Happy Wednesday!

- Assignment 2 due tonight, 11:59pm (midnight)
- Assignment 3 out!
- Quiz 7, Friday, Oct 9th 6am until Oct 10th 11:59am (noon)
  - PCA and linear regression

Project… what's next?

- Touch-point 2: deliverables due Mon, Oct 30th, live-event Wed, Nov 2nd
  - Single-slide presentation outlining progress highlights and current challenges
  - Three-minute pre-recorded presentation with your progress and current challenges
- Project midpoint report due Nov 6th 11:59pm (midnight)
  - GitHub page with the results you have achieved utilizing unsupervised learning

Unsupervised learning

- Probability and statistics and information theory
  - Covariance and correlation matrices
  - Entropy and mutual information
- Clustering
  - K-Means
  - DBSCAN
  - Probabilistic (using GMM)
  - Hierarchical
  - Clustering evaluation
- Probability density estimation
  - Parametric (exponential, Bernoulli, Gaussian)
  - Non-parametric (histograms, kernel density estimation)
- Dimensionality reduction
  - Feature selection
  - Principal component analysis

Project midterm report

- Apply unsupervised learning techniques to your data
- Create good visualizations that enable you to understand your data
  - PCA
  - Histograms
  - Confusion matrix
  - Heatmaps
- When appropriate utilize useful metrics to evaluate your work
  - External metrics
  - Internal metrics
- Ask insightful questions of your data
  - "Do these results I get when applying these techniques make sense with what I understand about the problem and the data?"
  - "I have 30 classes in my dataset, but when I use the elbow method with K-means clustering I get 16 clusters instead: what does this mean?"

