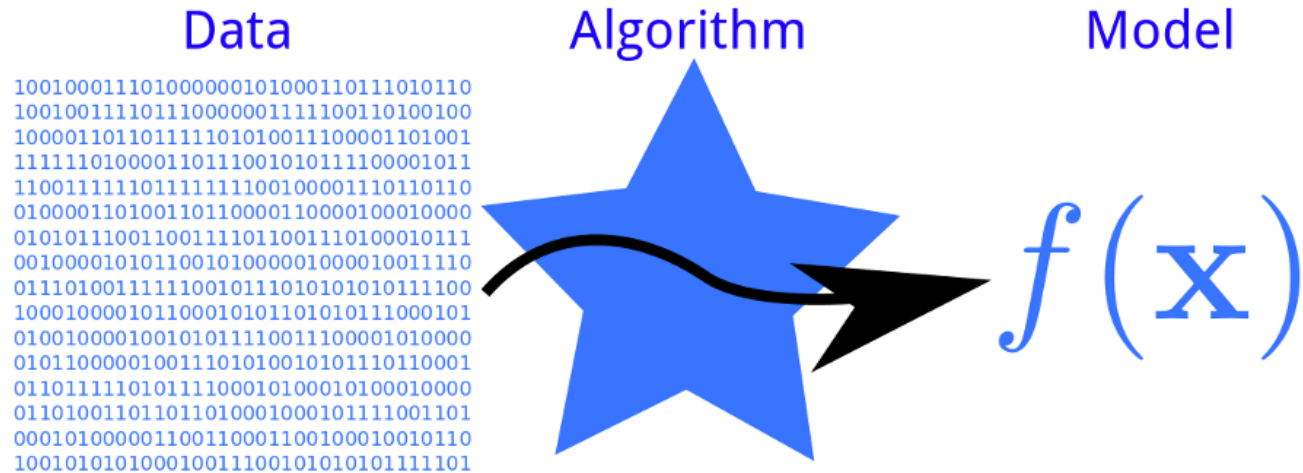


Introduction to Machine Learning



Lecture 1: Introduction

Iasonas Kokkinos

iasonas.kokkinos@gmail.com

University College London

Machine Learning

Principles, methods, and algorithms for learning and prediction based on past evidence

Goal: Machines that perform a task based on experience, instead of explicitly coded instructions

Why?

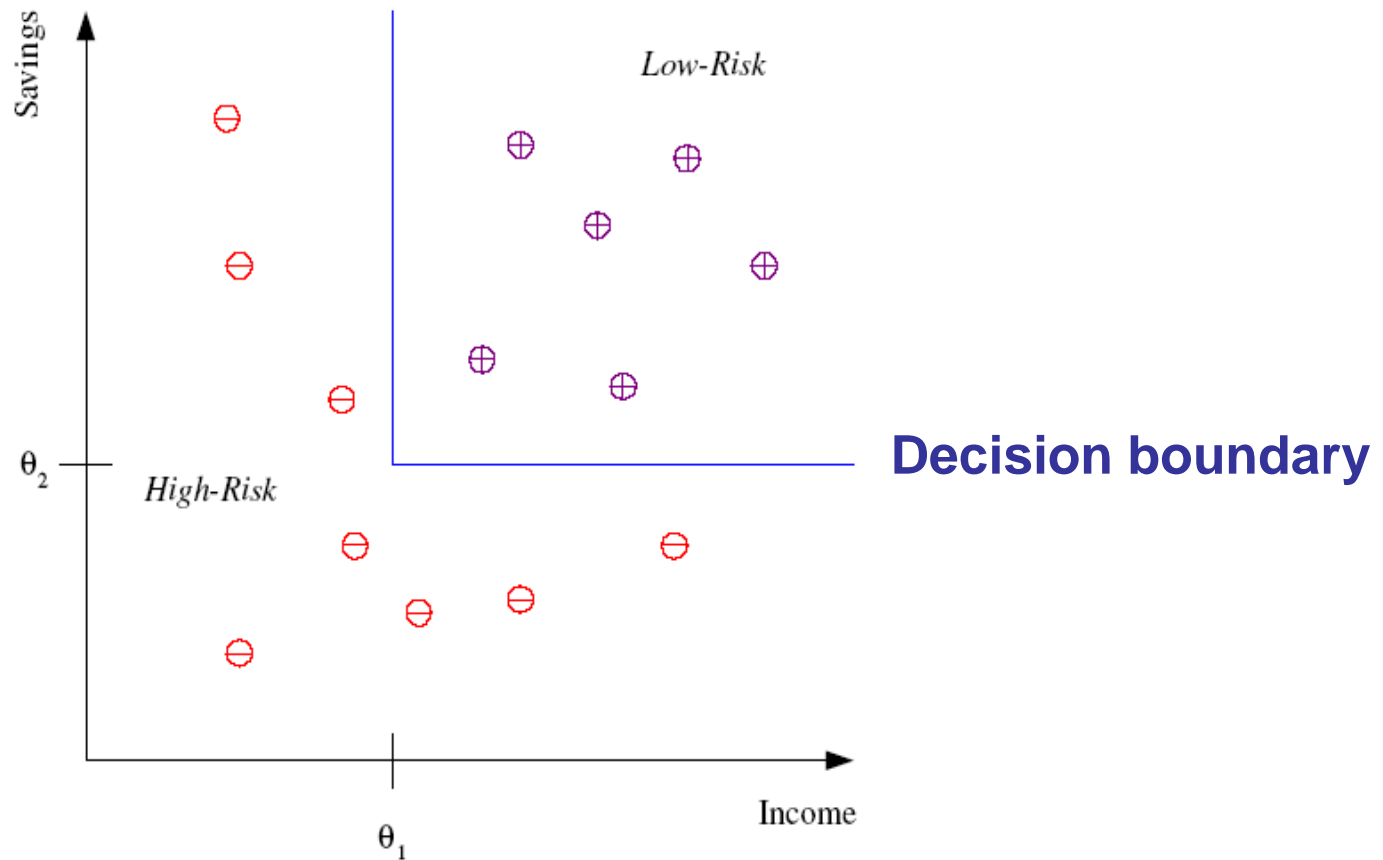
- Crucial component of every intelligent/autonomous system
- Important for a system's adaptability
- Important for a system's generalization capabilities
- Attempt to understand human learning

Machine Learning variants

- Supervised
 - Classification
 - Regression
- Unsupervised
 - Clustering
 - Dimensionality Reduction
- Weakly supervised/semi-supervised
 - Some data supervised, some unsupervised
- Reinforcement learning
 - Supervision: sparse reward for a sequence of decisions

Classification

- Based on our experience, should we give a loan to this customer?
 - Binary decision: yes/no



Classification examples

- Digit Recognition



- Spam Detection



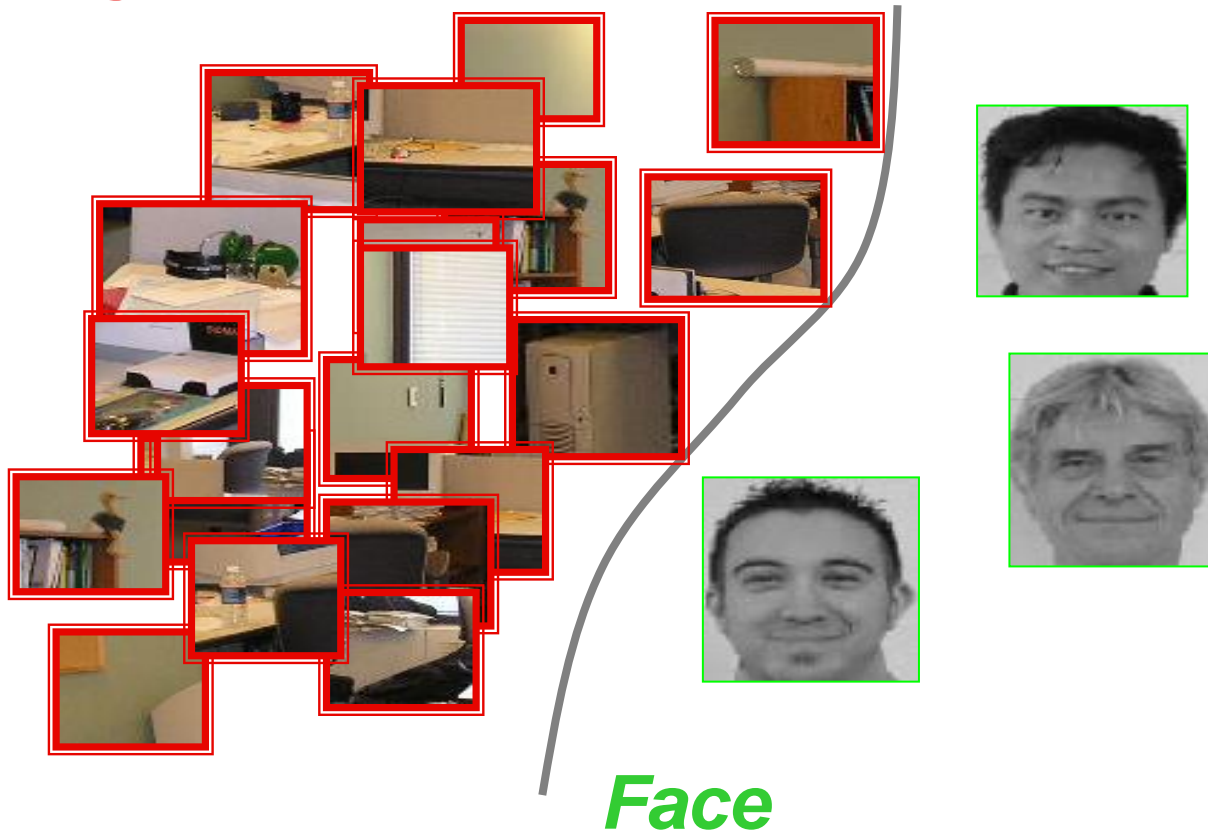
- Face detection



`Faceness function': classifier

Background

Decision boundary



Test time: deploy the learned function

- Scan window over image
 - Multiple scales
 - Multiple orientations
- Classify window as either:
 - Face
 - Non-face

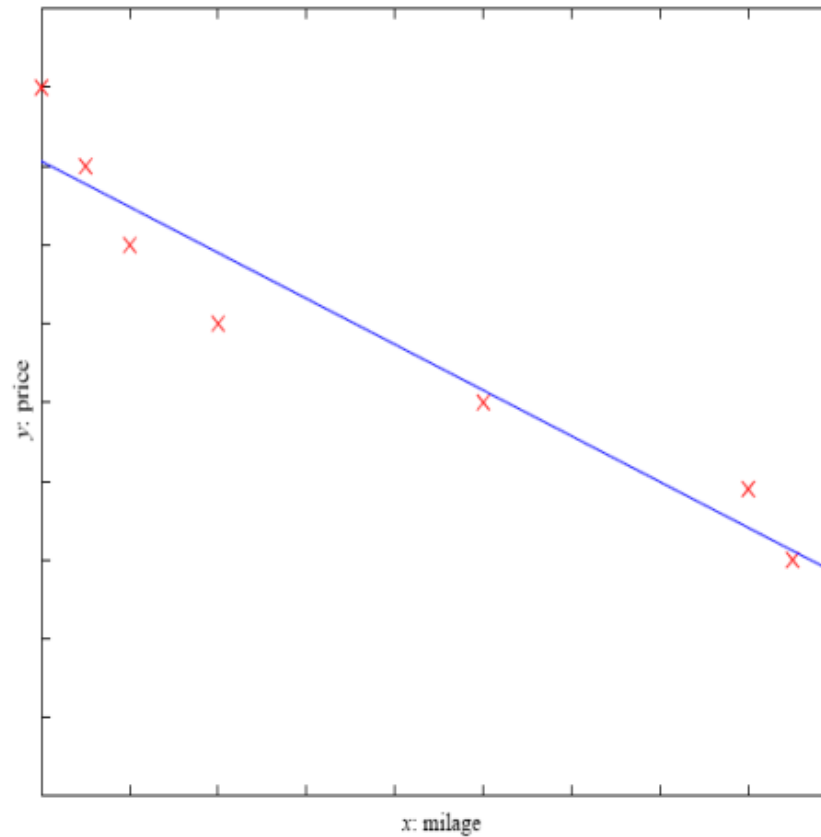


Machine Learning variants

- Supervised
 - Classification
 - Regression
- Unsupervised
 - Clustering
 - Dimensionality Reduction
- Weakly supervised
 - Some data supervised, some unsupervised
- Reinforcement learning
 - Supervision: reward for a sequence of decisions

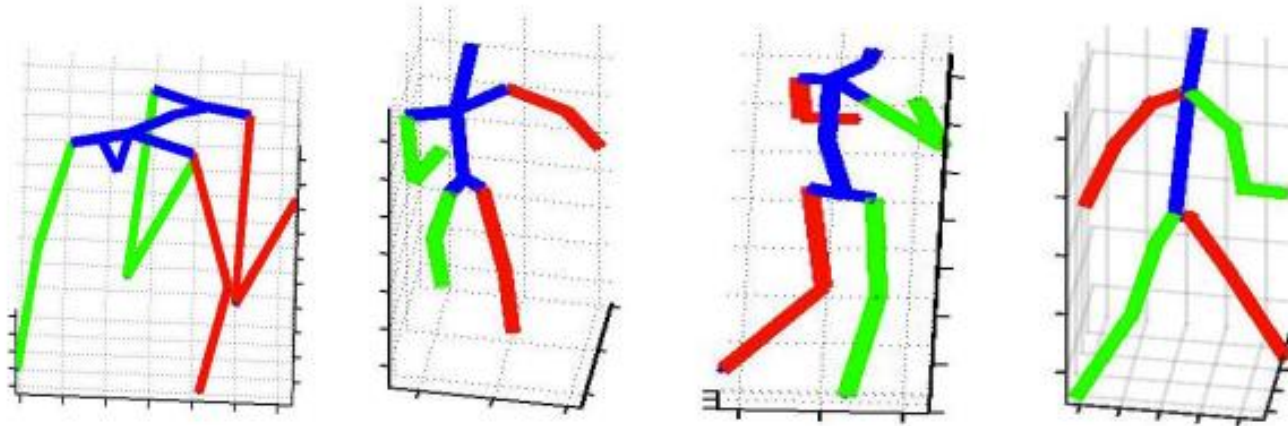
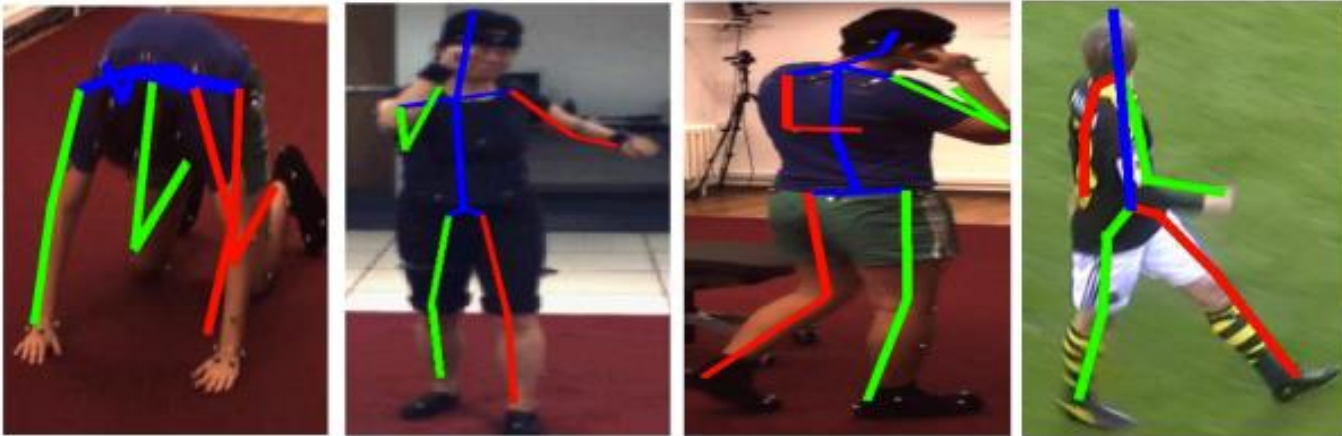
Regression

- Output: Continuous
 - E.g. price of a car based on years, mileage, condition,...



Computer vision example

- Human estimation: from image to vector-valued pose estimate

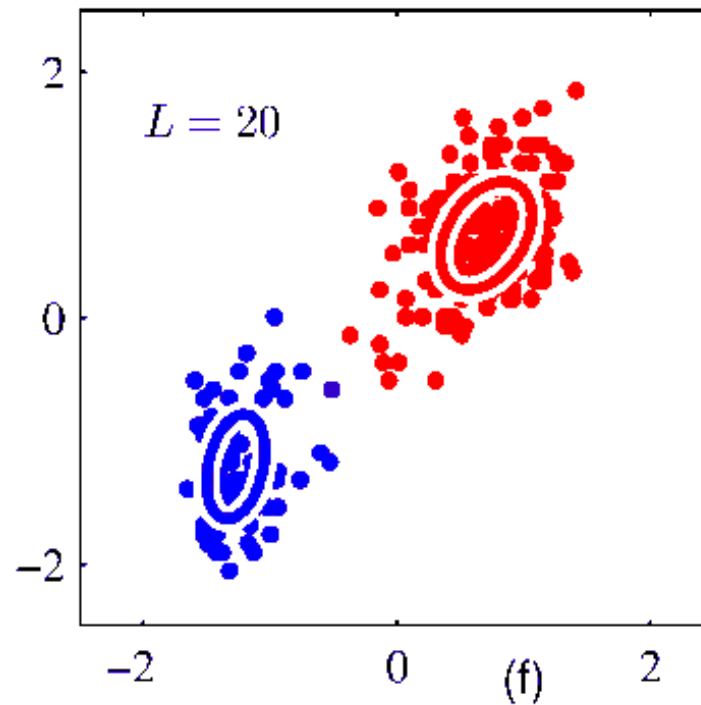


Machine Learning variants

- Supervised
 - Classification
 - Regression
- Unsupervised
 - Clustering
 - Dimensionality Reduction
- Weakly supervised
 - Some data supervised, some unsupervised
- Reinforcement learning
 - Supervision: reward for a sequence of decisions

Clustering

- Break a set of data into coherent groups
 - Labels are 'invented'



Clustering examples

- Spotify recommendations

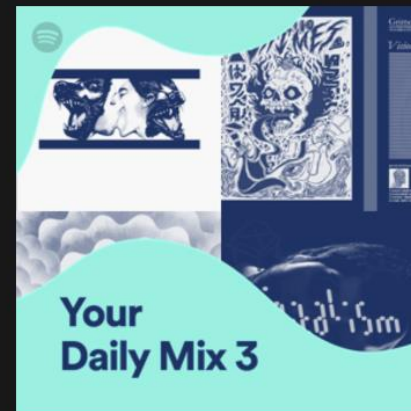
Play the music you love, without the effort. Packed with your favorites and new discoveries.



