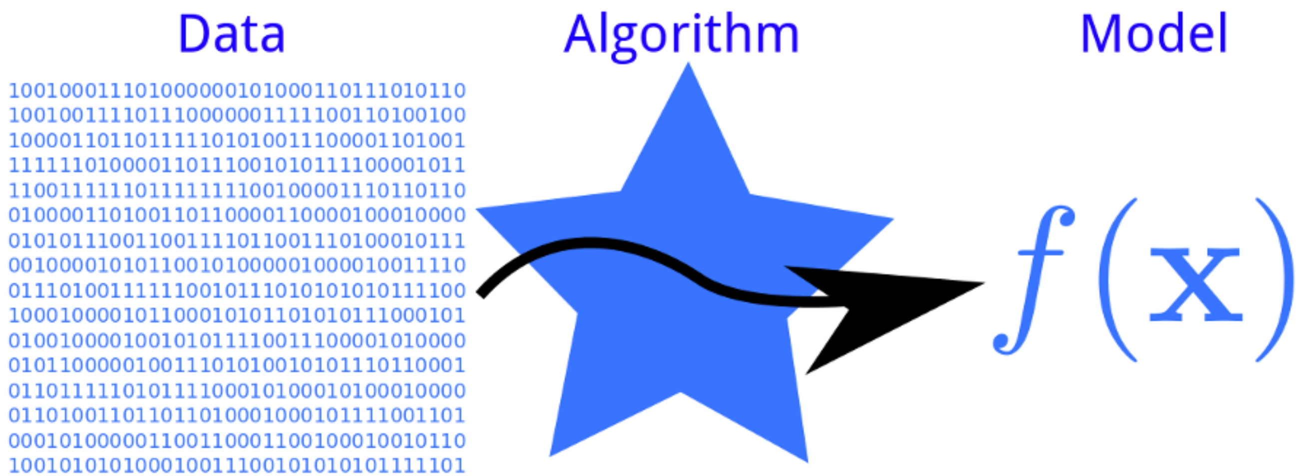
Introduction to Machine Learning



Week 1: Introduction

Iasonas Kokkinos

Iasonas.kokkinos@gmail.com

University College London

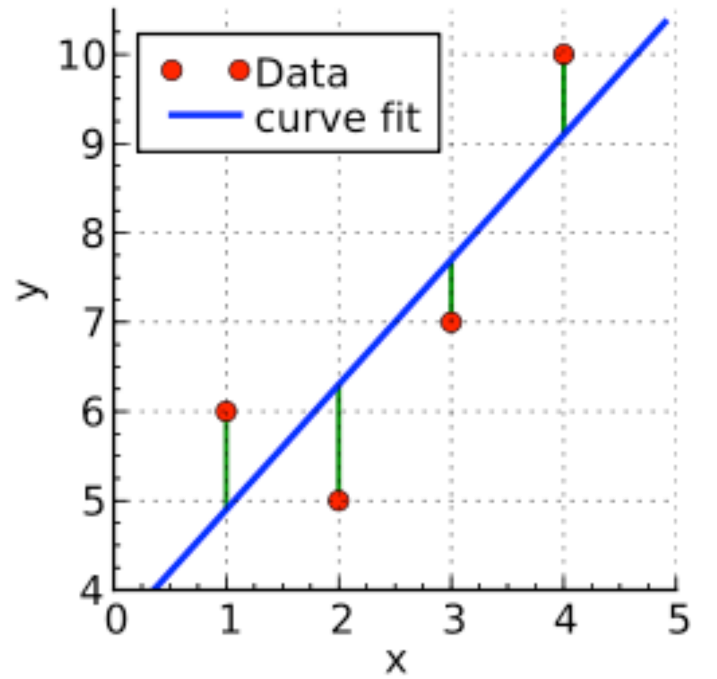
Lectures 1-7: all of that

- 1st week: Introduction & linear regression.
- 2nd week: Regularization, Ridge Regression, Cross-Validation,
- 3rd week: Logistic Regression
- 4th week: Support Vector Machines
- 5th week: Ensemble Models (Adaboost, Random Forests)
- 6th week: Deep Learning (neural networks, backpropagation, SGD)
- 7th week: Deep Learning and applications
- 8th week: Unsupervised learning (K-means, PCA, Sparse Coding)
- 9th week: Probabilistic modelling (hidden variable models, EM)
- 10th week: Introduction to Reinforcement Learning

No rush - stop me whenever something is not clear!