Pep talk

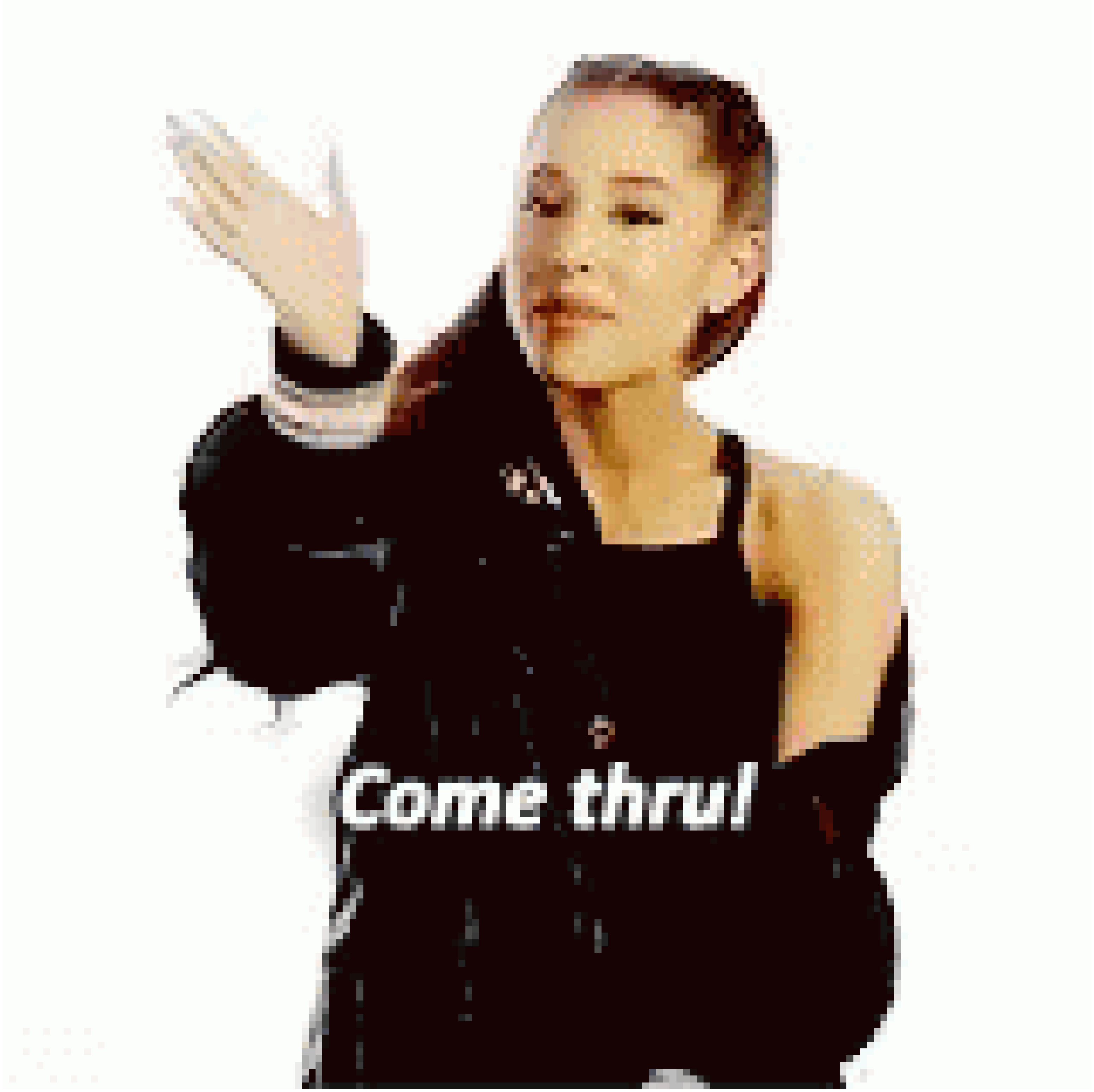


Image credit: Tenor

CS4641B Machine Learning

Lecture 14: Linear regression

Rodrigo Borela ▸ rborelav@gatech.edu

These slides are adapted based on slides from Le Song, Chao Zhang, Yaser Abu-Mostafa, Andrew Zisserman and Mahdi Roozbahani

Outline

- Supervised Learning
- Linear Regression: least squares with normal equations
- Linear Regression: least squares with gradient descent

- *Complementary reading: Bishop PRML – Chapter 1, Section 1.1; Chapter 3, Section 3.1 through 3.2. (Hot tip: check out section Chapter 1, Section 1.2.5)*

Outline

- Supervised Learning
- Linear Regression: least squares with normal equations
- Linear Regression: least squares with gradient descent