Daily Mix 1
Agar Agar, Juniore,
L'Impératrice and more



Daily Mix 2
Yo-Yo Ma, Glenn Gould,
Murray Perahia and more



Daily Mix 3
Holy Ghost!, Grimes,
Metronomy and more



Daily Mix 4
The Jesus and Mary Chain,
Belle & Sebastian, The Shins

Clustering examples

- Image segmentation

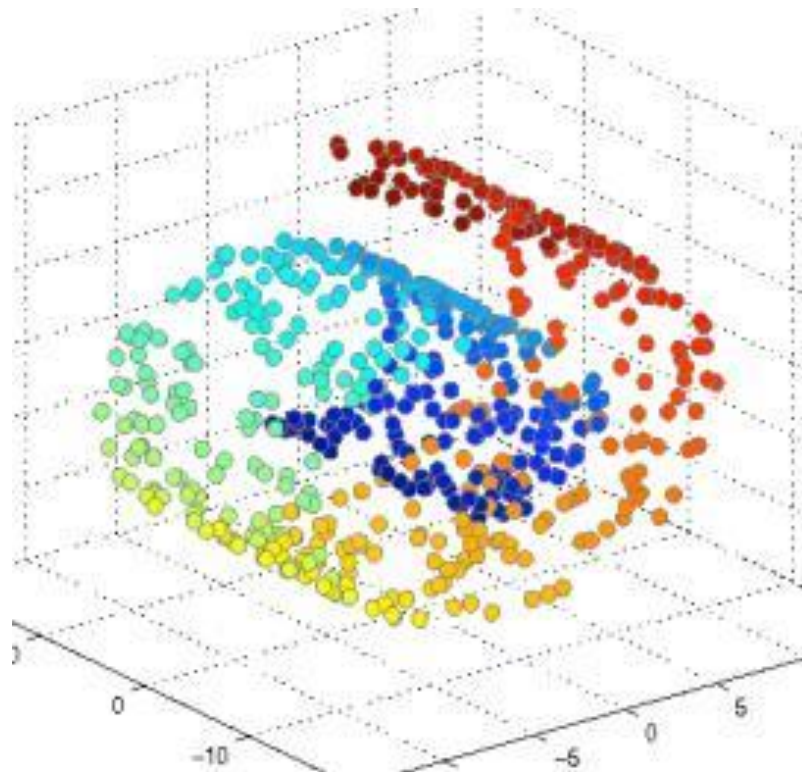


Machine Learning variants

- Supervised
 - Classification
 - Regression
- Unsupervised
 - Clustering
 - Dimensionality Reduction
- Weakly supervised
 - Some data supervised, some unsupervised
- Reinforcement learning
 - Supervision: reward for a sequence of decisions

Dimensionality reduction & manifold learning

- Find a low-dimensional representation of high-dimensional data
 - Continuous outputs are 'invented'



Example of nonlinear manifold: faces

Average of two faces is not a face



\mathbf{x}_1



$\frac{1}{2}(\mathbf{x}_1 + \mathbf{x}_2)$



\mathbf{x}_2

Moving along the learned face manifold



Trajectory along the “male” dimension



Trajectory along the “young” dimension

Lample et. al. Fader Networks, NIPS 2017

Machine Learning variants

- Supervised
 - Classification
 - Regression
- Unsupervised
 - Clustering
 - Dimensionality Reduction
- Weakly supervised/semi supervised
 - Partially supervised
- Reinforcement learning
 - Supervision: reward for a sequence of decisions

Weakly supervised learning: only part of the supervision signal

training images



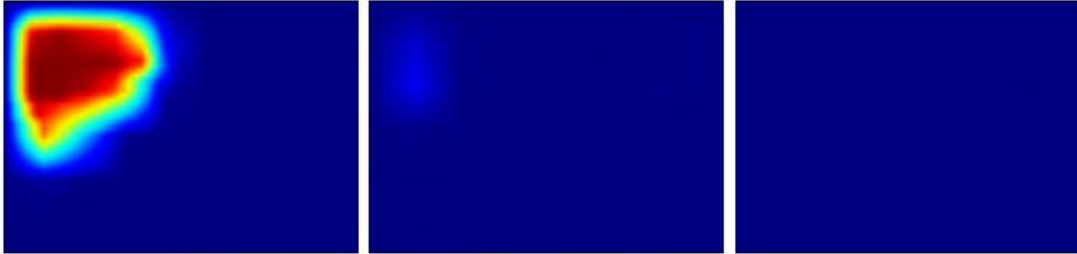
**Supervision signal:
“motorcycle”**

Weakly supervised learning: only part of the supervision signal

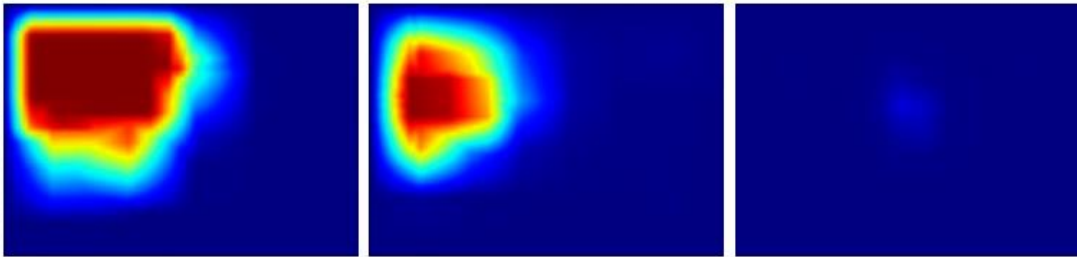
training images



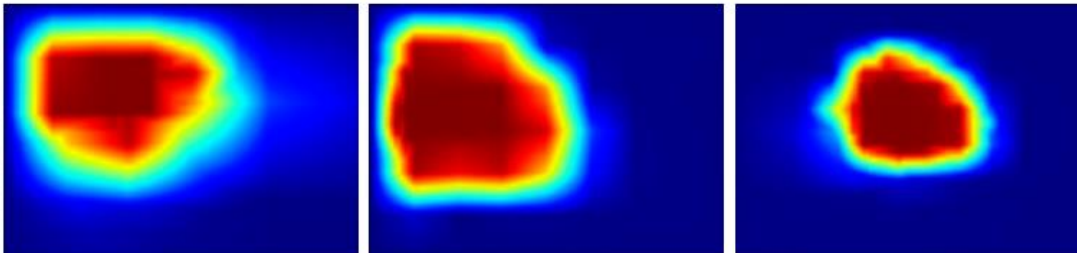
train iter. 210



train iter. 510



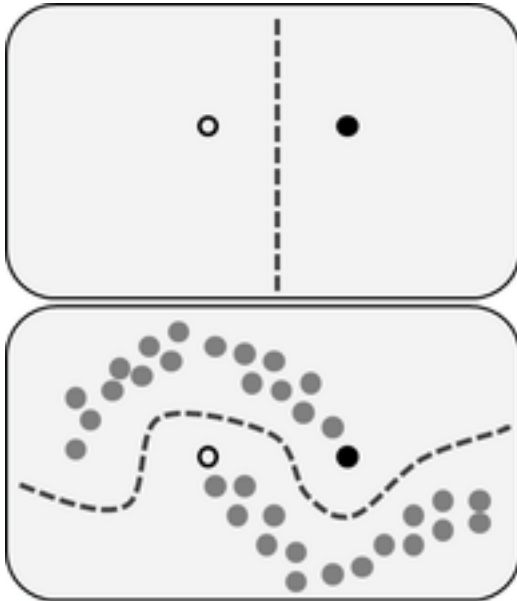
train iter. 4200



Supervision signal:
“motorcycle”

Inferred localization
information

Semi-supervised learning: only part of the data labelled



Labelled data

Labelled + unlabelled data

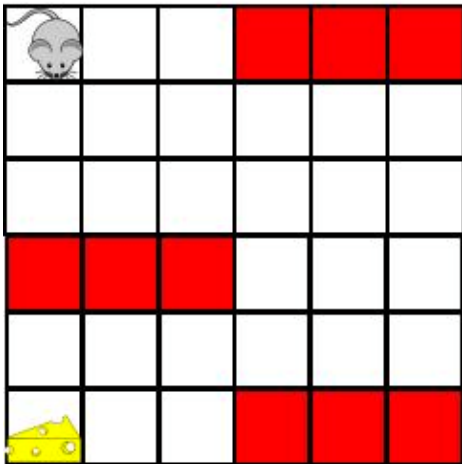
Machine Learning variants

- Supervised
 - Classification
 - Regression
- Unsupervised
 - Clustering
 - Dimensionality Reduction
- Weakly supervised/semi supervised learning
 - Some data supervised, some unsupervised
- Reinforcement learning
 - Supervision: reward for a sequence of decisions

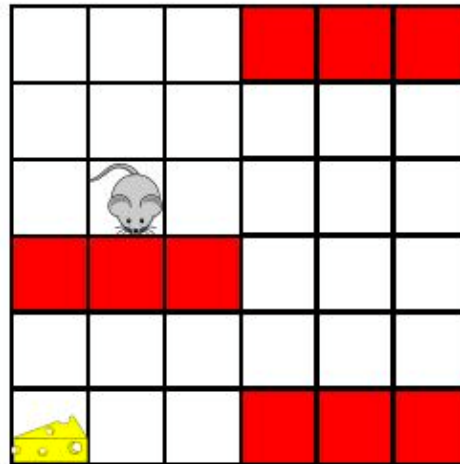
Reinforcement learning

- Agent interacts with environment repeatedly
 - Take actions, based on state
 - (occasionally) receive rewards
 - Update state
 - Repeat
- Goal: maximize cumulative reward

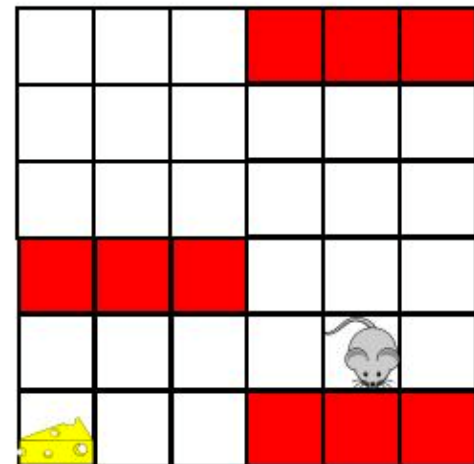
State 1



State 2



State 3



Reinforcement learning examples

- Beat human champions in games

Backgammon, 90's



GO, 2015



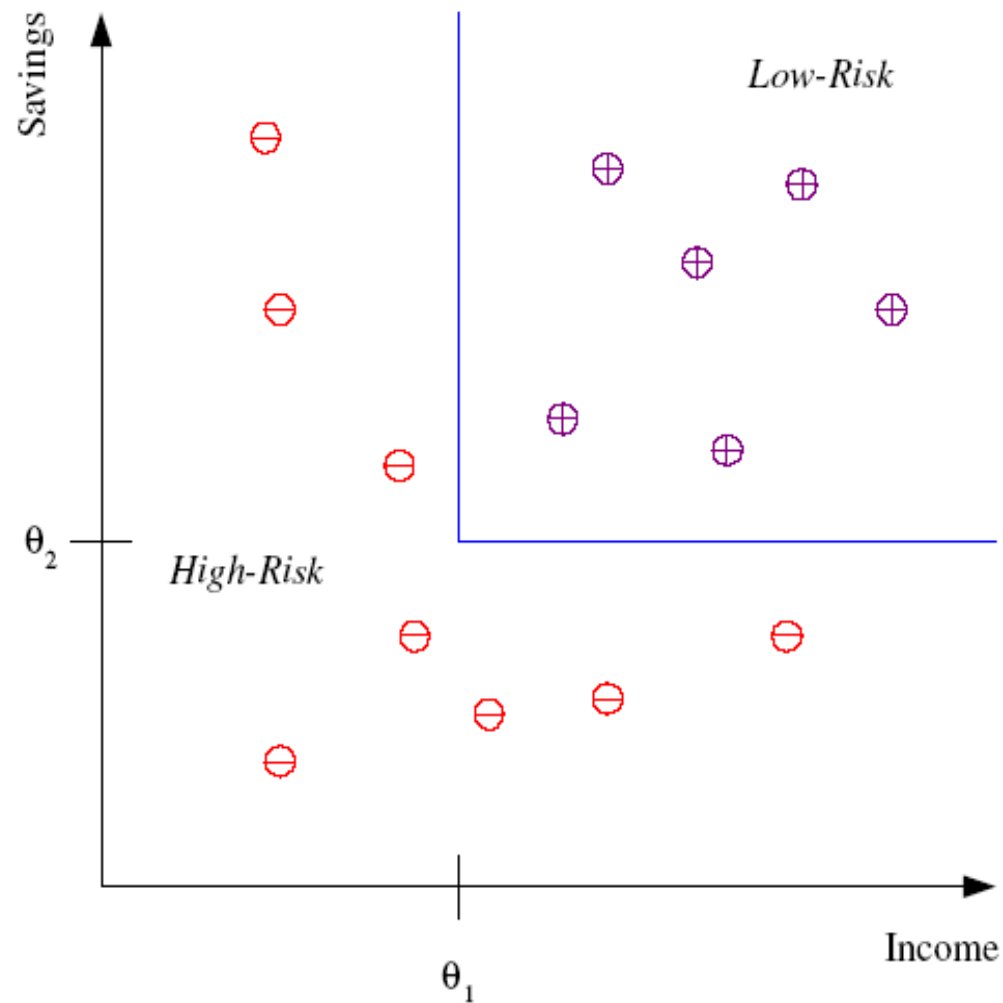
- Robotics



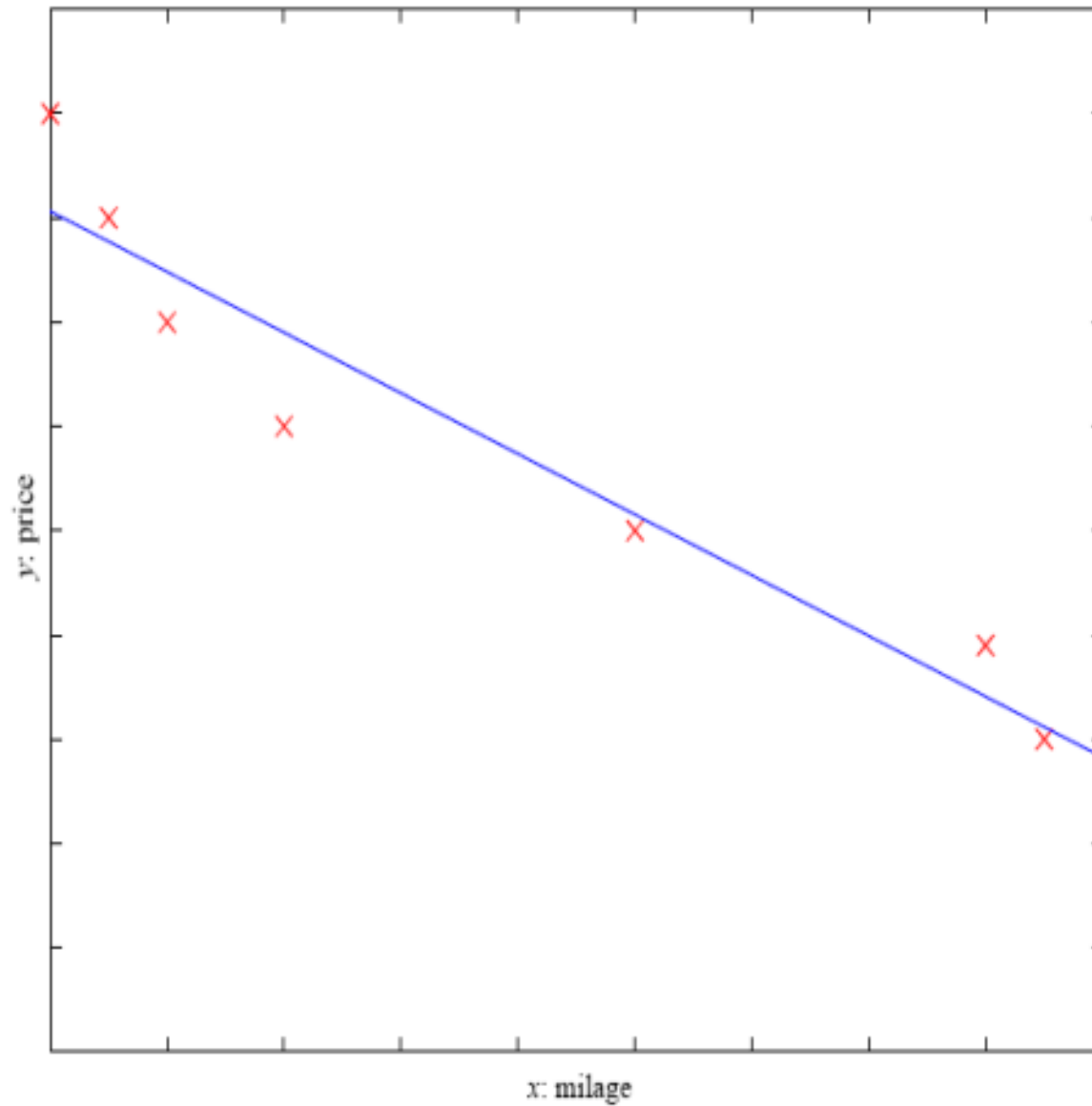
Focus of first part: supervised learning

- Supervised
 - Classification
 - Regression
- Unsupervised
 - Clustering
 - Dimensionality Reduction, Manifold Learning
- Weakly supervised
 - Some data supervised, some unsupervised
- Reinforcement learning
 - Supervision: reward for a sequence of decisions

Classification: yes/no decision



Regression: continuous output



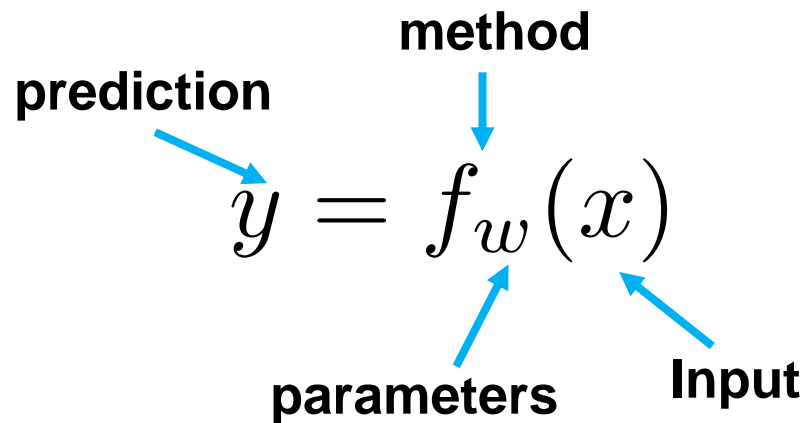
What we want to learn: a function

- Input-output mapping

$$y = f_w(x)$$

What we want to learn: a function

- Input-output mapping



What we want to learn: a function

A diagram showing the equation $y = f_w(x)$ with four blue arrows pointing to its components: 'prediction' points to y , 'method' points to f , 'parameters' points to w , and 'Input' points to x .

prediction

method

$y = f_w(x)$

parameters

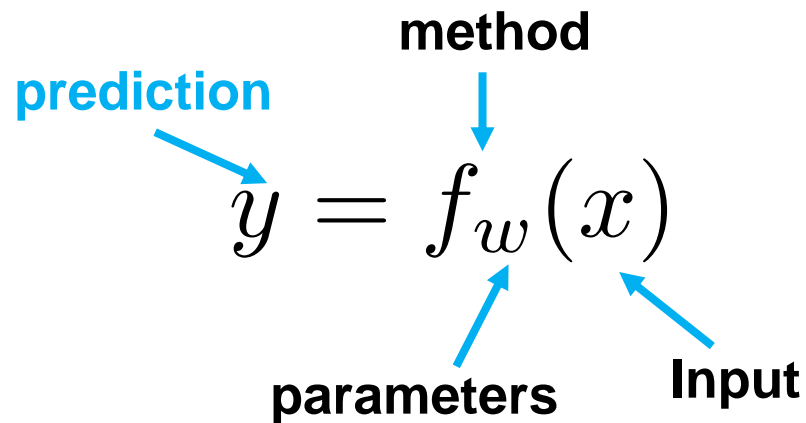
Input

Calculus $x \in \mathbb{R}$

Vector calculus $\mathbf{x} \in \mathbb{R}^D$

Machine learning: can work also for discrete inputs, strings, trees, graphs,...