Recap: 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:

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

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

Answer: least squares

Recap: condition for optimum

$$\frac{\partial L(w_0, w_1)}{\partial w_0} = 0 \quad \Leftrightarrow$$

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

$$\frac{\partial L(w_0, w_1)}{\partial w_1} = 0 \quad \Leftrightarrow$$

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

2 linear equations, 2 unknowns

Recap: least squares solution

2x2 system of equations:

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

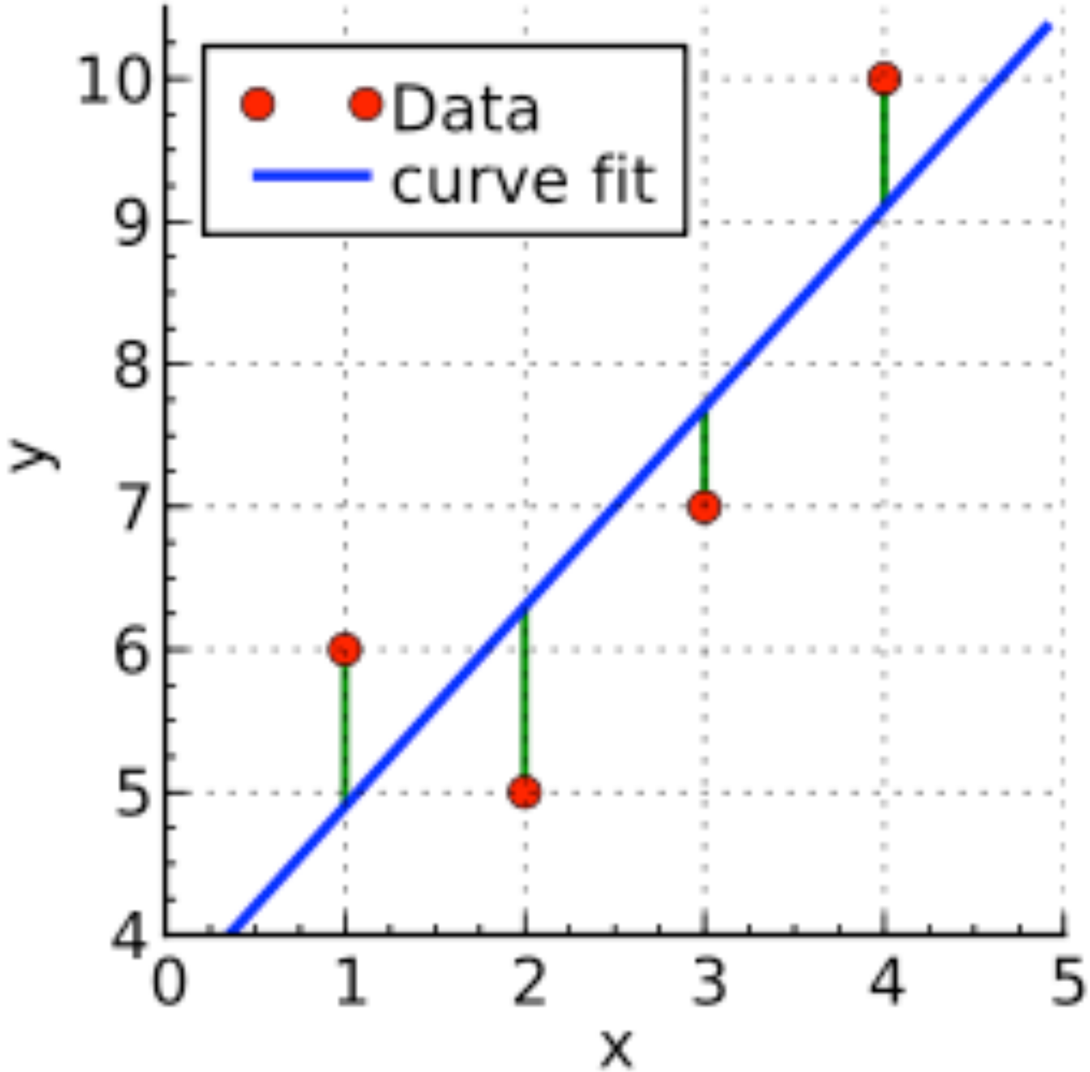
Or, without summations:

$$\mathbf{y} = \begin{bmatrix} y^1 \\ \vdots \\ y^N \end{bmatrix} \quad \mathbf{X} = \begin{bmatrix} x_0^1 & x_1^1 \\ x_0^2 & x_1^2 \\ \vdots & \vdots \\ x_0^N & x_1^N \end{bmatrix}$$