Supervised learning: overview

Functions $\mathcal{F}$

$f : \mathcal{X} \to \mathcal{Y}$

Training data

$\{(x_i, y_i) \in \mathcal{X} \times \mathcal{Y}\}$

LEARNING

find $\hat{f} \in \mathcal{F}$
s.t. $y_i \approx \hat{f}(x_i)$

Learning machine

New data

x

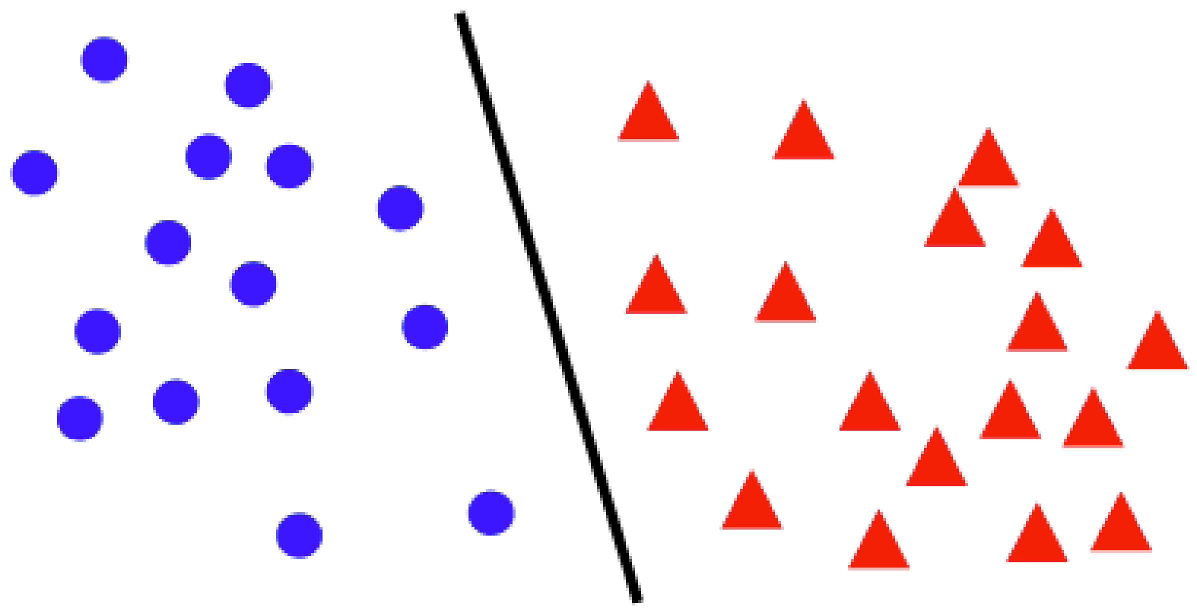
PREDICTION

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

Supervised learning: two types of tasks

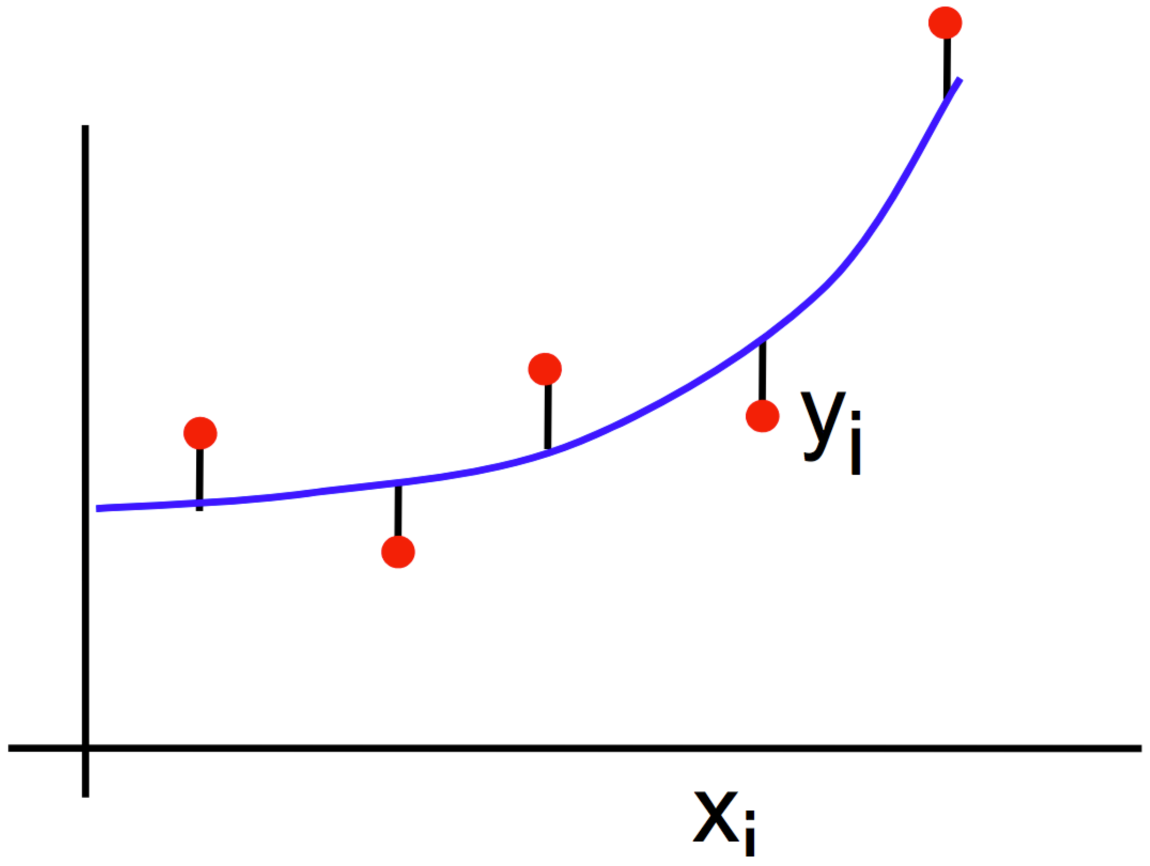
- Given training data: $\{(\mathbf{x}_1, t_1), (\mathbf{x}_2, t_2), \dots, (\mathbf{x}_N, t_N)\}$
- Learn a function: $f(\mathbf{x})$: $y = f(\mathbf{x})$