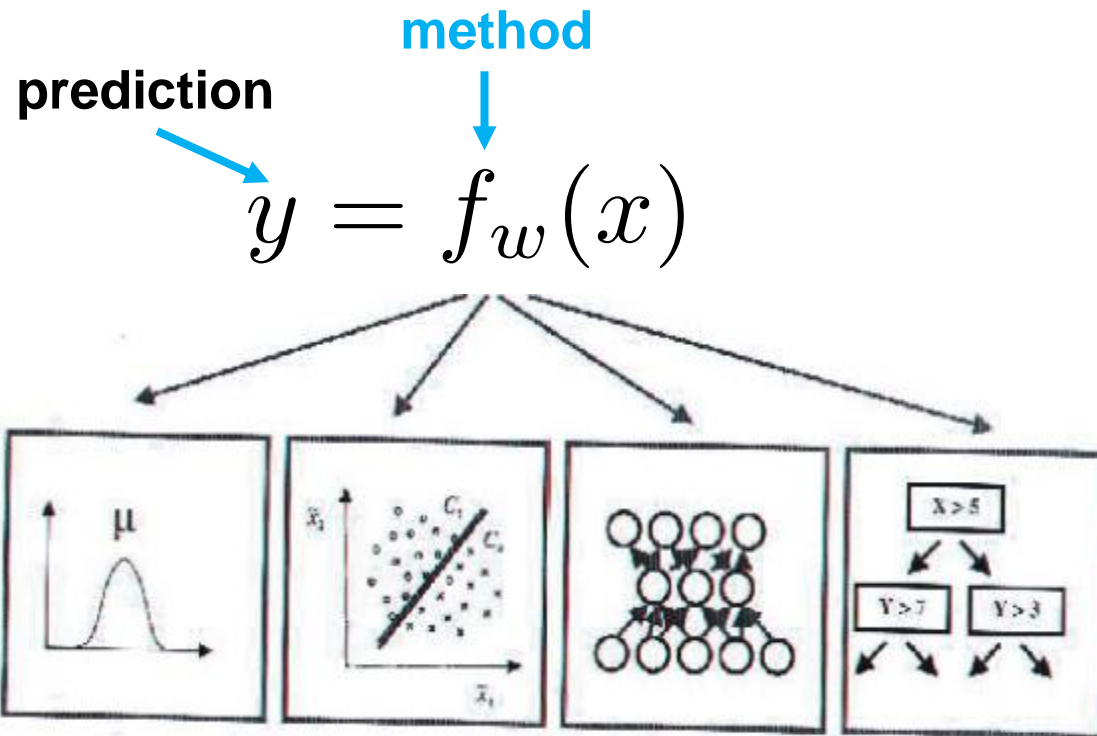
What we want to learn: a function



Classification: $y \in \{0, 1\}$

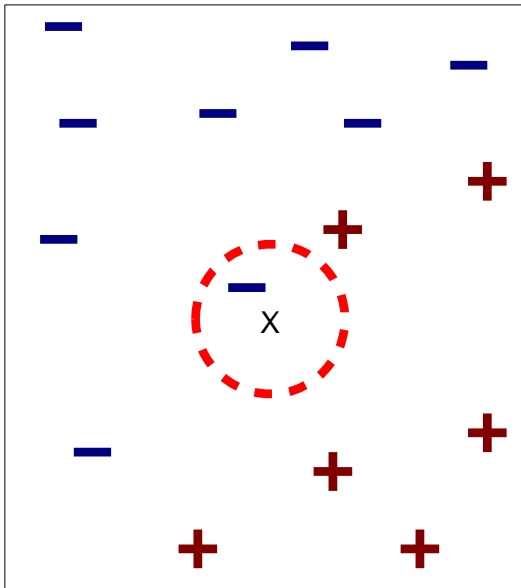
Regression: $y \in \mathbb{R}$

What we want to learn: a function

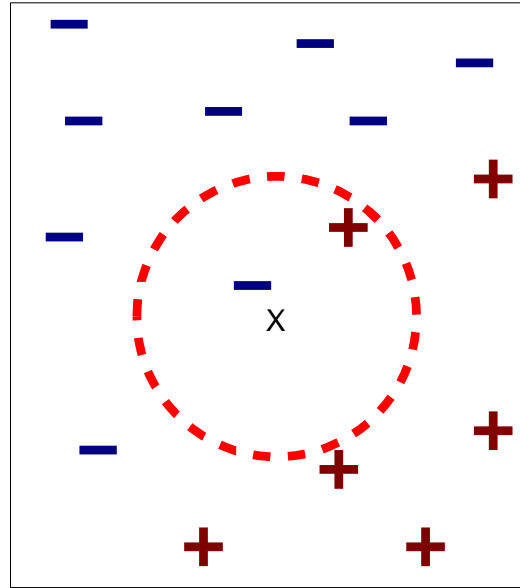


Linear classifiers, neural networks, decision trees, ensemble models, probabilistic classifiers, ...

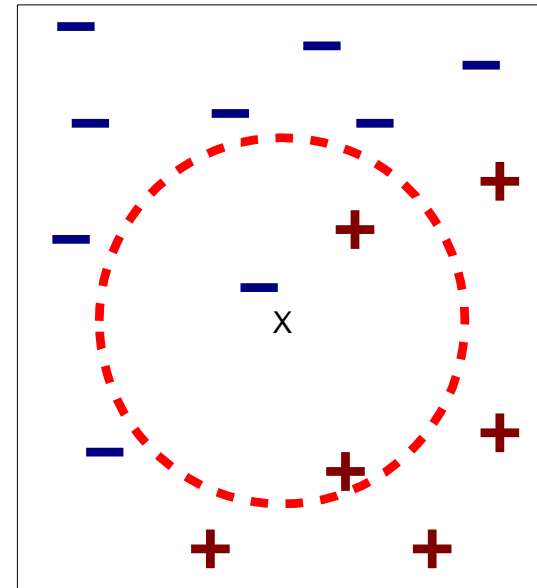
Example of method: K-nearest neighbor classifier



(a) 1-nearest neighbor



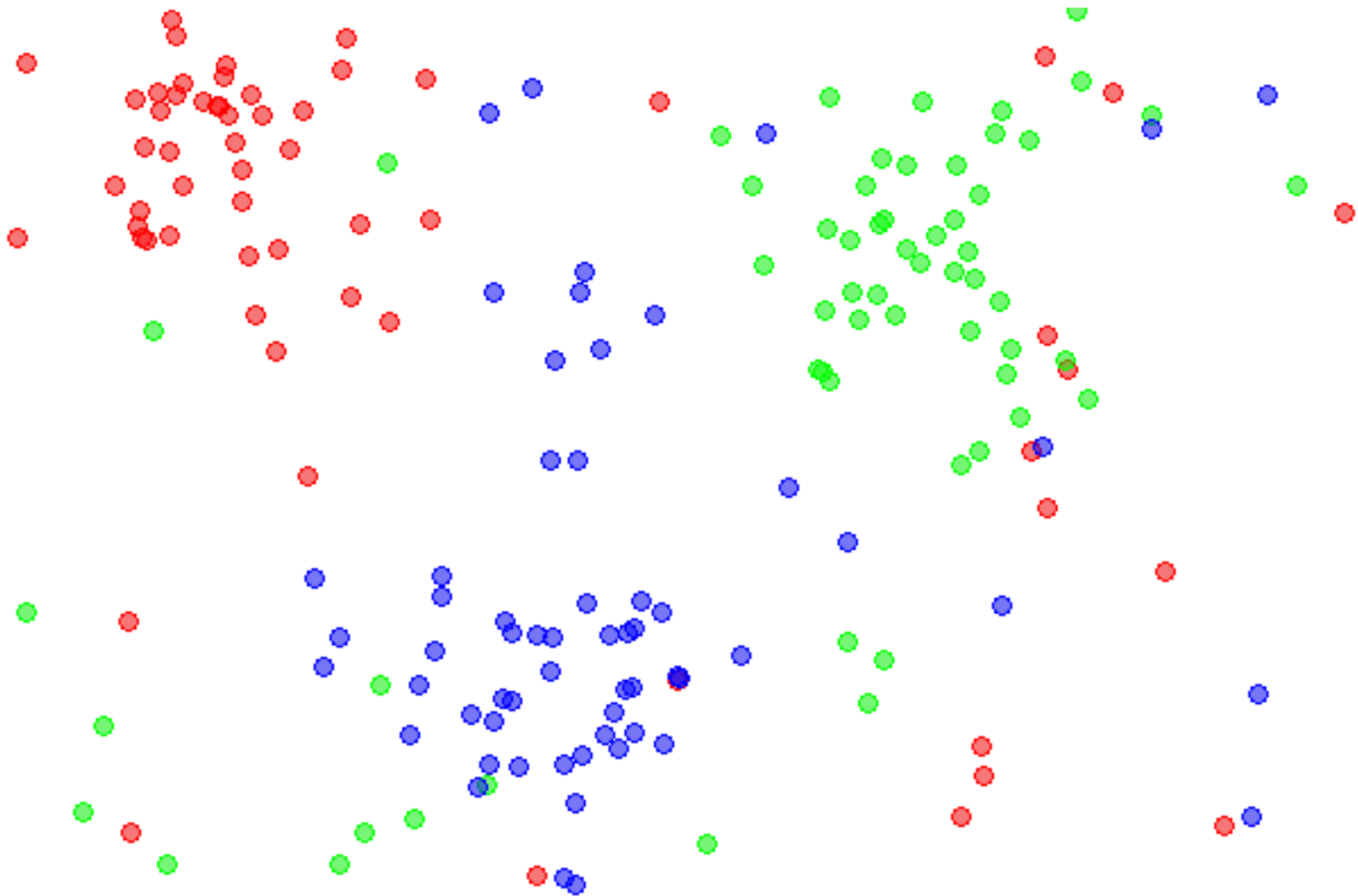
(b) 2-nearest neighbor



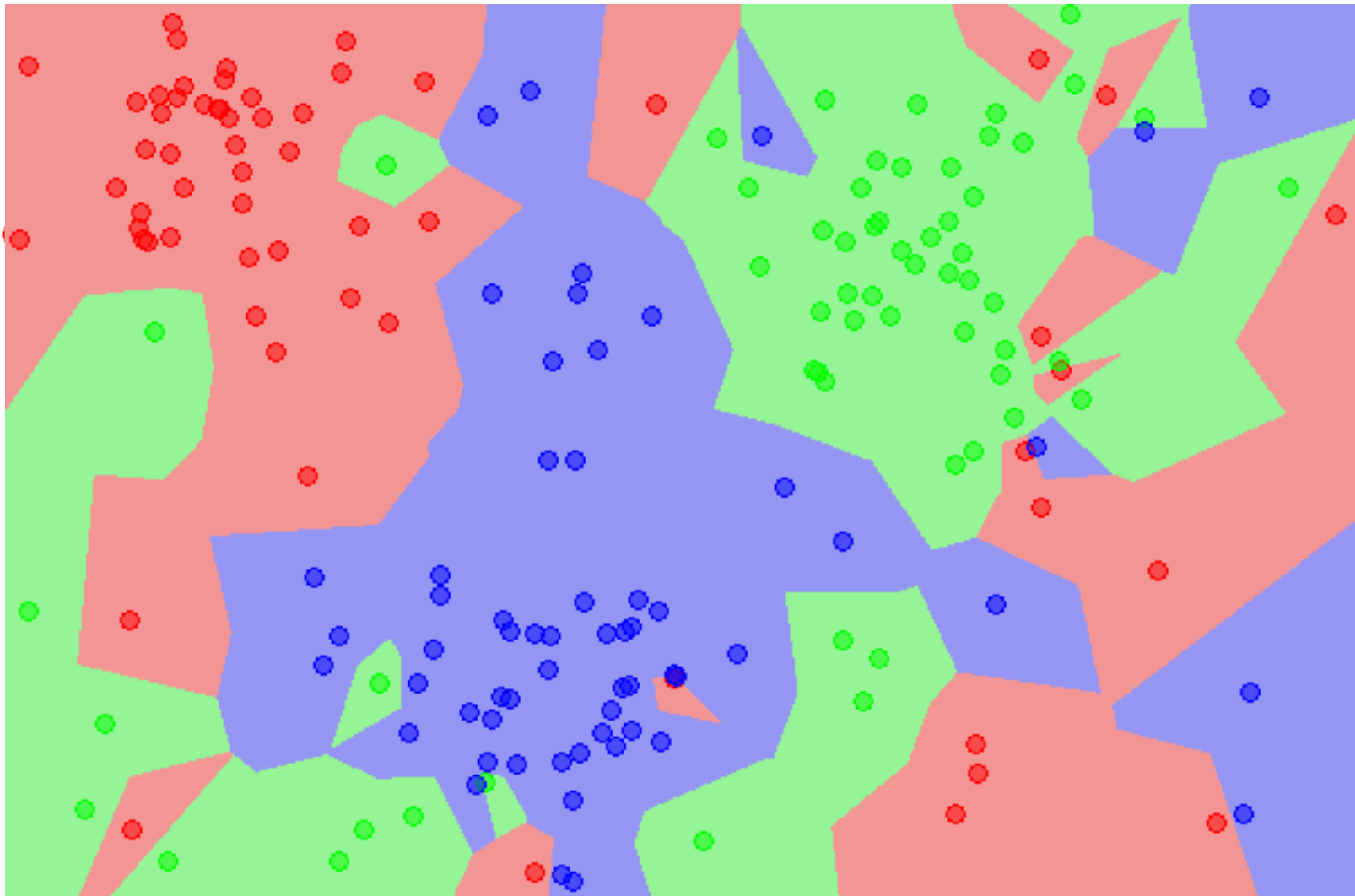
(c) 3-nearest neighbor

- Compute distance to other training records
- Identify K nearest neighbors
- Take majority vote

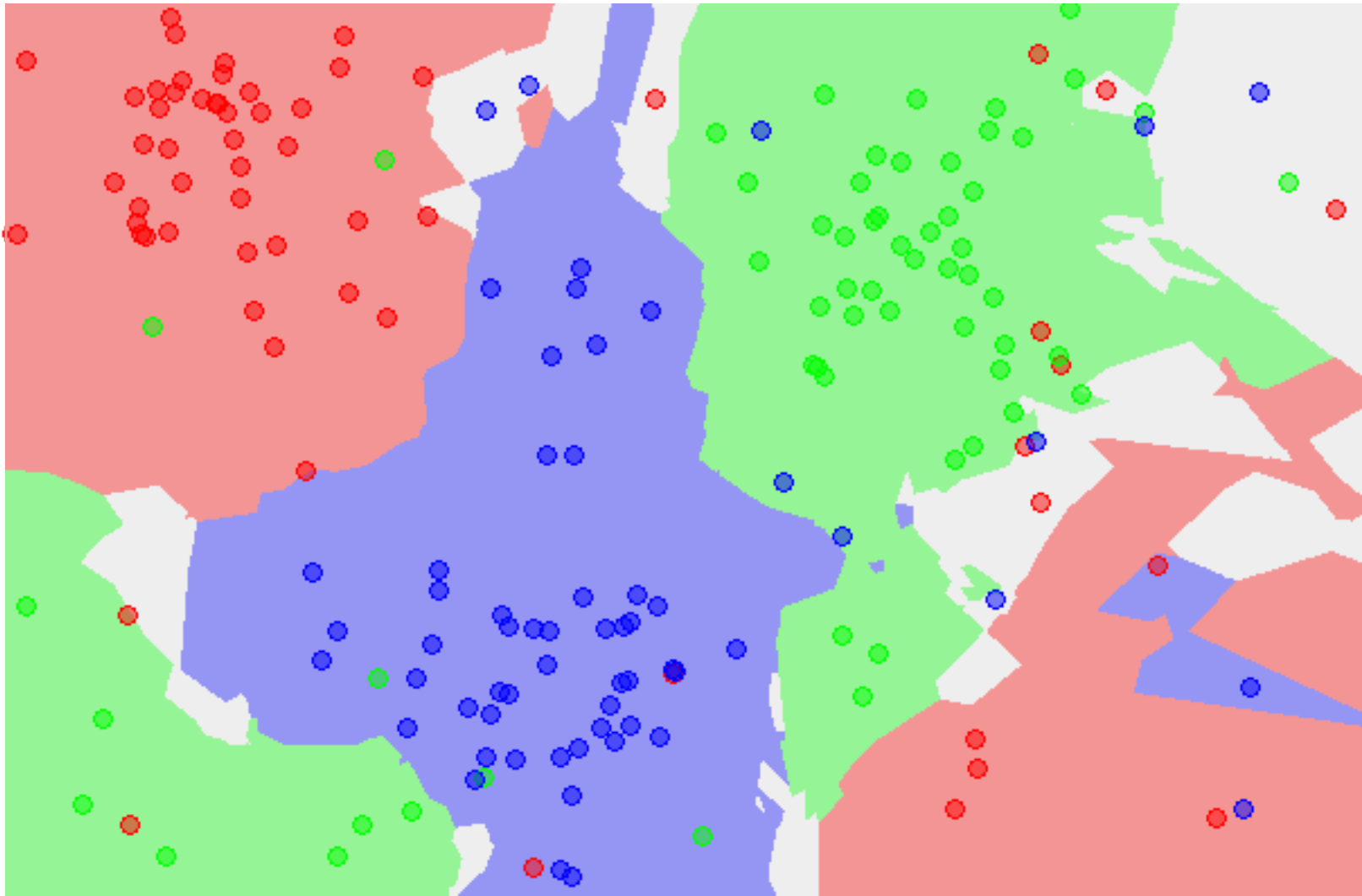
Training data for NN classifier (in \mathbb{R}^2)