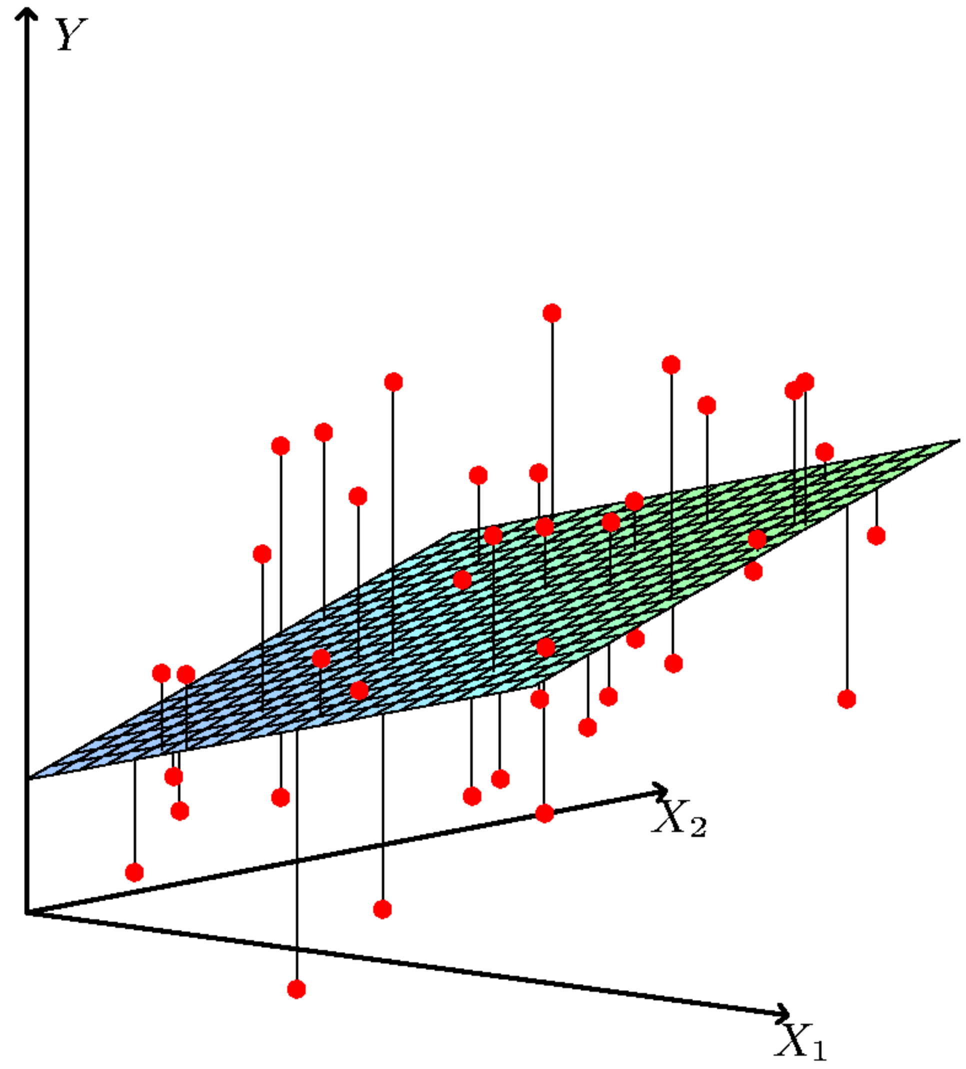
$$\mathbf{X}^T \mathbf{y} = \mathbf{X}^T \mathbf{X} \mathbf{w}$$