When y is discrete: **classification**



Class estimation

When y is continuous: **regression**



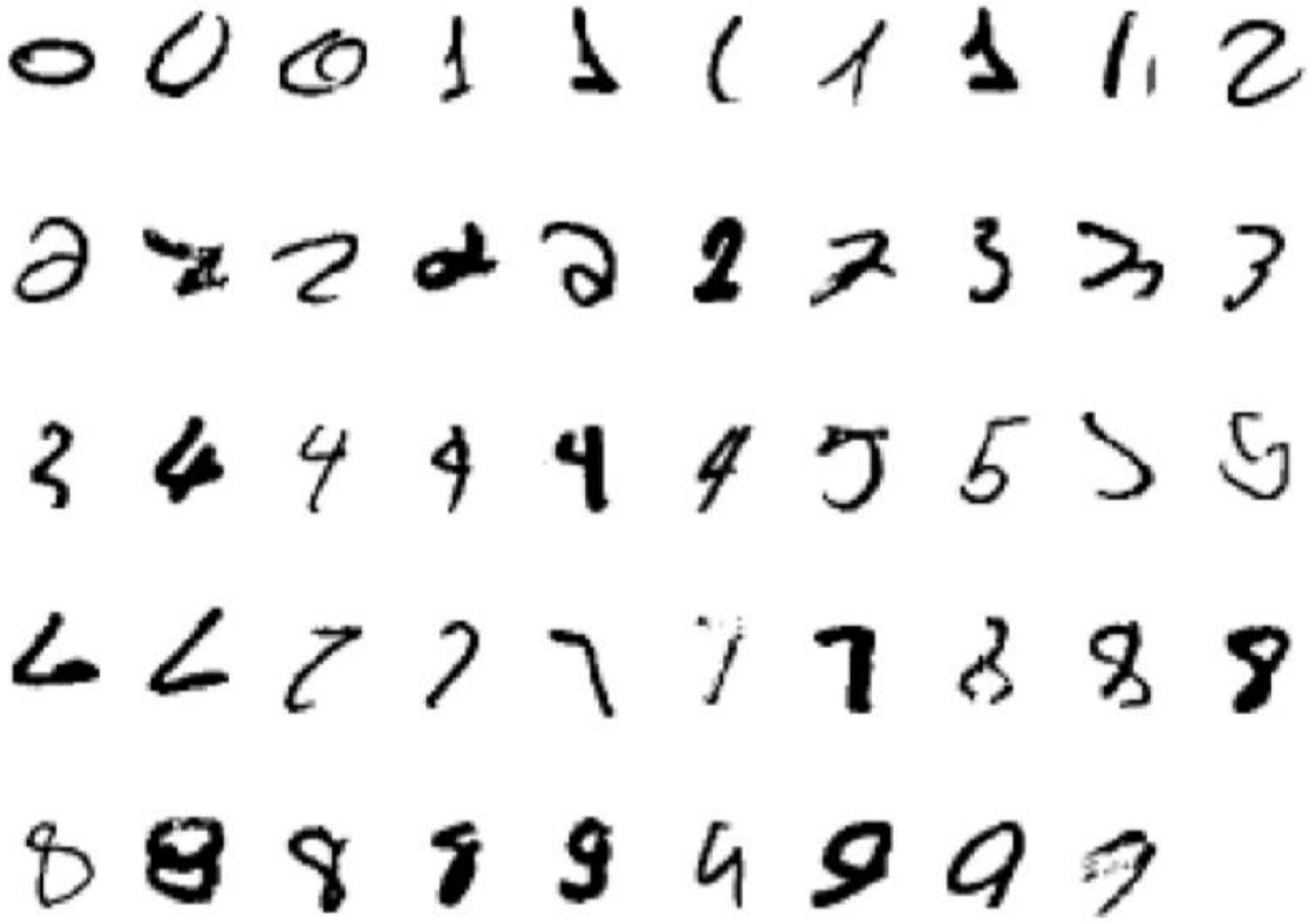
Curve fitting

Unsupervised vs. supervised learning

- Example: clustering vs. classification

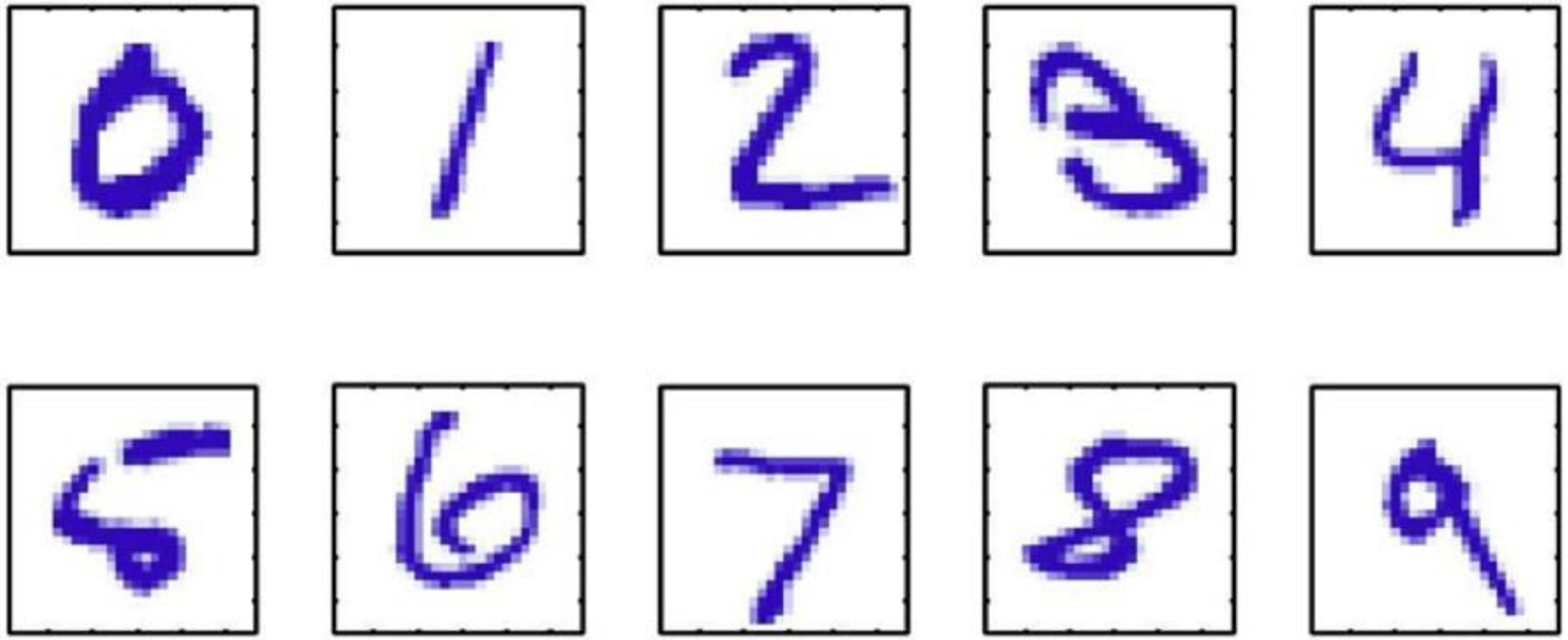