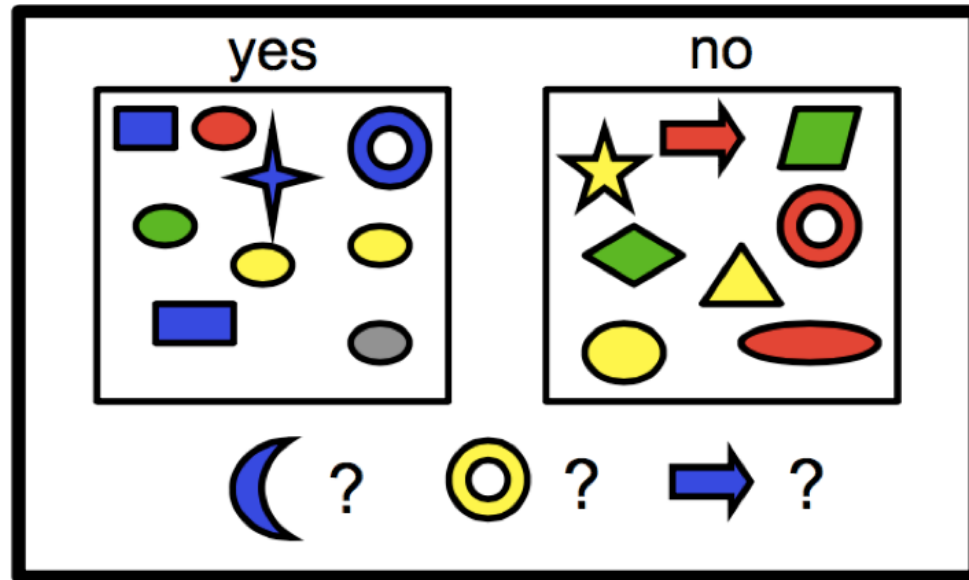
1-nn classifier prediction (in R^2)



3-nn classifier prediction



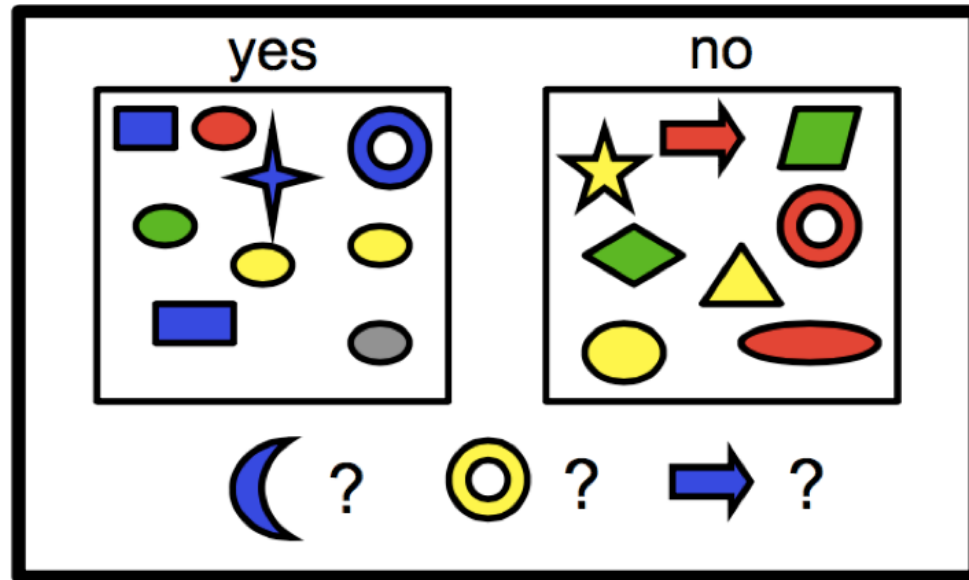
Method example: decision tree



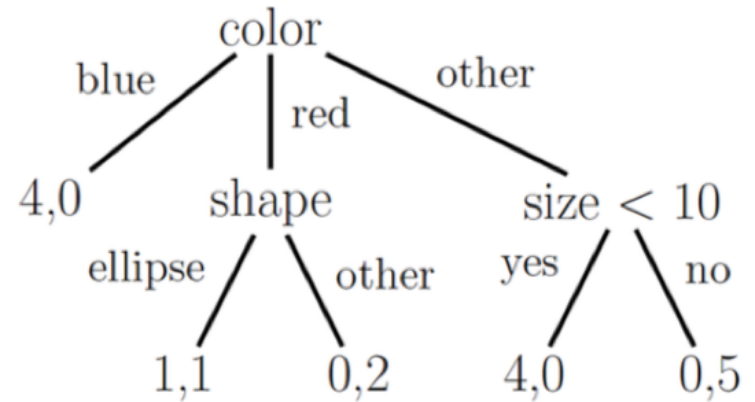
Features: color, shape, size

Machine learning: can work also for discrete inputs, strings, trees, graphs,...

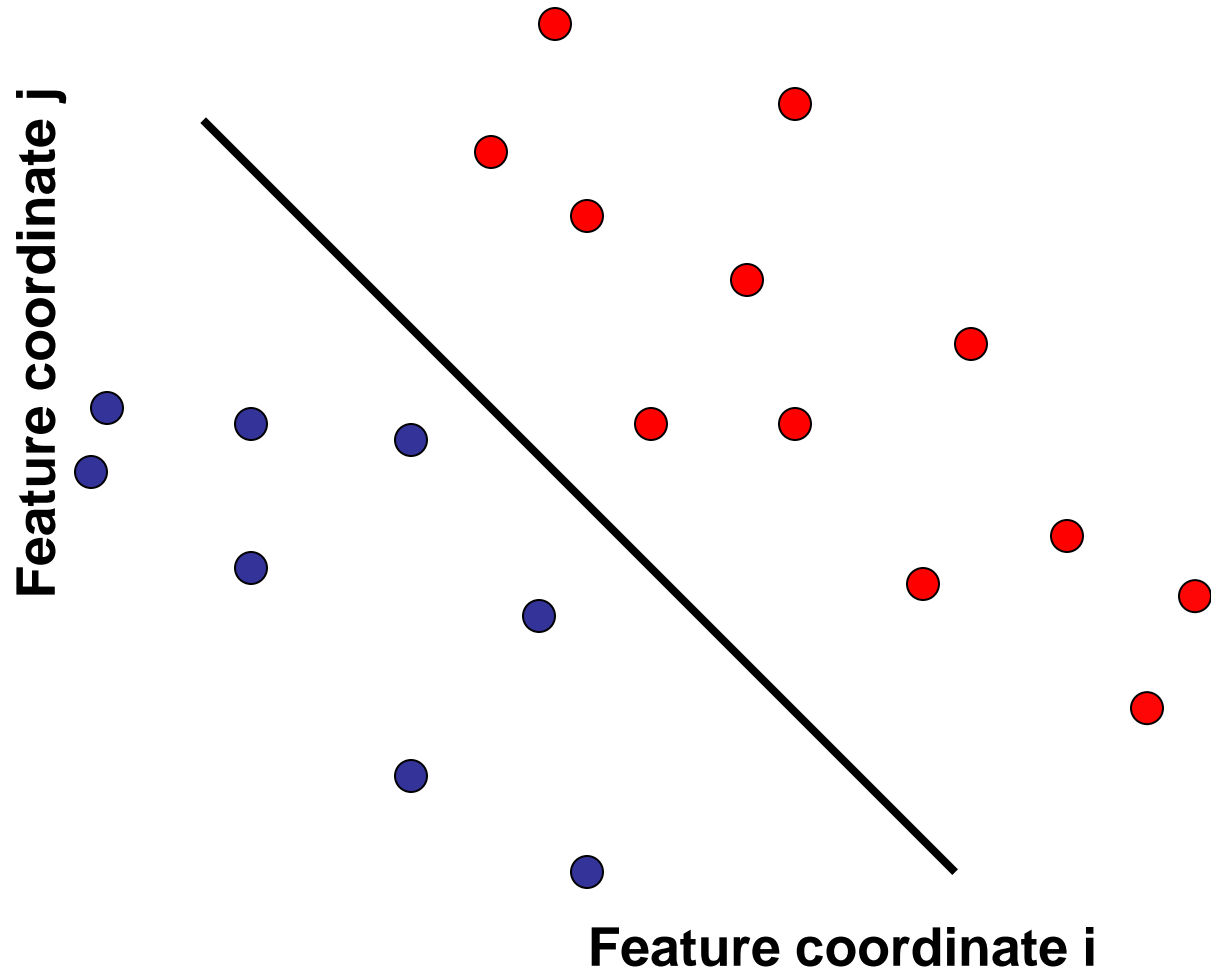
Method example: decision tree



Features: color, shape, size

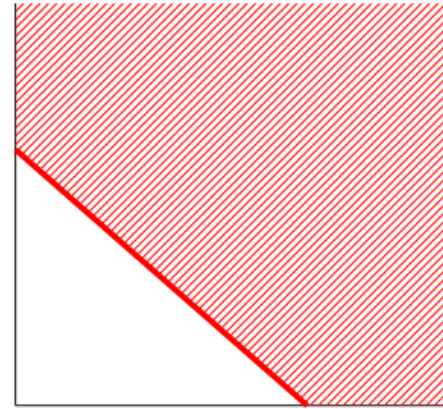
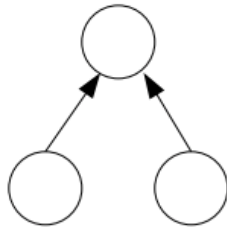


Method example: linear classifier



Method example: neural network

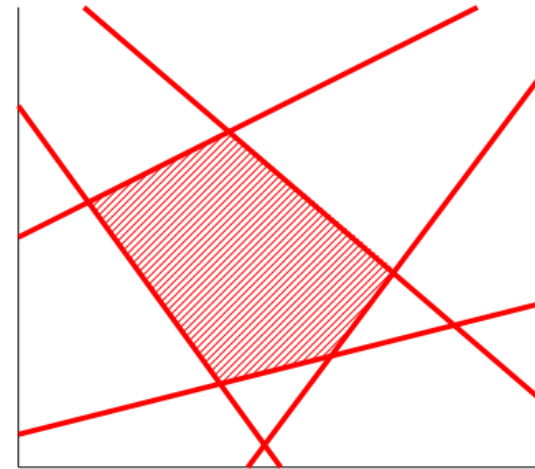
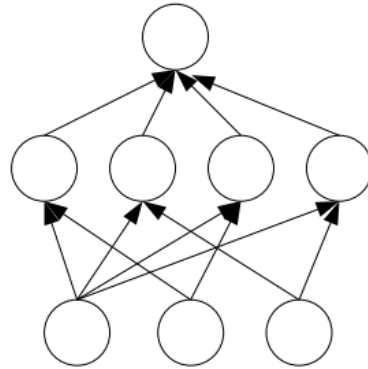
1 layer of
trainable
weights



separating hyperplane

Method example: neural network

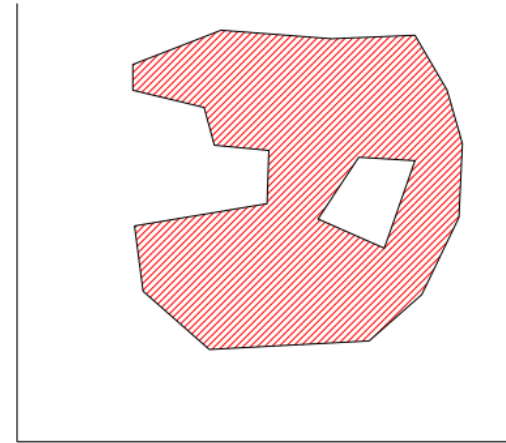
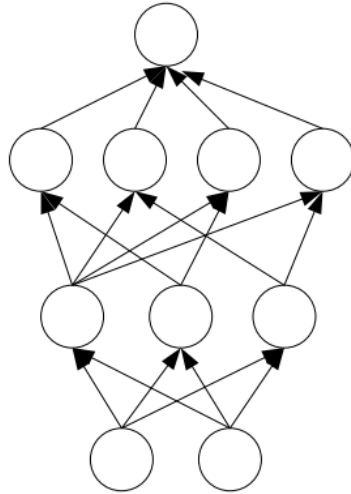
2 layers of
trainable
weights



convex polygon region

Method example: neural network

3 layers of
trainable
weights



composition of polygons:
convex regions

We have two centuries of material to cover!

https://en.wikipedia.org/wiki/Least_squares

The first clear and concise exposition of the method of least squares was published by Legendre in **1805**.

The technique is described as an **algebraic procedure** for **fitting linear equations to data** and Legendre demonstrates the new method by analyzing the same data as Laplace for the shape of the earth.

The value of Legendre's method of least squares was immediately recognized by leading astronomers and geodesists of the time

What we want to learn: a function

- Input-output mapping

The diagram shows the equation $y = f_w(x) = f(x; w)$ with four blue arrows pointing to its components: 'prediction' points to y , 'method' points to f , 'parameters' points to w , and 'Input' points to x .

$$\text{prediction} \rightarrow y = f_w(x) = f(x; w)$$

method
parameters Input

$$w \in \mathbb{R}$$

$$\mathbf{w} \in \mathbb{R}^K$$

Assumption: linear function

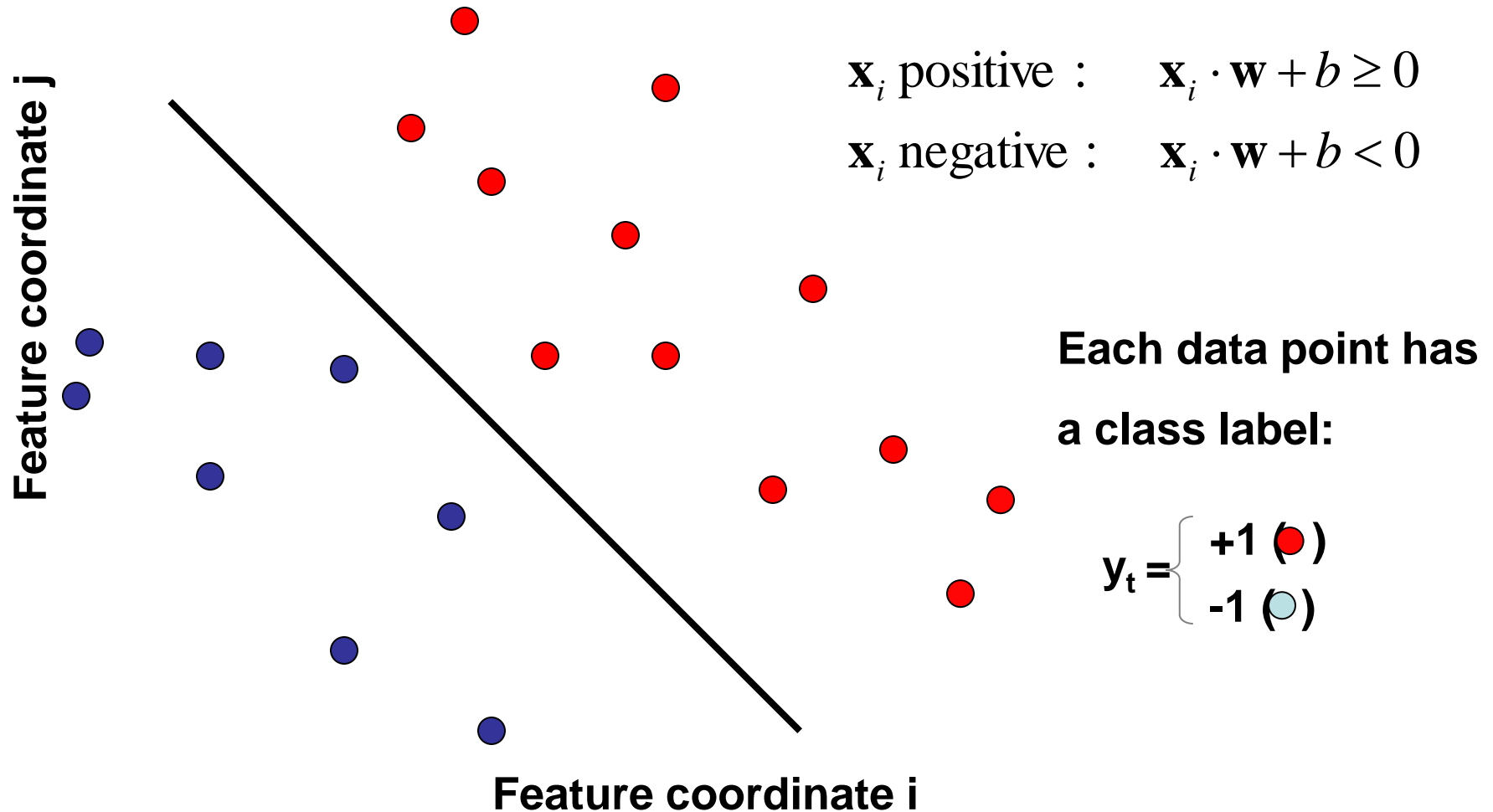
$$y = f_{\mathbf{w}}(\mathbf{x}) = f(\mathbf{x}, \mathbf{w}) = \mathbf{w}^T \mathbf{x}$$

Inner product:

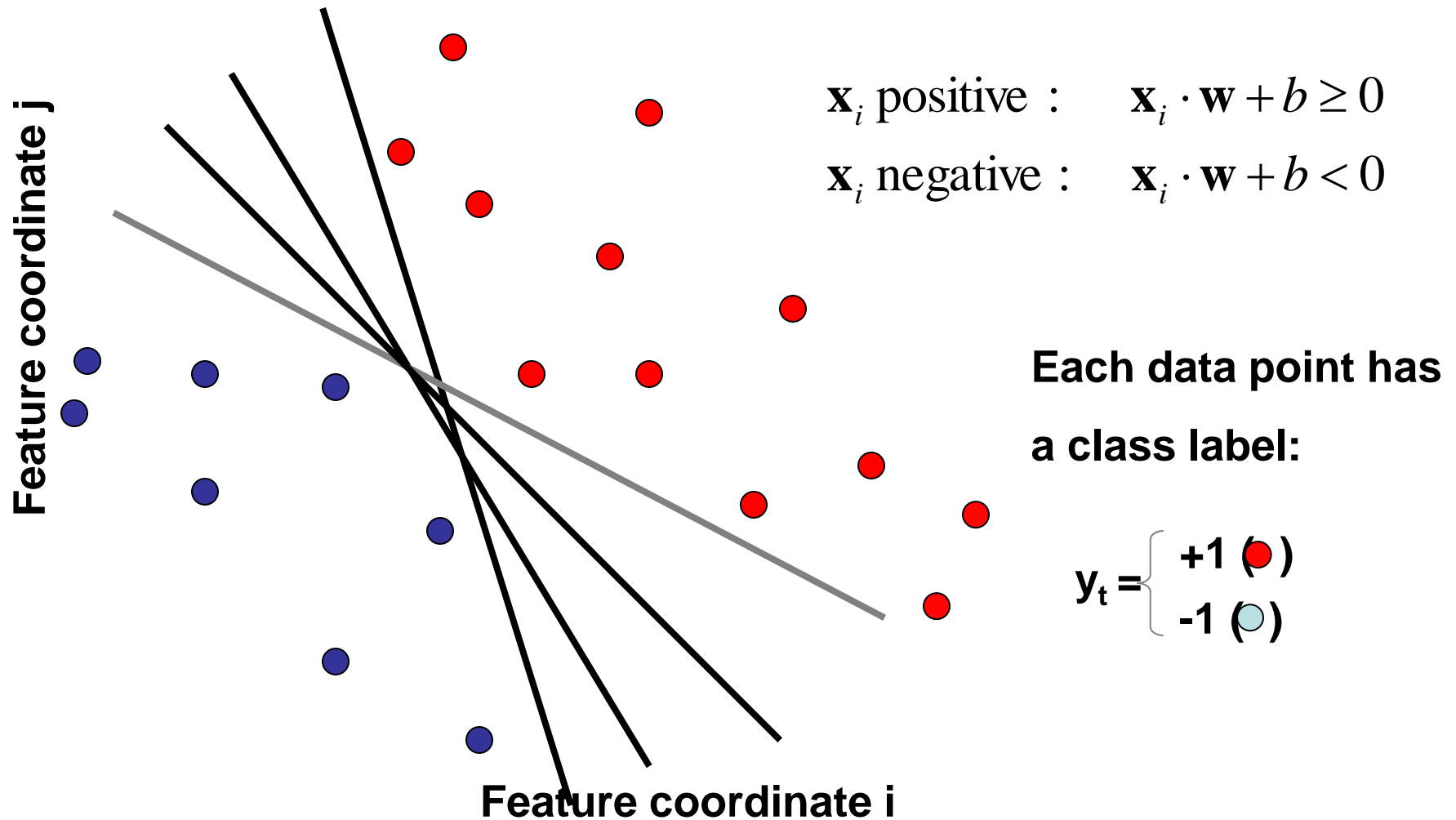
$$\mathbf{w}^T \mathbf{x} = \langle \mathbf{w}, \mathbf{x} \rangle = \sum_{d=1}^D \mathbf{w}_d \mathbf{x}_d$$

$$\mathbf{x} \in \mathbb{R}^D, \mathbf{w} \in \mathbb{R}^D$$

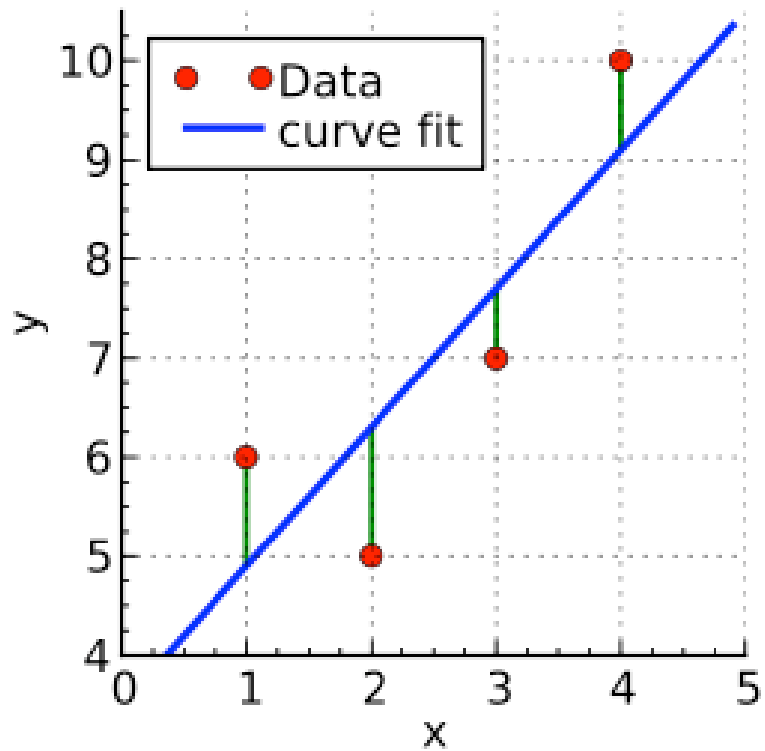
Reminder: linear classifier



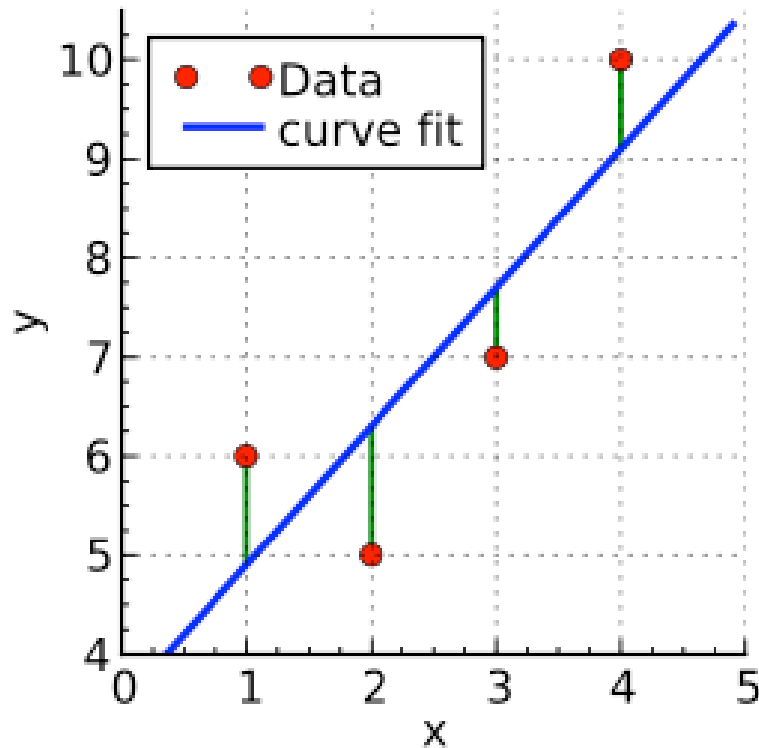
Question: which one?



Linear regression in 1D



Linear regression in 1D



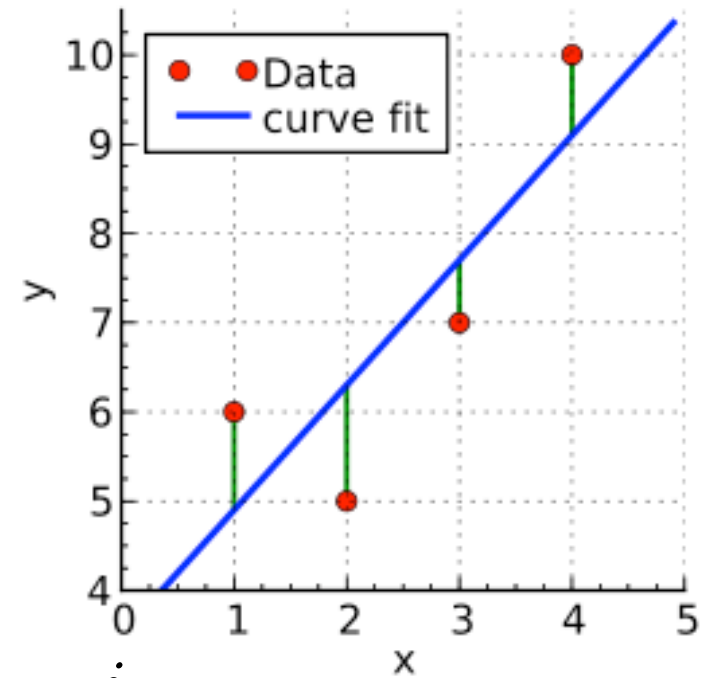
Training set: input–output pairs $\mathcal{S} = \{(x^i, y^i)\}, \quad i = 1 \dots, N$
 $x^i \in \mathbb{R}, \quad y^i \in \mathbb{R}$

Linear regression in 1D

$$y^i = w_0 + w_1 x_1^i + \epsilon^i$$

$$= w_0 x_0^i + w_1 x_1^i + \epsilon^i, \quad x_0^i = 1, \quad \forall i$$

$$= \mathbf{w}^T \mathbf{x}^i + \epsilon^i$$

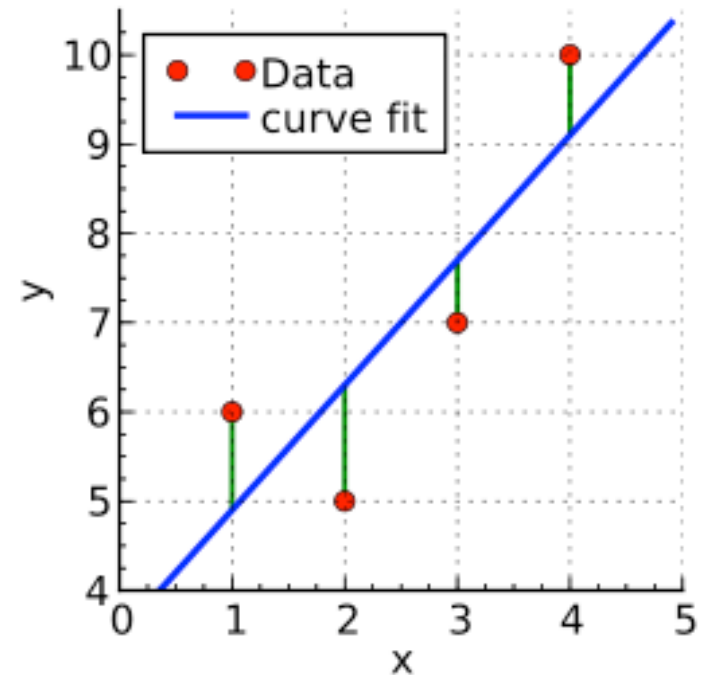


Sum of squared errors criterion

$$y^i = \mathbf{w}^T \mathbf{x}^i + \epsilon^i$$

Loss function: sum of squared errors

$$L(\mathbf{w}) = \sum_{i=1}^N (\epsilon^i)^2$$



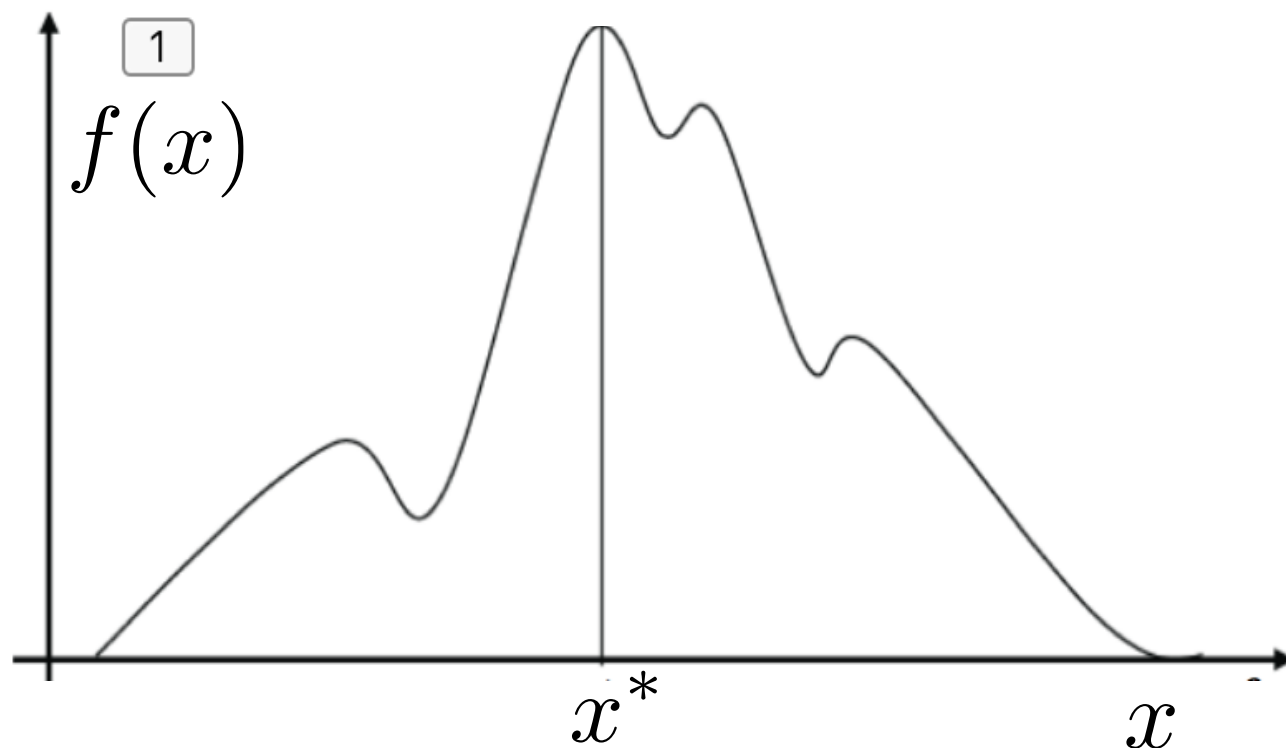
Expressed as a function of two variables:

$$L(w_0, w_1) = \sum_{i=1}^N [y^i - (w_0 x_0^i + w_1 x_1^i)]^2$$

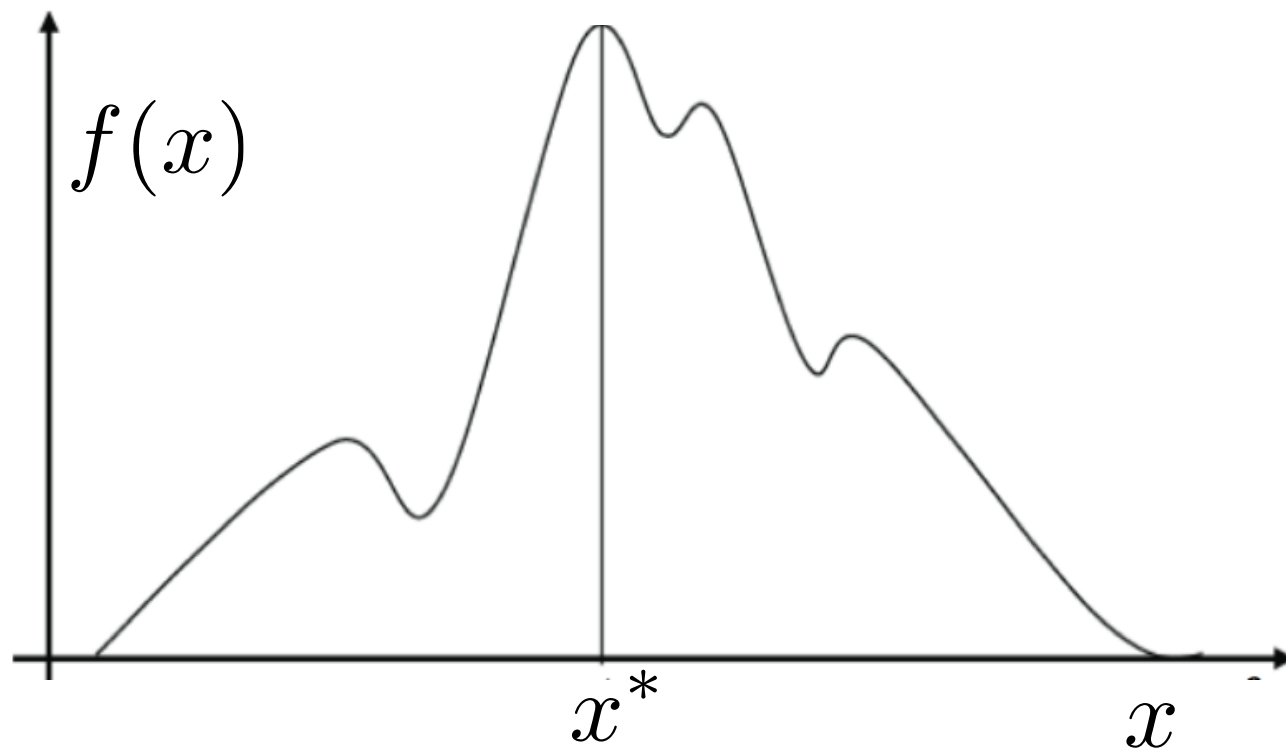
Question: what is the best (or least bad) value of w?