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

N: training set size

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

Nx1 NxD Dx1 Nx1

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

Least squares solution for linear regression

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

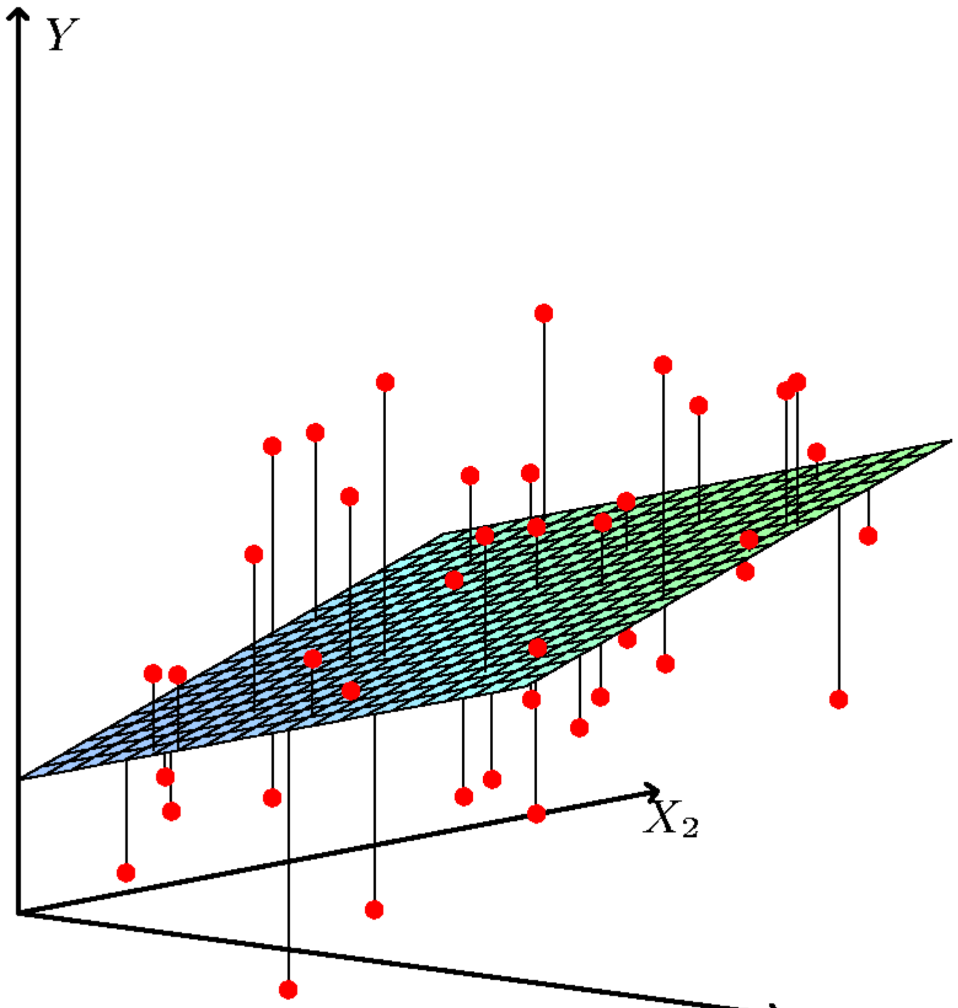
$$L(\mathbf{w}) = \begin{bmatrix} \epsilon^1 & \epsilon^2 & \ldots & \epsilon^N \end{bmatrix} \begin{bmatrix} \epsilon^1 \\ \epsilon^2 \\ \vdots \\ \epsilon^N \end{bmatrix}$$

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

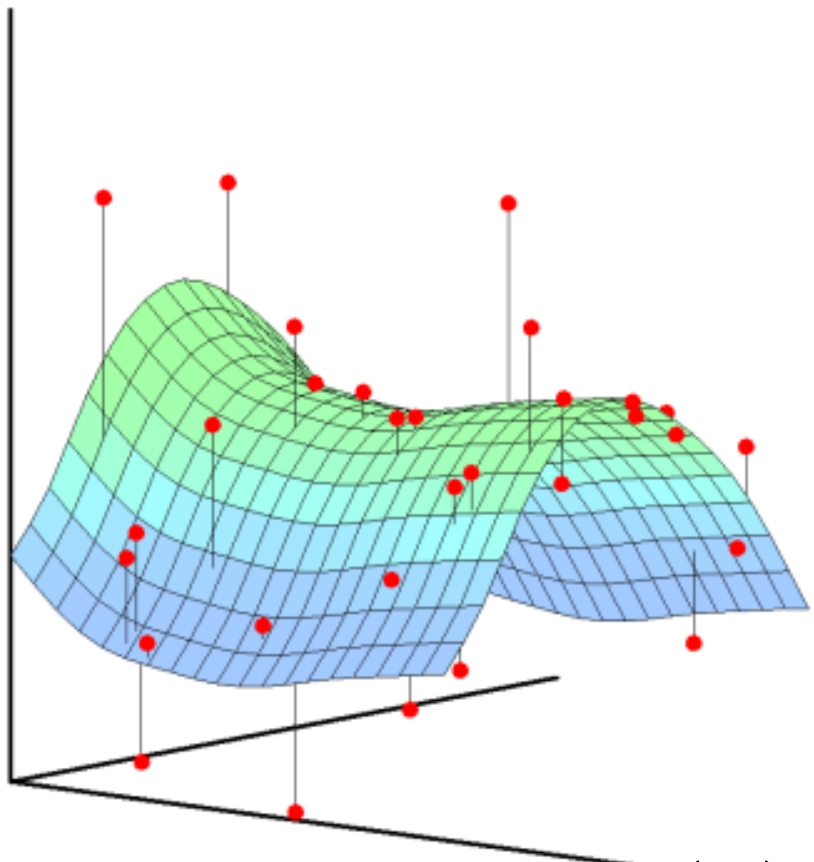
where: $$\mathbf{y} = \mathbf{X}\mathbf{w} + \boldsymbol{\epsilon}$$