Example (I): handwritten digit recognition

- Start with training data, e.g. 6,000 examples of each digit



- Can achieve testing error of 0.4%
- One of the first commercial and widely used ML systems (for zip codes and checks)

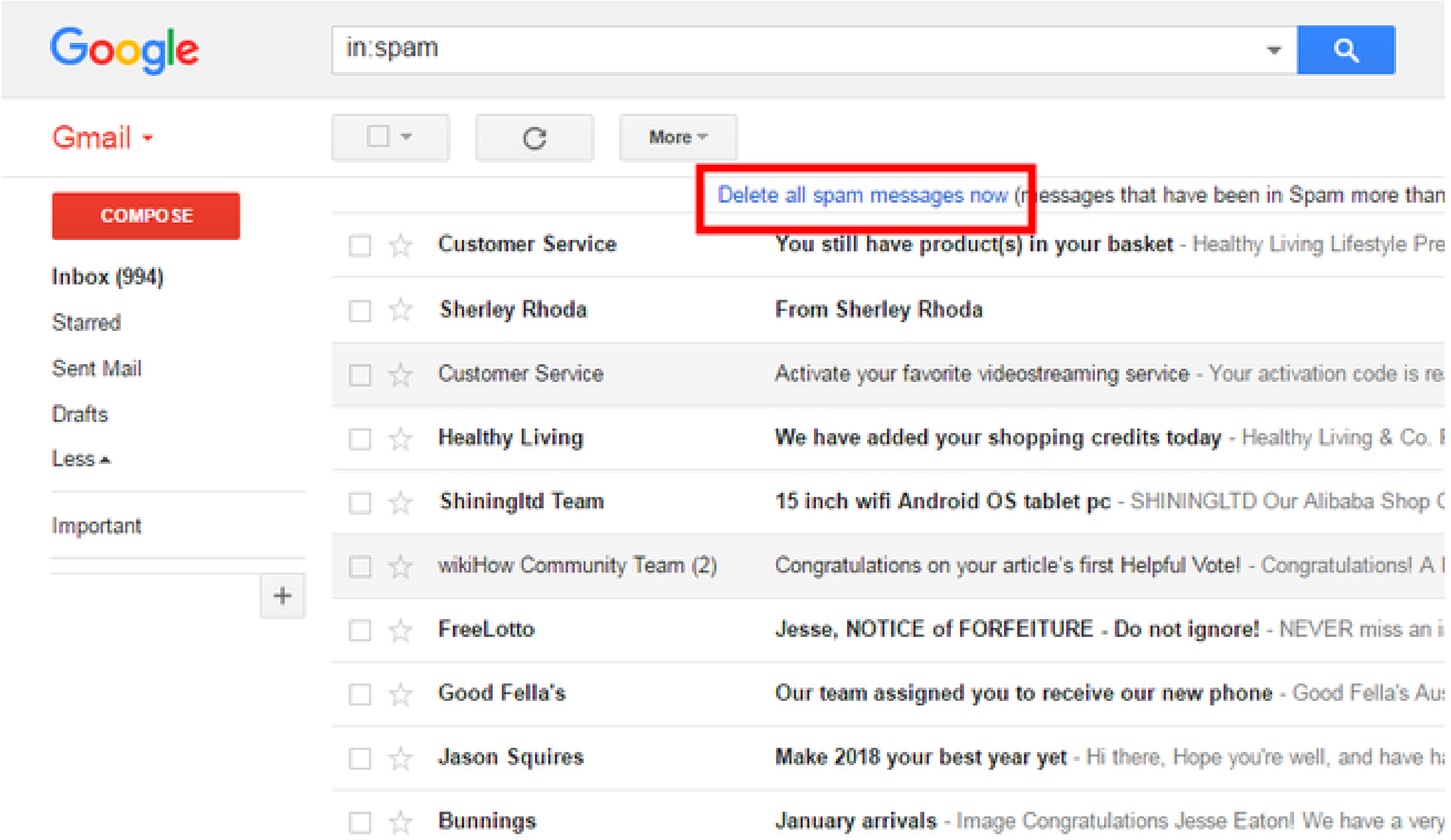
Example (I): handwritten digit recognition