Answer: least squares

Calculus

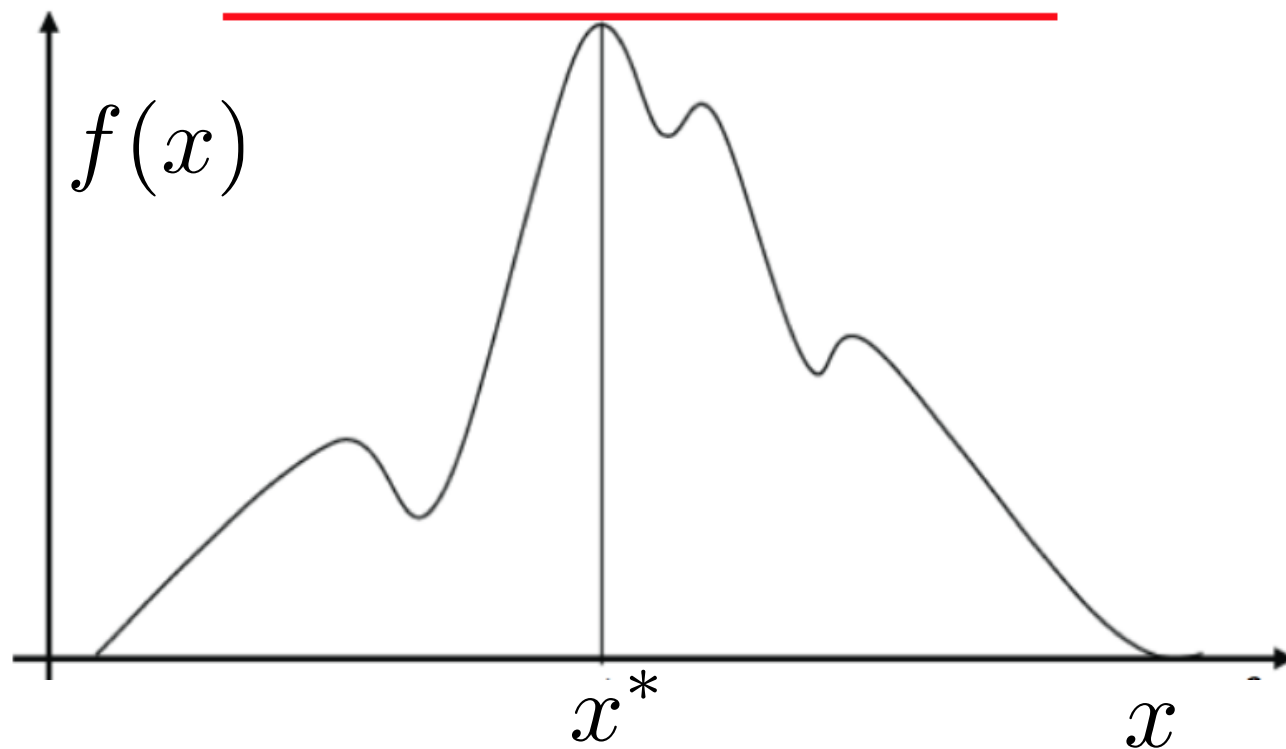


Calculus



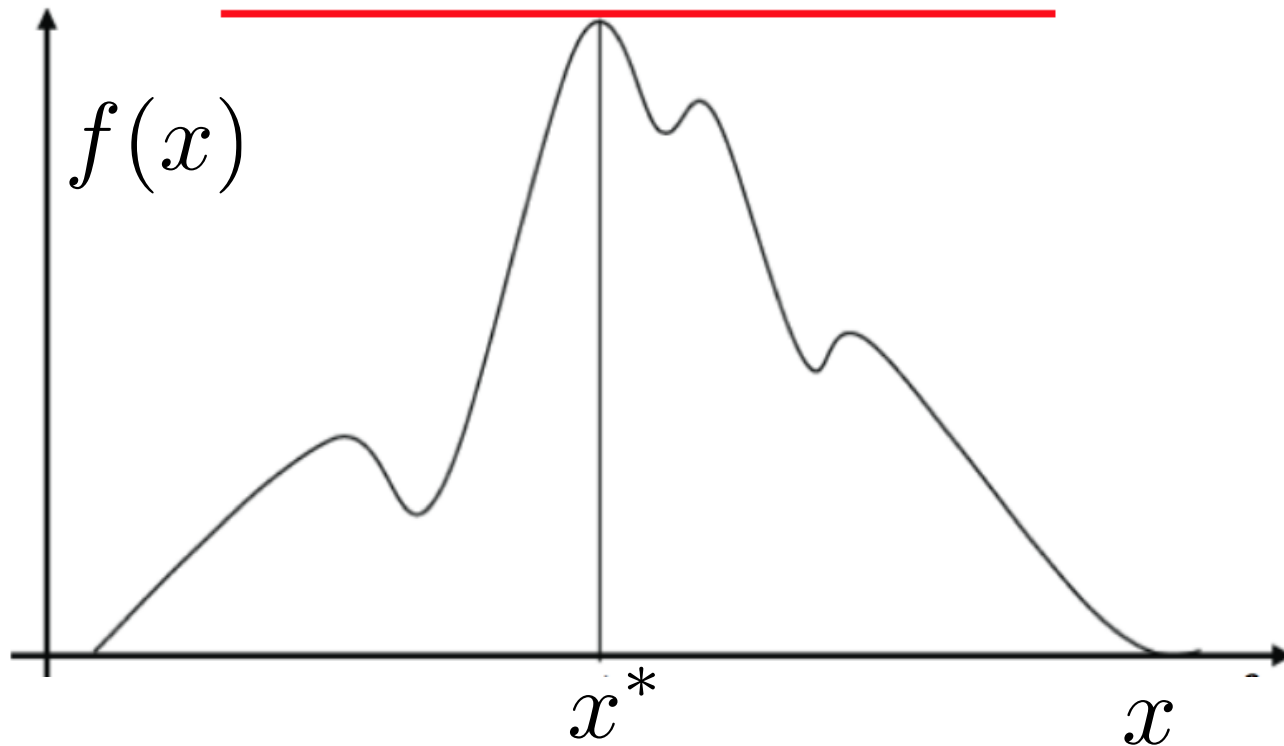
$$x^* = \operatorname{argmax}_x f(x)$$

Condition for maximum: derivative is zero



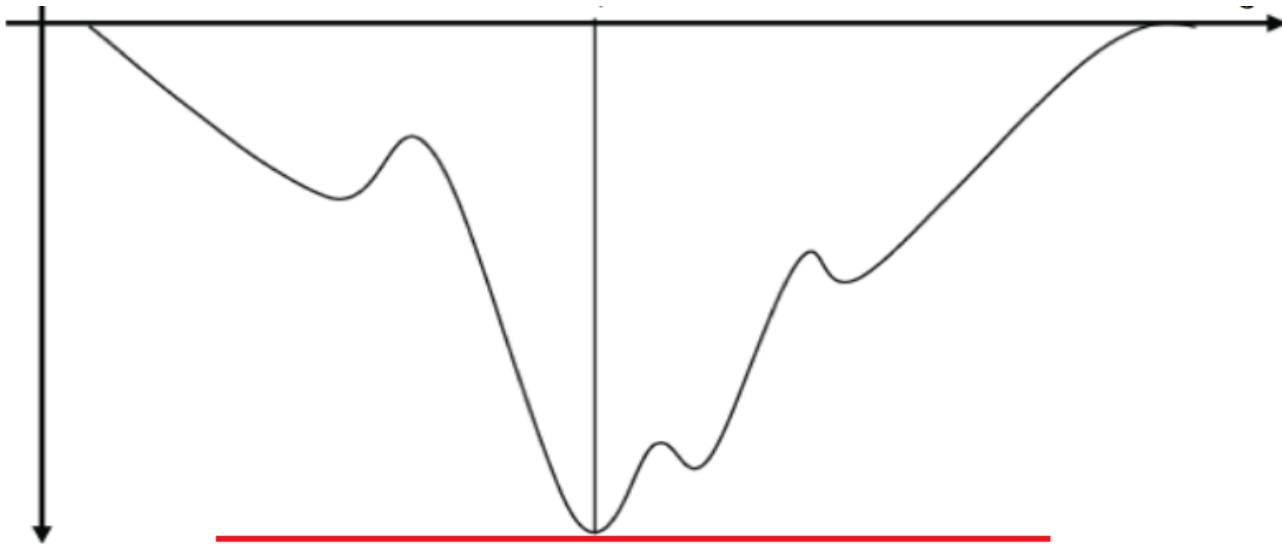
$$x^* = \operatorname{argmax}_x f(x)$$

Condition for maximum: derivative is zero



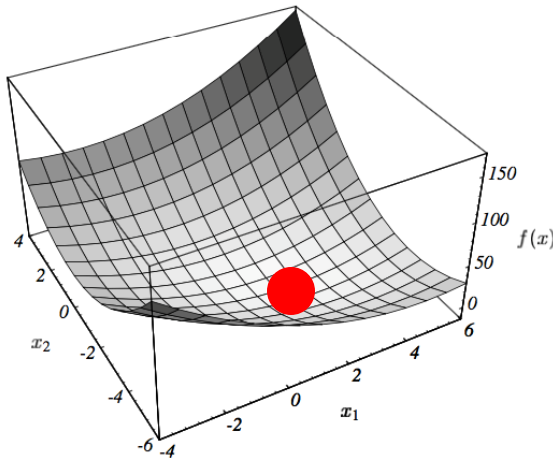
$$x^* = \operatorname{argmax}_x f(x) \quad \rightarrow \quad f'(x^*) = 0$$

Condition for minimum: derivative is zero



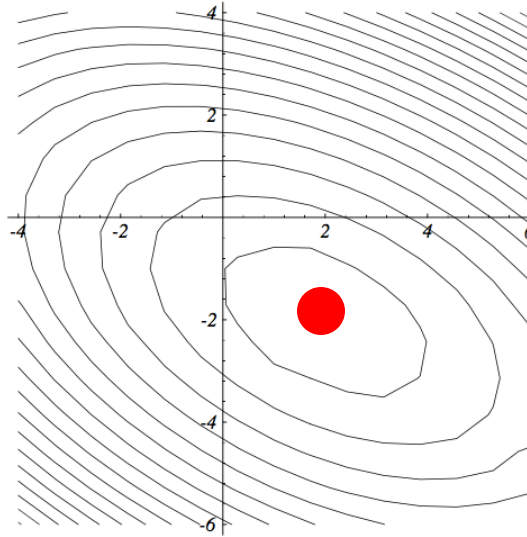
$$x^* = \operatorname{argmin}_x f(x) \quad \rightarrow \quad f'(x^*) = 0$$

Vector calculus



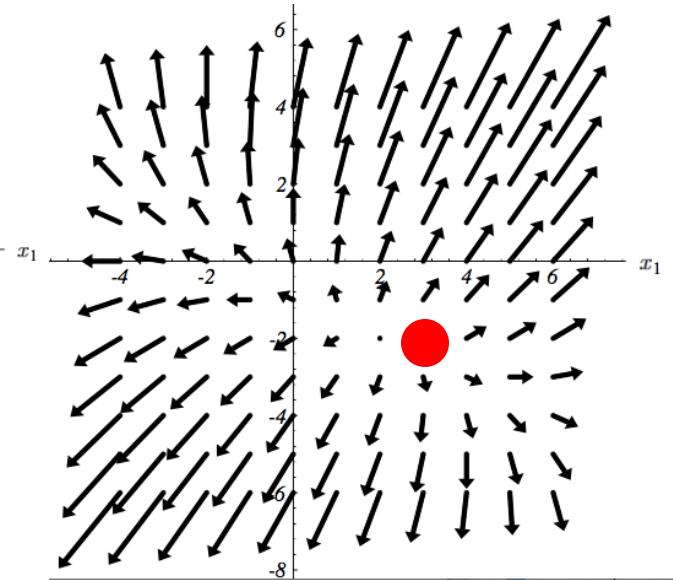
$$f(\mathbf{x})$$

2D function graph



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

isocontours



$$\nabla f(\mathbf{x}) = \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \end{bmatrix}$$

gradient field

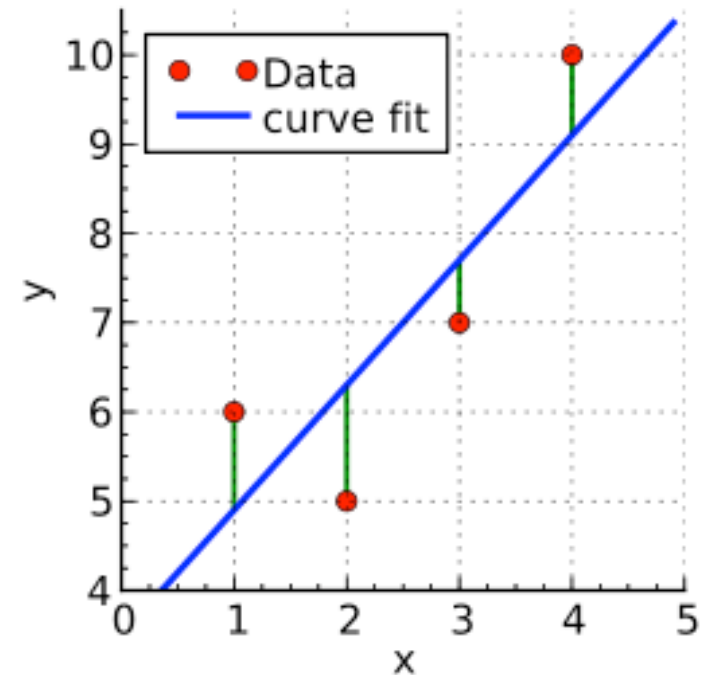
● at minimum of function: $\nabla f(\mathbf{x}) = \mathbf{0}$

Back to least squares..

$$y^i = \mathbf{w}^T \mathbf{x}^i + \epsilon^i$$

Loss function: sum of squared errors

$$L(\mathbf{w}) = \sum_{i=1}^N (\epsilon^i)^2$$



Expressed as a function of two variables:

$$L(w_0, w_1) = \sum_{i=1}^N [y^i - (w_0 x_0^i + w_1 x_1^i)]^2$$

training sample

feature dimension

Question: what is the best (or least bad) value of \mathbf{w} ?

Answer: least squares

Fitting a line

$$L(w_0, w_1) = \sum_{i=1}^N [y^i - (w_0 x_0^i + w_1 x_1^i)]^2$$

$$\begin{aligned} \frac{\partial L(w_0, w_1)}{\partial w_0} &= \sum_{i=1}^N \frac{\partial [y^i - (w_0 x_0^i + w_1 x_1^i)]^2}{\partial w_0} \\ &= \sum_{i=1}^N 2 [y^i - (w_0 x_0^i + w_1 x_1^i)] (-x_0^i) \\ &= -2 \sum_{i=1}^N (y^i x_0^i - w_0 x_0^i x_0^i - w_1 x_1^i x_0^i) \end{aligned}$$

$$\begin{aligned} \frac{\partial L(w_0, w_1)}{\partial w_0} = 0_N &\Leftrightarrow \sum_{i=1}^N y^i x_0^i = w_0 \sum_{i=1}^N x_0^i x_0^i + w_1 \sum_{i=1}^N x_1^i x_0^i \end{aligned}$$

Fitting a line, continued

$$\frac{\partial L(w_0, w_1)}{\partial w_0} = 0 \quad \Leftrightarrow \quad \sum_{i=1}^N y^i x_0^i = w_0 \sum_{i=1}^N x_0^i x_0^i + w_1 \sum_{i=1}^N x_1^i x_0^i$$

$$\frac{\partial L(w_0, w_1)}{\partial w_1} = 0 \quad \Leftrightarrow \quad \sum_{i=1}^N y^i x_1^i = w_0 \sum_{i=1}^N x_0^i x_1^i + w_1 \sum_{i=1}^N x_1^i x_1^i$$

2 linear equations, 2 unknowns

Fitting a line, continued

$$\sum_{i=1}^N y^i x_0^i = w_0 \sum_{i=1}^N x_0^i x_0^i + w_1 \sum_{i=1}^N x_1^i x_0^i$$

$$\sum_{i=1}^N y^i x_1^i = w_0 \sum_{i=1}^N x_0^i x_1^i + w_1 \sum_{i=1}^N x_1^i x_1^i$$

2x2 system of equations:

$$\begin{bmatrix} \sum_{i=1}^N y^i x_0^i \\ \sum_{i=1}^N y^i x_1^i \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^N x_0^i x_0^i & \sum_{i=1}^N x_0^i x_1^i \\ \sum_{i=1}^N x_0^i x_1^i & \sum_{i=1}^N x_1^i x_1^i \end{bmatrix} \begin{bmatrix} w_0 \\ w_1 \end{bmatrix}$$

That's it!