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

Linear regression



Generalized linear regression



$$\mathbf{x} \rightarrow \boldsymbol{\phi}(\mathbf{x}) = \left[\begin{array}{c} \phi_1(\mathbf{x}) \\ \vdots \\ \phi_M(\mathbf{x}) \end{array} \right]$$

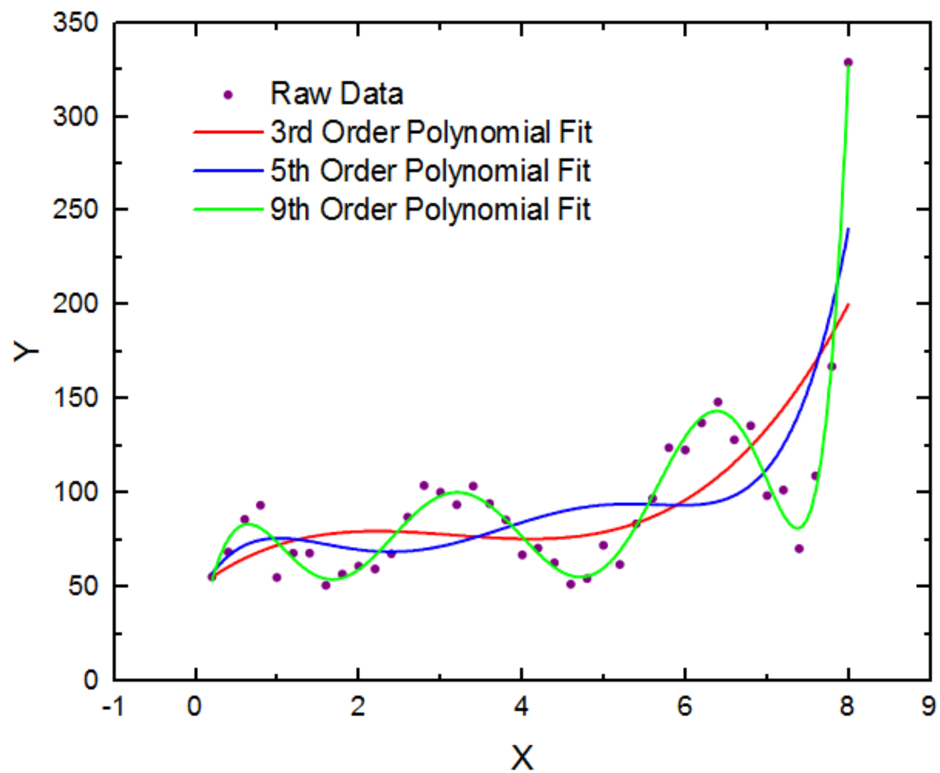
1D Example: 2nd degree polynomial fitting

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

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

1D Example: k-th degree polynomial fitting

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



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

2D example: second-order polynomials

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

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

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

Reminder: linear regression

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

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

Reminder: linear regression

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

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

Generalized linear regression

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

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

Nx1 **NxM** **Mx1** **Nx1**

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

Least squares solution for linear regression

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

$$\mathbf{X} = \begin{bmatrix} (\mathbf{x}^1)^T \\ \hline (\mathbf{x}^2)^T \\ \hline \vdots \\ \hline (\mathbf{x}^N)^T \end{bmatrix}$$

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

Minimize:

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

Least squares solution for generalized linear regression

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

$$\mathbf{\Phi} = \begin{bmatrix} \phi(\mathbf{x}^1)^T \\ \hline \phi(\mathbf{x}^2)^T \\ \hline \vdots \\ \hline \phi(\mathbf{x}^N)^T \end{bmatrix}$$

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

Minimize:

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

2D example: second-order polynomials

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

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

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

5D Example: fourth-order polynomials in 5D

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

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

15625 Dimensions =>15625 parameters

What was happening before: approximations

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

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

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

$$\vdots$$

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

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

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

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

$$\vdots$$

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

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

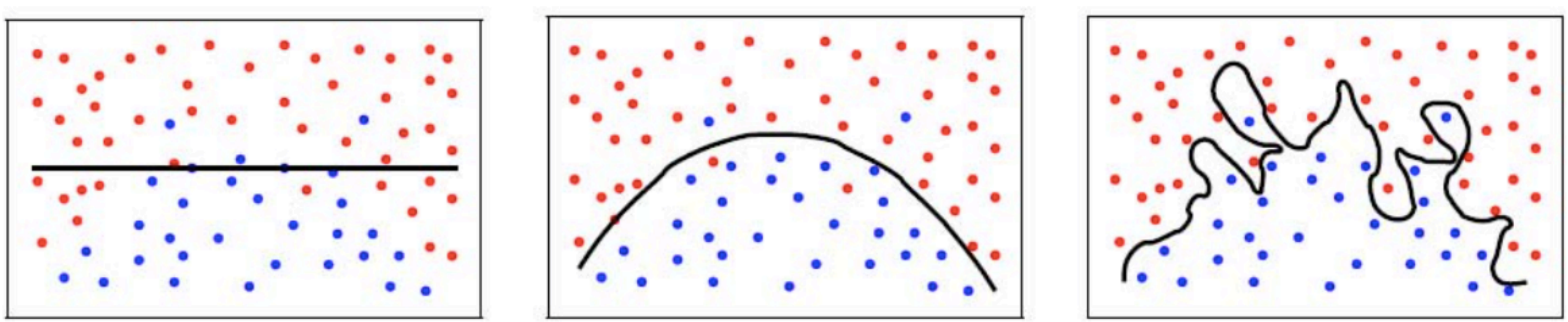
Input-output equations hold exactly, but we are simply memorizing data