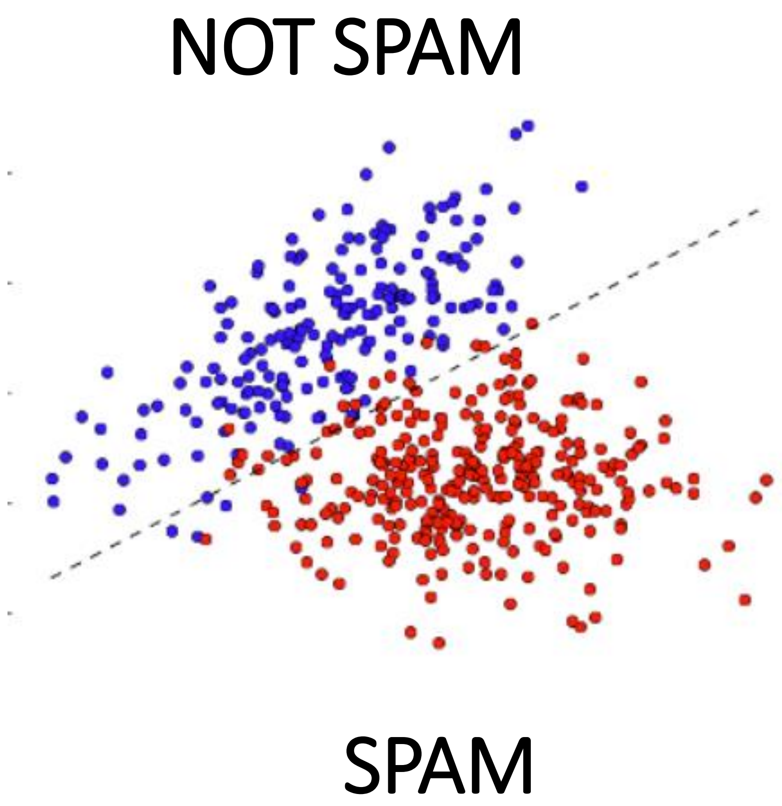


- Images are $28 \times\ 28$ pixels
- Represent input image as a vector $\mathbf{x} \in \mathbb{R}^{784}$
- Learn a classifier $f(\mathbf{x})$ such that,

$$f: \mathbf{x} \rightarrow \{0,1,2,3,4,5,6,7,8,9\}$$

Example (II): spam detection





- Task is to classify email into spam/non-spam
- Data $\mathbf{x}$ bag-of-words vector
- Requires a learning system as "enemy" keeps innovating

Regression example (I): apt. rent prediction

- Suppose you are to move to Atlanta and you want to find the most reasonably priced apartment satisfying your needs:

  square-footage, number of bedrooms, distance to campus

Living area (ft^2)	# bedroom	Rent ($)
230	1	600
506	2	1000
433	2	1100
109	1	500
…		
150	1	?
270	1.5	?