Fitting a line, continued

2x2 system of equations:

$$\begin{bmatrix} \sum_{i=1}^N y^i x_0^i \\ \sum_{i=1}^N y^i x_1^i \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^N x_0^i x_0^i & \sum_{i=1}^N x_0^i x_1^i \\ \sum_{i=1}^N x_0^i x_1^i & \sum_{i=1}^N x_1^i x_1^i \end{bmatrix} \begin{bmatrix} w_0 \\ w_1 \end{bmatrix}$$

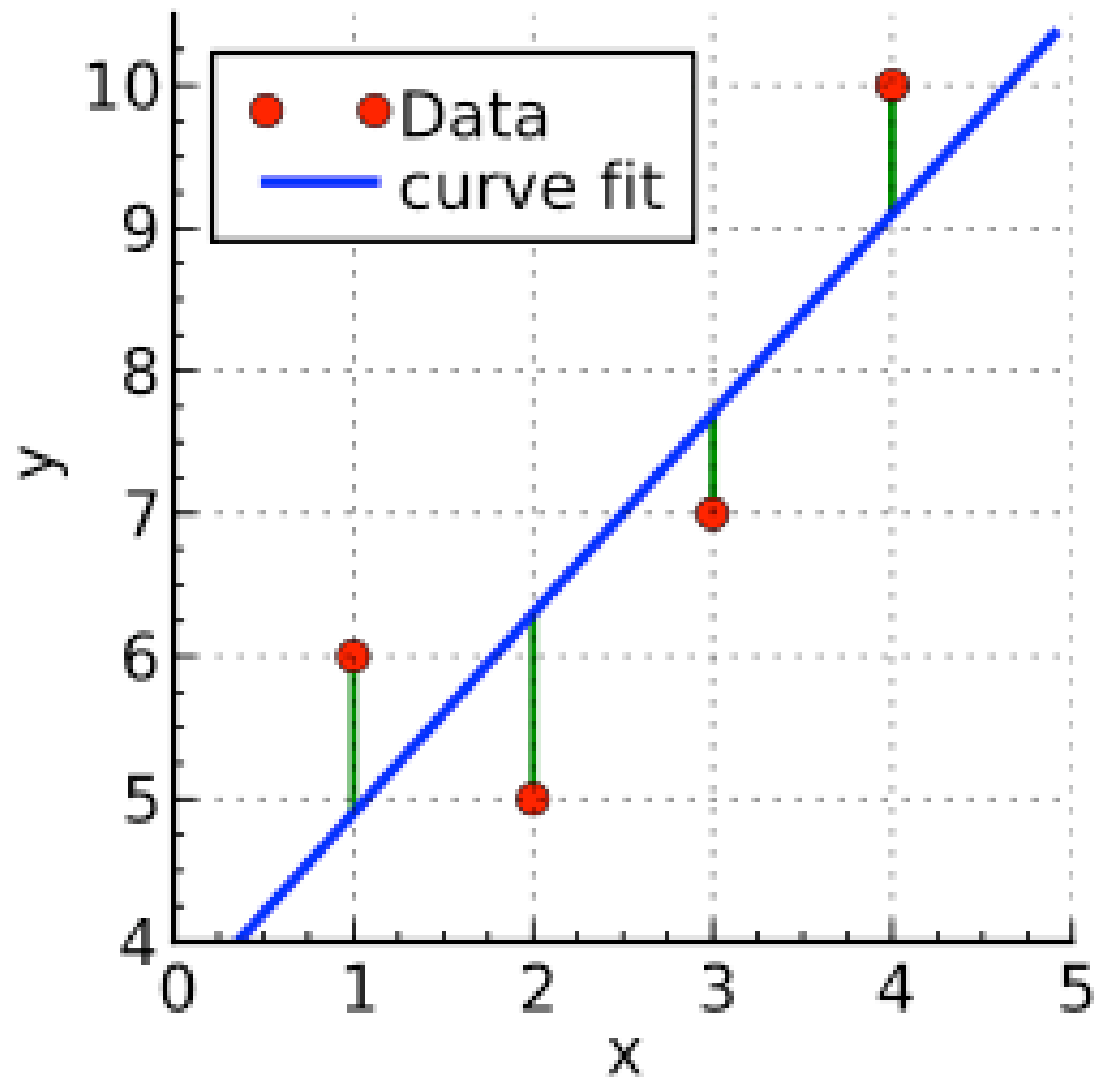
Or, without summations:

$$\mathbf{X}^T \mathbf{y} = \mathbf{X}^T \mathbf{X} \mathbf{w}$$

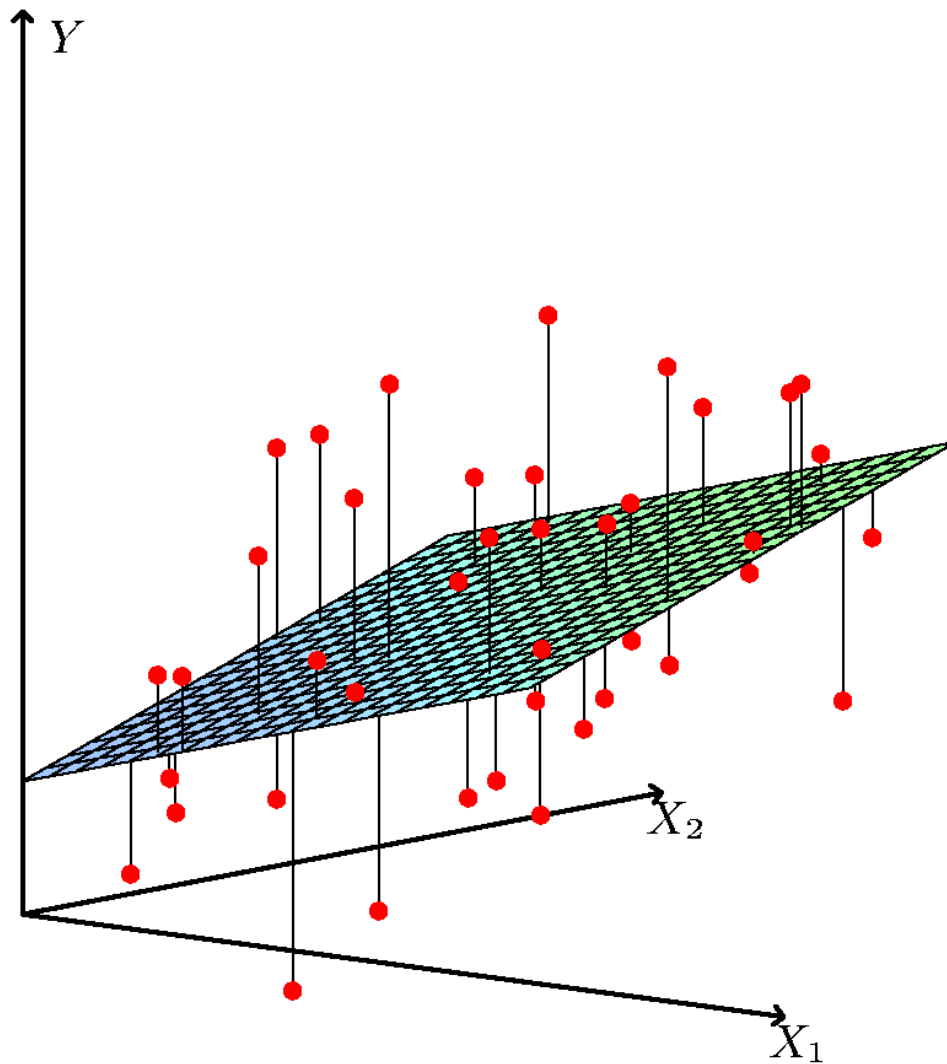
$$\mathbf{y} = \begin{bmatrix} y^1 \\ \vdots \\ y^N \end{bmatrix} \quad \mathbf{X} = \begin{bmatrix} x_0^1 & x_1^1 \\ \vdots & \vdots \\ x_0^N & x_1^N \end{bmatrix}$$

Solution: $\mathbf{w} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y}$

Linear regression in 1D



Linear regression in 2D (or ND)



Least squares solution for linear regression

D: problem dimension

$$\begin{array}{c} \text{N: training set size} \end{array} \begin{bmatrix} y^1 \\ y^2 \\ \vdots \\ y^N \end{bmatrix} \begin{array}{c} \text{Nx1} \end{array} = \begin{bmatrix} x_1^1 & \dots & x_D^1 \\ x_1^2 & \dots & x_D^2 \\ \vdots & & \vdots \\ x_1^N & \dots & x_D^N \end{bmatrix} \begin{array}{c} \text{NxD} \end{array} \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_D \end{bmatrix} \begin{array}{c} \text{Dx1} \end{array} + \begin{bmatrix} \epsilon^1 \\ \epsilon^2 \\ \vdots \\ \epsilon^N \end{bmatrix} \begin{array}{c} \text{Nx1} \end{array}$$

Least squares solution for linear regression

$$\mathbf{y} = \mathbf{X}\mathbf{w} + \boldsymbol{\epsilon}$$

Least squares solution for linear regression

Loss function:
$$L(\mathbf{w}) = \sum_{i=1}^N (y^i - \mathbf{w}^T \mathbf{x}^i)^2 = \sum_{i=1}^N (\epsilon^i)^2$$

$$L(\mathbf{w}) = \begin{bmatrix} \epsilon^1 & \epsilon^2 & \dots & \epsilon^N \end{bmatrix} \begin{bmatrix} \epsilon^1 \\ \epsilon^2 \\ \vdots \\ \epsilon^N \end{bmatrix}$$

Least squares solution for linear regression

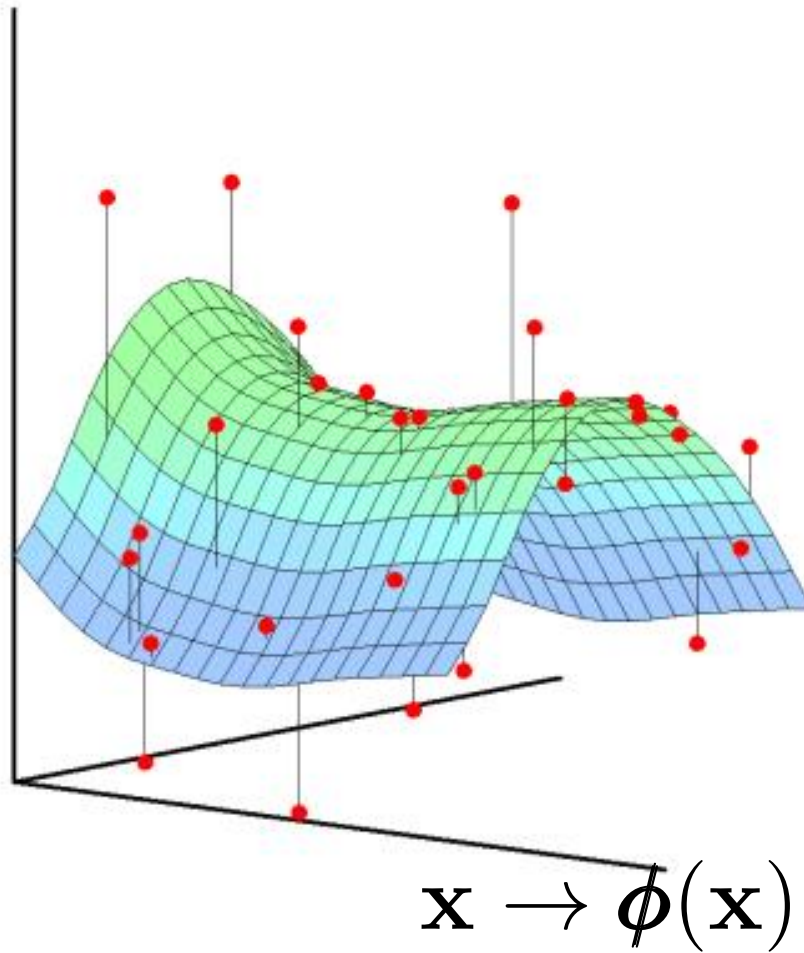
Loss function:
$$L(\mathbf{w}) = \sum_{i=1}^N (y^i - \mathbf{w}^T \mathbf{x}^i)^2 = \sum_{i=1}^N (\epsilon^i)^2$$

$$L(\mathbf{w}) = \begin{bmatrix} \epsilon^1 & \epsilon^2 & \dots & \epsilon^N \end{bmatrix} \begin{bmatrix} \epsilon^1 \\ \epsilon^2 \\ \vdots \\ \epsilon^N \end{bmatrix}$$

$$L(\mathbf{w}) = \boldsymbol{\epsilon}^T \boldsymbol{\epsilon}$$

$$\mathbf{y} = \mathbf{X}\mathbf{w} + \boldsymbol{\epsilon}$$

Generalized linear regression



$$\mathbf{x} \rightarrow \boldsymbol{\phi}(\mathbf{x}) = \begin{bmatrix} \phi_1(\mathbf{x}) \\ \vdots \\ \phi_M(\mathbf{x}) \end{bmatrix}$$

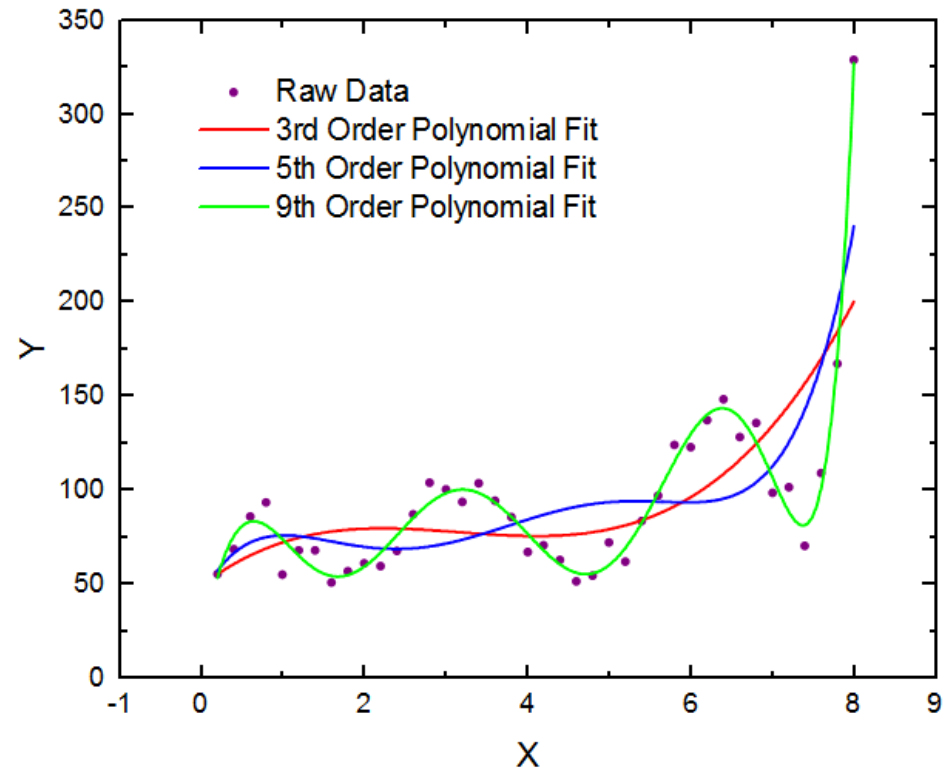
1D Example: 2nd degree polynomial fitting

$$\phi(x) = \begin{bmatrix} 1 \\ x \\ (x)^2 \end{bmatrix}$$

$$\langle \mathbf{w}, \phi(x) \rangle = w_0 + w_1 x + w_2 (x)^2$$

1D Example: k-th degree polynomial fitting

$$\phi(\mathbf{x}) = \begin{bmatrix} 1 \\ x \\ \vdots \\ (x)^K \end{bmatrix}$$



$$\langle \mathbf{w}, \phi(x) \rangle = w_0 + w_1 x + \dots + w_k (x)^K$$

2D example: second-order polynomials

$$\mathbf{x} = (x_1, x_2)$$

$$\phi(\mathbf{x}) = \begin{bmatrix} 1 \\ x_1 \\ x_2 \\ (x_1)^2 \\ (x_2)^2 \\ x_1 x_2 \end{bmatrix}$$

$$\langle \mathbf{w}, \phi(\mathbf{x}) \rangle = w_0 + w_1 x_1 + w_2 x_2 + w_3 x_1^2 + w_4 x_2^2 + w_5 x_1 x_2$$

Reminder: linear regression

Loss function:
$$L(\mathbf{w}) = \sum_{i=1}^N (y^i - \mathbf{w}^T \mathbf{x}^i)^2 = \sum_{i=1}^N (\epsilon^i)^2$$

$$\begin{bmatrix} y^1 \\ y^2 \\ \vdots \\ y^N \end{bmatrix} = \begin{bmatrix} x_1^1 & \dots & x_D^1 \\ x_1^2 & \dots & x_D^2 \\ \vdots & & \vdots \\ x_1^N & \dots & x_D^N \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_D \end{bmatrix} + \begin{bmatrix} \epsilon^1 \\ \epsilon^2 \\ \vdots \\ \epsilon^N \end{bmatrix}$$

Reminder: linear regression

Loss function:
$$L(\mathbf{w}) = \sum_{i=1}^N (y^i - \mathbf{w}^T \mathbf{x}^i)^2 = \sum_{i=1}^N (\epsilon^i)^2$$

$$\begin{bmatrix} y^1 \\ y^2 \\ \vdots \\ y^N \end{bmatrix} = \begin{bmatrix} \frac{(\mathbf{x}^1)^T}{(\mathbf{x}^2)^T} \\ \vdots \\ \frac{(\mathbf{x}^N)^T}{(\mathbf{x}^N)^T} \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_D \end{bmatrix} + \begin{bmatrix} \epsilon^1 \\ \epsilon^2 \\ \vdots \\ \epsilon^N \end{bmatrix}$$

Generalized linear regression

Loss function:
$$L(\mathbf{w}) = \sum_{i=1}^N (y^i - \mathbf{w}^T \boldsymbol{\phi}(\mathbf{x}^i))^2 = \sum_{i=1}^N (\epsilon^i)^2$$

$$\begin{bmatrix} y^1 \\ y^2 \\ \vdots \\ y^N \end{bmatrix}_{\mathbf{N} \times 1} = \begin{bmatrix} \frac{\boldsymbol{\phi}(\mathbf{x}^1)^T}{\boldsymbol{\phi}(\mathbf{x}^2)^T} \\ \vdots \\ \boldsymbol{\phi}(\mathbf{x}^N)^T \end{bmatrix}_{\mathbf{N} \times \mathbf{M}} \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_M \end{bmatrix}_{\mathbf{M} \times 1} + \begin{bmatrix} \epsilon^1 \\ \epsilon^2 \\ \vdots \\ \epsilon^N \end{bmatrix}_{\mathbf{N} \times 1}$$

$$\boldsymbol{\phi}(\mathbf{x}) : \mathbb{R}^D \rightarrow \mathbb{R}^M$$

Least squares solution for linear regression

$$\mathbf{y} = \mathbf{X}\mathbf{w} + \boldsymbol{\epsilon} \quad \mathbf{X} = \begin{bmatrix} (\mathbf{x}^1)^T \\ \hline (\mathbf{x}^2)^T \\ \hline \vdots \\ \hline (\mathbf{x}^N)^T \end{bmatrix}$$
$$L(\mathbf{w}) = \boldsymbol{\epsilon}^T \boldsymbol{\epsilon}$$

Minimize:

$$\mathbf{w}^* = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y}$$

Least squares solution for generalized linear regression

$$\mathbf{y} = \Phi \mathbf{w} + \boldsymbol{\epsilon} \quad \Phi = \begin{bmatrix} \frac{\phi(\mathbf{x}^1)^T}{\phi(\mathbf{x}^2)^T} \\ \vdots \\ \frac{\phi(\mathbf{x}^N)^T}{\phi(\mathbf{x}^N)^T} \end{bmatrix}$$
$$L(\mathbf{w}) = \boldsymbol{\epsilon}^T \boldsymbol{\epsilon}$$

Minimize:

$$\mathbf{w}^* = (\Phi^T \Phi)^{-1} \Phi \mathbf{y}$$

2D example: second-order polynomials

$$\mathbf{x} = (x_1, x_2)$$

$$\phi(\mathbf{x}) = \begin{bmatrix} 1 \\ x_1 \\ x_2 \\ (x_1)^2 \\ (x_2)^2 \\ x_1 x_2 \end{bmatrix}$$

$$\langle \mathbf{w}, \phi(\mathbf{x}) \rangle = w_0 + w_1 x_1 + w_2 x_2 + w_3 x_1^2 + w_4 x_2^2 + w_5 x_1 x_2$$

5D Example: fourth-order polynomials in 5D

$$\mathbf{x} = (x_1, \dots, x_5)$$

$$\phi(\mathbf{x}) = \begin{bmatrix} 1 \\ x_1 \\ \vdots \\ x_5 \\ \vdots \\ (x_1 x_2 x_3 x_4 x_5)^4 \end{bmatrix}$$

15625 Dimensions => 15625 parameters

What was happening before: approximations

Training: $S = \{(\mathbf{x}^i, y^i)\}, i = 1, \dots, N$

$$y^1 \simeq w_0 x_0^1 + w_1 x_1^1 + \dots + w_D x_D^1$$

$$y^2 \simeq w_0 x_0^2 + w_1 x_1^2 + \dots + w_D x_D^2$$

$$\vdots$$

$$y^N \simeq w_0 x_0^N + w_1 x_1^N + \dots + w_D x_D^N$$

If $N > D$ (e.g. 30 points, 2 dimensions) we have more equations than unknowns: **overdetermined** system!