Overfitting, in images

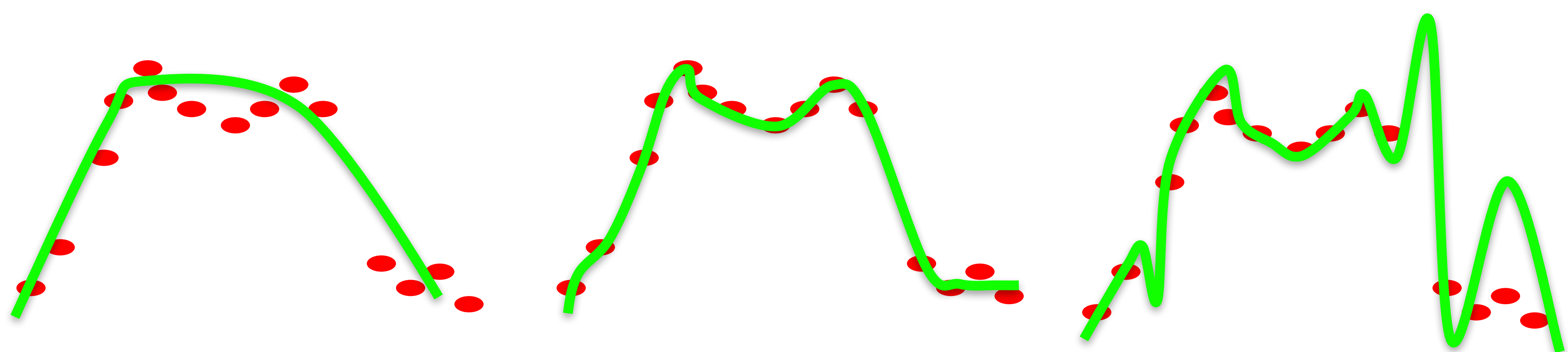
Classification

just right

Underfitting ⟷ Overfitting



Regression



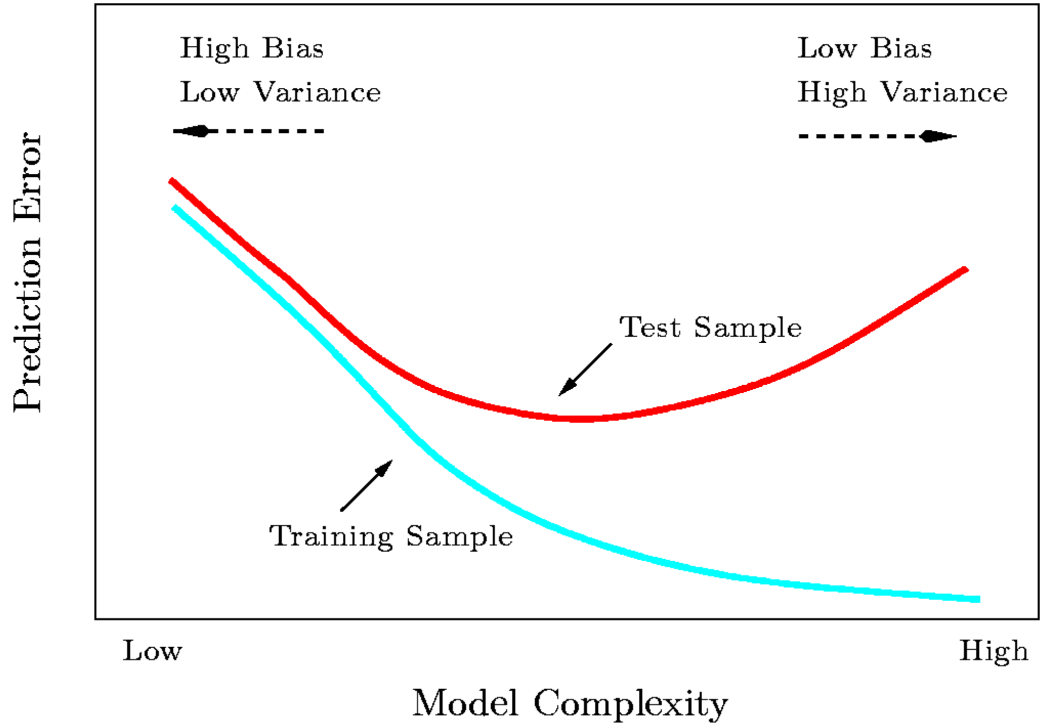
Tuning the model's complexity

A flexible model approximates the target function well in the training set

but can "overtrain" and have poor performance on the test set ("variance")

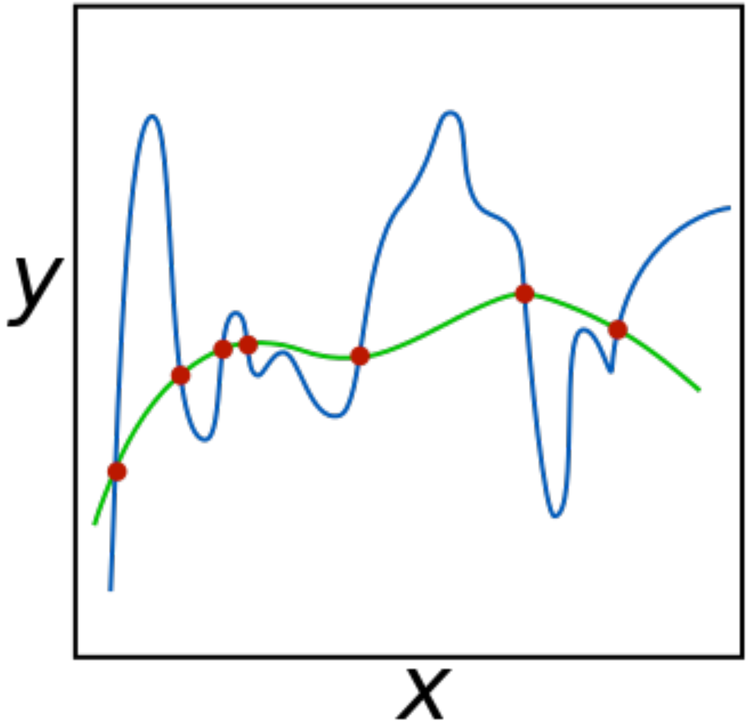
A rigid model's performance is more predictable in the test set

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”

(Week 4: we will guess again!)

Intuition: with many parameters, we can fit anything

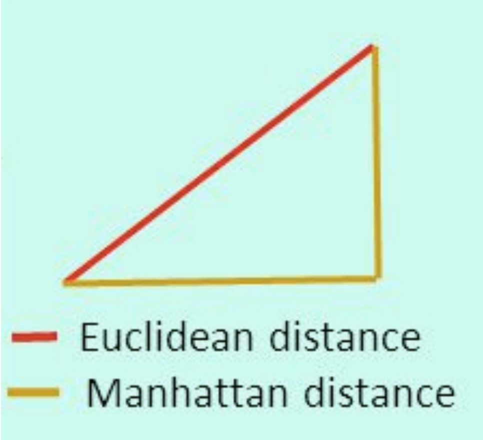
But what if we force the classifier not to use all of the parameters?