Regression example (I): apt. rent prediction

- **Features**: living area, distance to campus, number of bedrooms
- Denoted as $\mathbf{x} = [x_1, x_2, \dots, x_D]^T$

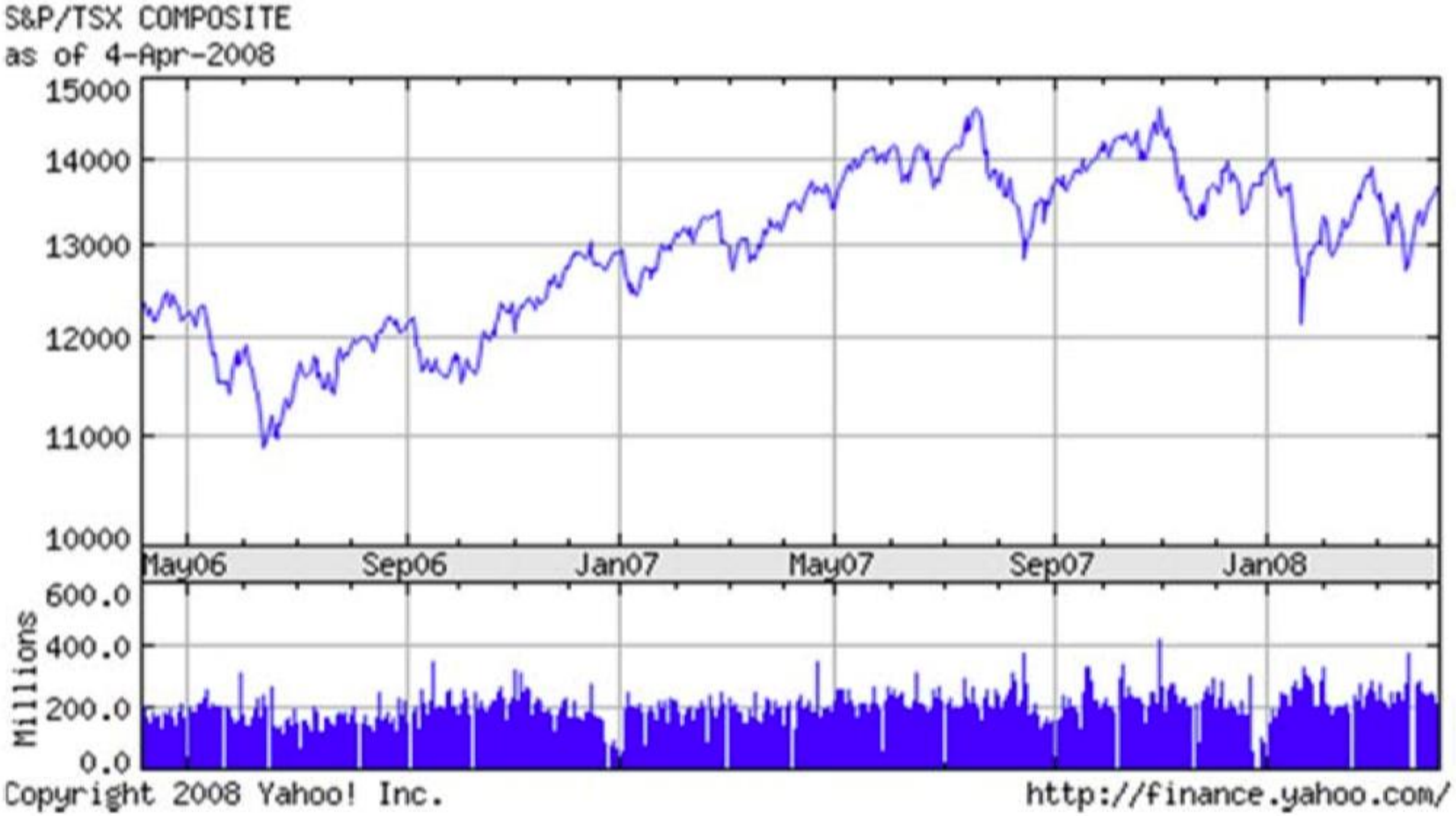
- **Target**: rent
- Denoted as t

- **Training set**:

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}_1^T \\ \mathbf{x}_2^T \\ \dots \\ \mathbf{x}_N^T \end{bmatrix} \in \mathbb{R}^{N \times D} \text{ and } \mathbf{t} = \begin{bmatrix} t_1 \\ t_2 \\ \dots \\ t_N \end{bmatrix} \in \mathbb{R}^N$$

Regression example (II): stock price prediction

- Task is to predict stock price at future date



Outline

- Supervised Learning
- Linear Regression: least squares with normal equations
- Linear Regression: least squares with gradient descent