Input-output relations can only hold approximately!

What is happening now: overfitting

Training: $S = \{(\mathbf{x}^i, y^i)\}, i = 1, \dots, N$

$$y^1 = w_0 x_0^1 + w_1 x_1^1 + \dots + w_D x_D^1$$

$$y^2 = w_0 x_0^2 + w_1 x_1^2 + \dots + w_D x_D^2$$

$$\vdots$$

$$y^N = w_0 x_0^N + w_1 x_1^N + \dots + w_D x_D^N$$

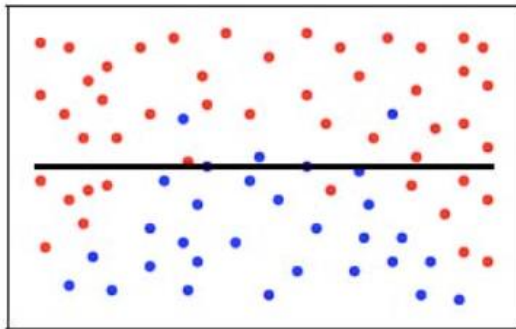
If $N < D$ (e.g. 30 points, 15265 dimensions) we have more unknowns than equations: **underdetermined** system!

Input-output equations hold exactly, but we are simply memorizing data

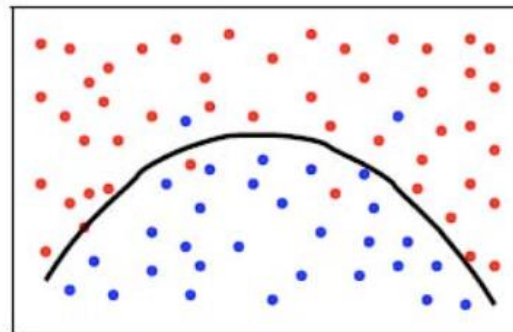
Overfitting, in images

Classification

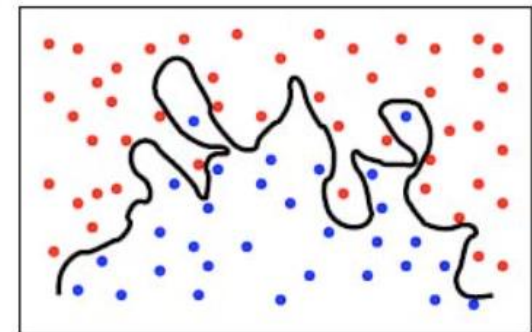
Underfitting



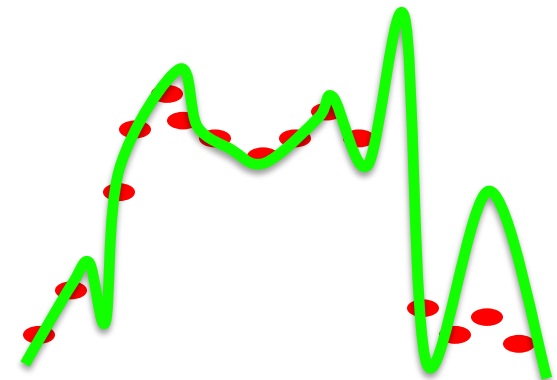
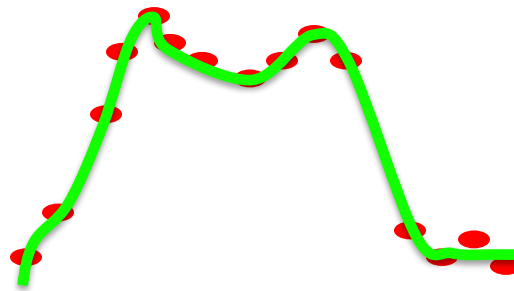
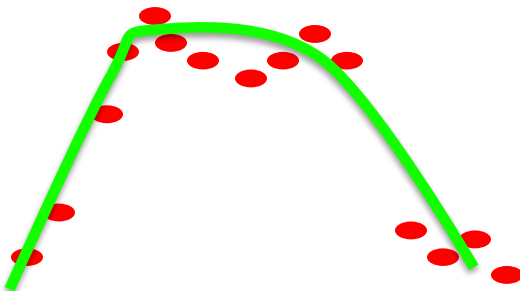
just right



Overfitting



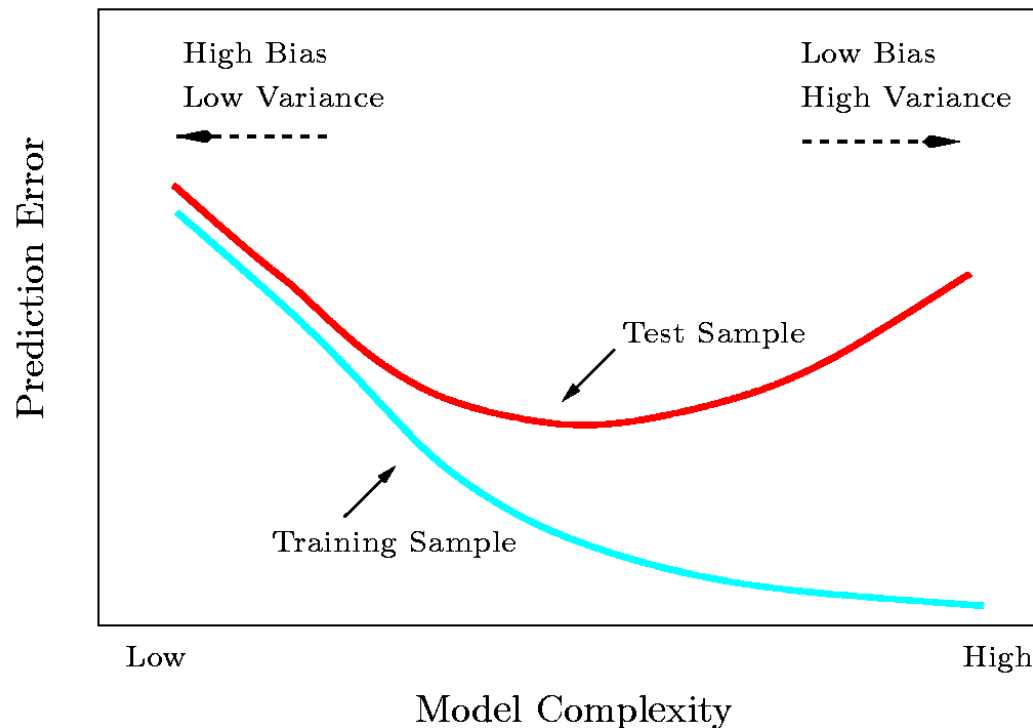
Regression



Tuning the model's complexity

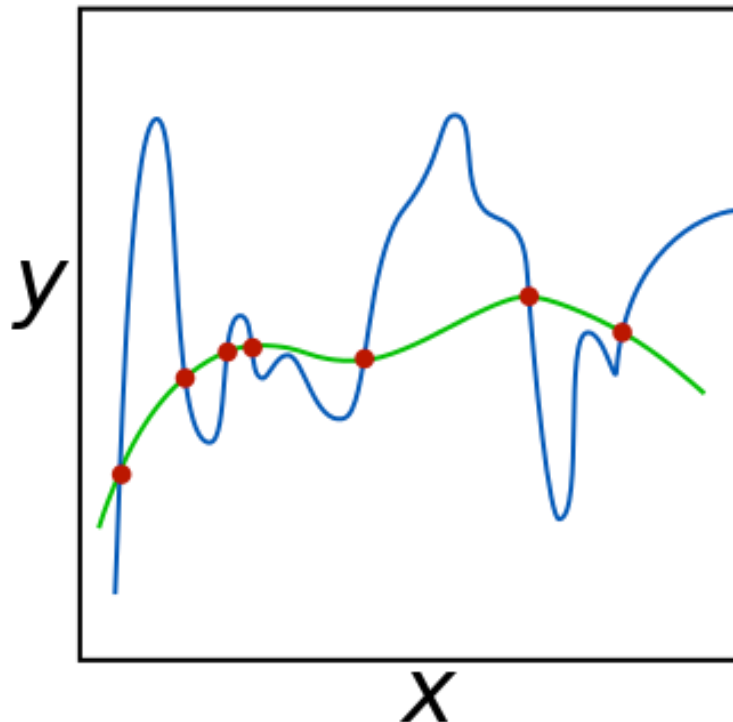
A flexible model approximates the target function well in the training set
but can “overtrain” and have poor performance on the test set (“variance”)

A rigid model's performance is more predictable in the test set
but the model may not be good even on the training set (“bias”)



Regularization: keeping it simple

In high dimensions: too many solutions for the same problem



Regularization: prefer the least complex among them

How? Penalize complexity

How to control complexity?

Observation: problem started with high-dimensional embeddings

Guess: Number of dimensions relates to “complexity”

Intuition: with many parameters, we can fit anything

But what if we force the regressor not to use all of the parameters?

Idea: penalize the use of large parameter values

How do we measure “large”?

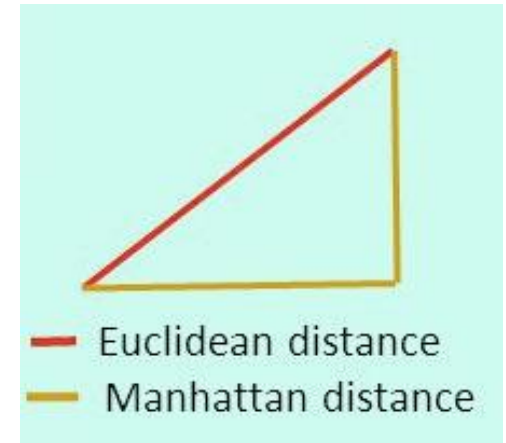
How do we enforce small values?

How do we measure “large”?

Method parameters: D-dimensional vector

$$\mathbf{w} = [w_1, w_2, \dots, w_D]$$

“Large” vector: vector **norm**



L2, (“euclidean”) norm:

$$\|\mathbf{w}\|_2 \doteq \sqrt{\sum_{d=1}^D w_d^2} = \sqrt{\langle \mathbf{w}, \mathbf{w} \rangle}$$

L1, (“manhattan”) norm:

$$\|\mathbf{w}\|_1 \doteq \sum_{d=1}^D |w_d|$$

Lp norm, $p > 1$:

$$\|\mathbf{w}\|_p \doteq \left(\sum_{d=1}^D w_d^p \right)^{1/p}$$

Regularized linear regression

$$\boldsymbol{\epsilon} = \mathbf{y} - \Phi \mathbf{w}$$

residual vector


$$L(\mathbf{w}) = \boldsymbol{\epsilon}^T \boldsymbol{\epsilon}$$

linear regression: minimize model error

Complexity term:
(regularizer)


$$R(\mathbf{w}) \doteq \|\mathbf{w}\|_2^2 = \mathbf{w}^T \mathbf{w}$$

$$L(\mathbf{w}) = \boldsymbol{\epsilon}^T \boldsymbol{\epsilon} + \lambda \mathbf{w}^T \mathbf{w}$$



minimum remains
to be determined

“data fidelity”



complexity

scalar, remains to
be determined

Least squares solution

$$\begin{aligned} L(\mathbf{w}) &= \boldsymbol{\epsilon}^T \boldsymbol{\epsilon} \\ &= (\mathbf{y} - \mathbf{X}\mathbf{w})^T (\mathbf{y} - \mathbf{X}\mathbf{w}) \\ &= \mathbf{y}^T \mathbf{y} - 2\mathbf{y}^T \mathbf{X}\mathbf{w} + \mathbf{w}^T \mathbf{X}^T \mathbf{X} \mathbf{w} \end{aligned}$$

Condition for minimum:

$$\nabla L(\mathbf{w}^*) = \mathbf{0}$$

$$-2\mathbf{X}^T \mathbf{y} + 2\mathbf{X}^T \mathbf{X} \mathbf{w}^* = \mathbf{0}$$

$$\mathbf{w}^* = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y}$$

Ridge regression: L2-regularized linear regression

$$\begin{aligned} L(\mathbf{w}) &= \boldsymbol{\epsilon}^T \boldsymbol{\epsilon} + \lambda \mathbf{w}^T \mathbf{w} \\ &= \mathbf{y}^T \mathbf{y} - 2\mathbf{y}^T \mathbf{X} \mathbf{w} + \mathbf{w}^T \mathbf{X}^T \mathbf{X} \mathbf{w} + \lambda \mathbf{w}^T \mathbf{I} \mathbf{w} \\ &\quad \text{as before, for linear regression} \qquad \text{identity matrix} \\ &= \mathbf{y}^T \mathbf{y} - 2\mathbf{y}^T \mathbf{X} \mathbf{w} + \mathbf{w}^T (\mathbf{X}^T \mathbf{X} + \lambda \mathbf{I}) \mathbf{w} \end{aligned}$$

Condition for minimum:


$$\begin{aligned} \nabla L(\mathbf{w}^*) &= \mathbf{0} \\ -2\mathbf{X}^T \mathbf{y} + 2(\mathbf{X}^T \mathbf{X} + \lambda \mathbf{I}) \mathbf{w}^* &= \mathbf{0} \\ \mathbf{w}^* &= (\mathbf{X}^T \mathbf{X} + \lambda \mathbf{I})^{-1} \mathbf{X}^T \mathbf{y} \end{aligned}$$


Ridge regression, continued

Regularizer: $R(\mathbf{w}) \doteq \|\mathbf{w}\|_2^2 = \mathbf{w}^T \mathbf{w}$

New objective:

$$L(\mathbf{w}) = \underbrace{\boldsymbol{\epsilon}^T \boldsymbol{\epsilon}}_{\text{"data fidelity"}} + \underbrace{\lambda \mathbf{w}^T \mathbf{w}}_{\text{complexity}}$$

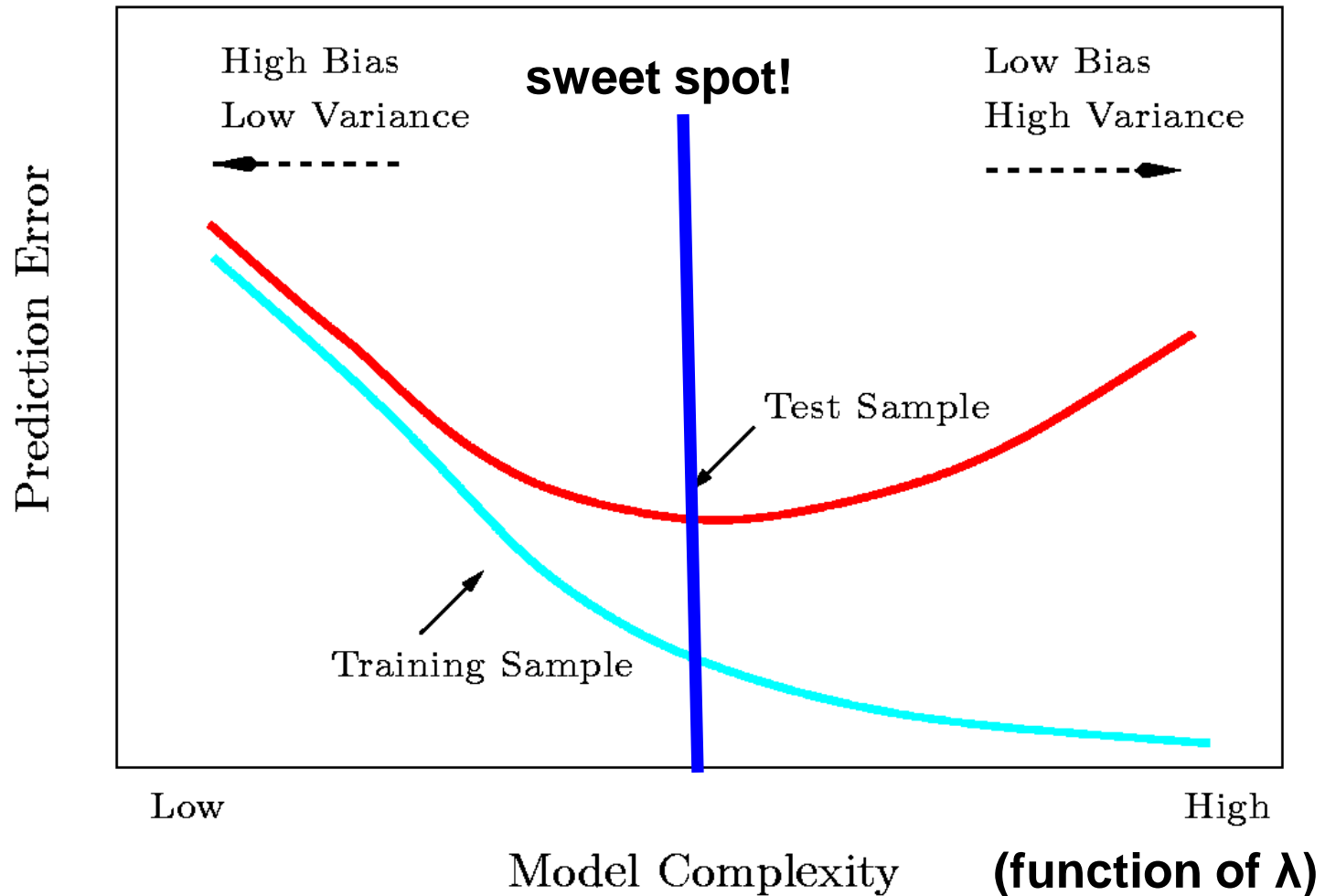

We just
determined
minimum


scalar, remains to
be determined

λ : “hyperparameter”

Note: direct minimization w.r.t. it would lead to $\lambda=0$

Bias-Variance tradeoff as a function of λ



Selecting λ with cross-validation

- Cross validation technique
 - Exclude part of the training data from parameter estimation
 - Use them only to predict the test error
- K-fold cross validation:
 - K splits, average K errors
- Use cross-validation for different values of λ parameter
 - pick value that minimizes cross-validation error

**Least glorious, most effective
of all methods**