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

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

"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

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

residual vector

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

linear regression: minimize model error

Complexity term: (regularizer)

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

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

"data fidelity"

complexity

minimum remains to be determined

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

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

$$= \mathbf{y}^T \mathbf{y} - 2\mathbf{y}^T \mathbf{X}\mathbf{w} + \mathbf{w}^T \mathbf{X}^T \mathbf{X}\mathbf{w} + \lambda \mathbf{w}^T \mathbf{I}\mathbf{w}$$

as before, for linear regression

identity matrix

$$= \mathbf{y}^T \mathbf{y} - 2\mathbf{y}^T \mathbf{X}\mathbf{w} + \mathbf{w}^T \left(\mathbf{X}^T \mathbf{X} + \lambda \mathbf{I}\right) \mathbf{w}$$

Condition for minimum:

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

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

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

Ridge regression, continued

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

New objective:

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

"data fidelity"

complexity

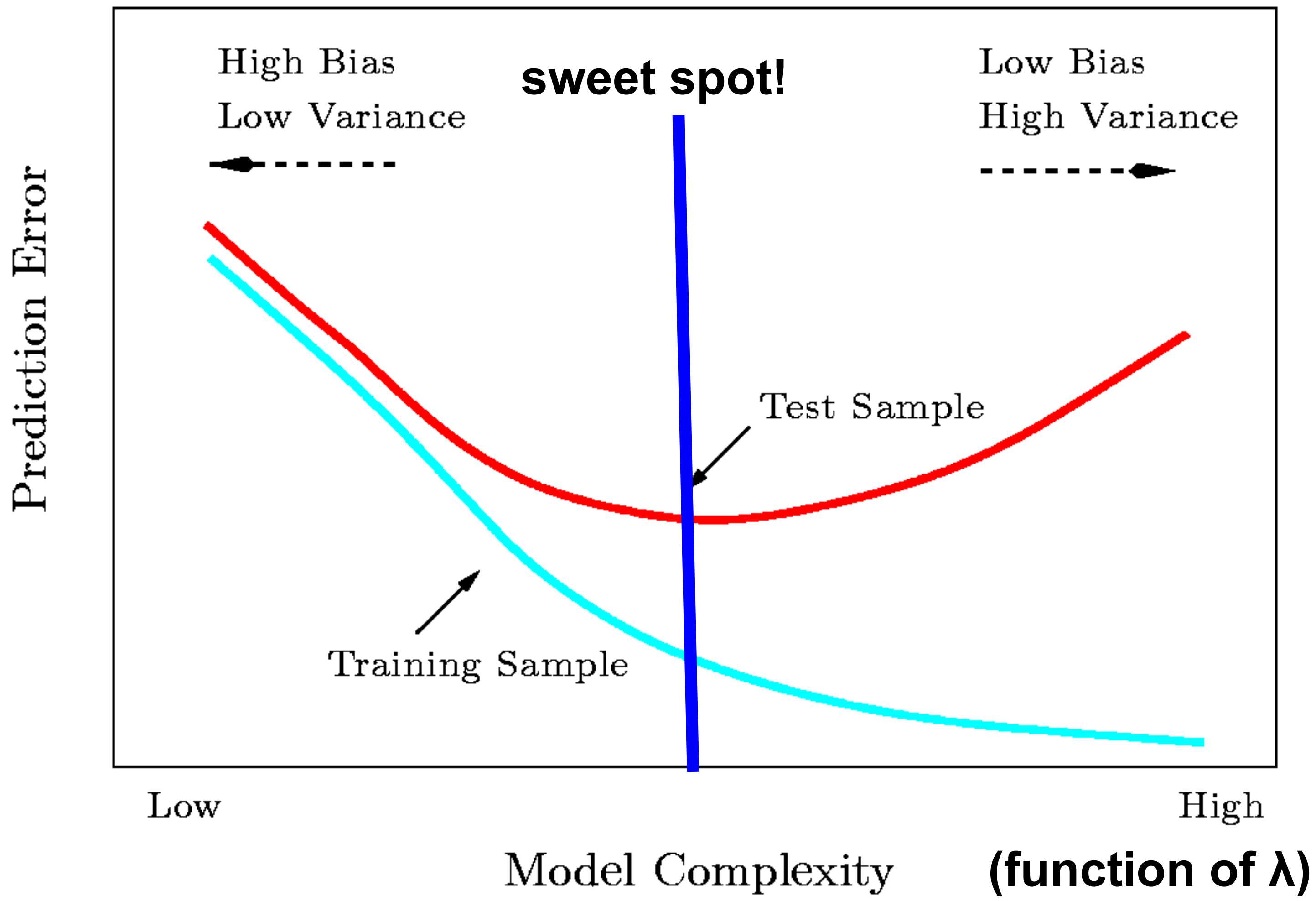
We just determined minimum

scalar, remains to be determined

λ: "hyperparameter"

Nοτε: direct minimization w.r.t. it would lead to λ=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