Linear regression

- Assume y is a linear function of x (features)

$$y(\mathbf{x}, \mathbf{w}) = w_0 + w_1 x_1 + \cdots + w_D x_D$$

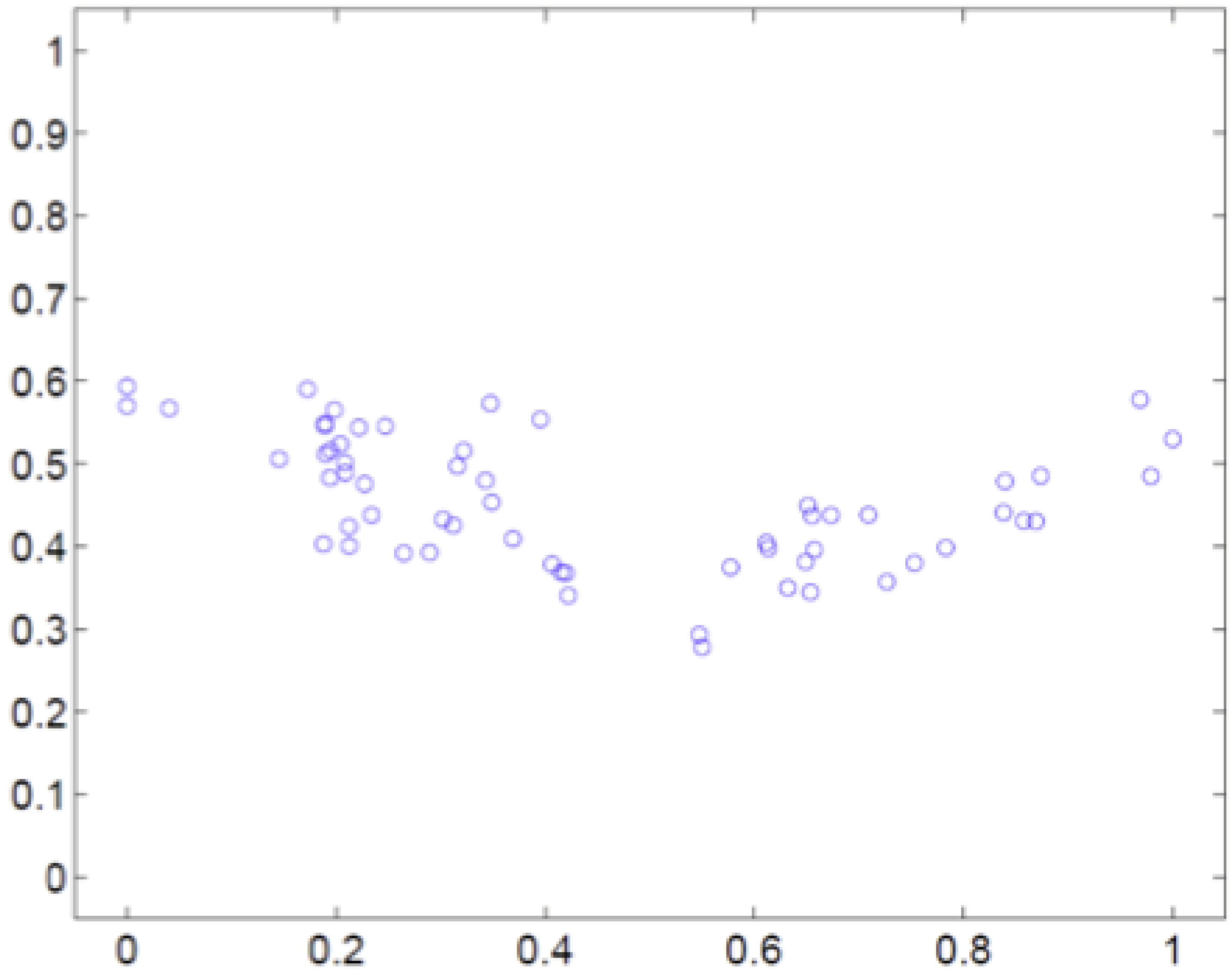
- We can extend these using a basis function $\phi_m(\mathbf{x})$:

$$y(\mathbf{x}, \mathbf{w}) = w_0 + \sum_{m=1}^{M-1} w_m \phi_m(\mathbf{x})$$

Or if we pick $\phi_0(\mathbf{x}) = 1$

$$y(\mathbf{x}, \mathbf{w}) = \sum_{m=0}^{M-1} w_m \phi_m(\mathbf{x}) \; = \mathbf{w}^{\boldsymbol{T}} \boldsymbol{\phi}(\mathbf{x})$$

What are these basis functions?

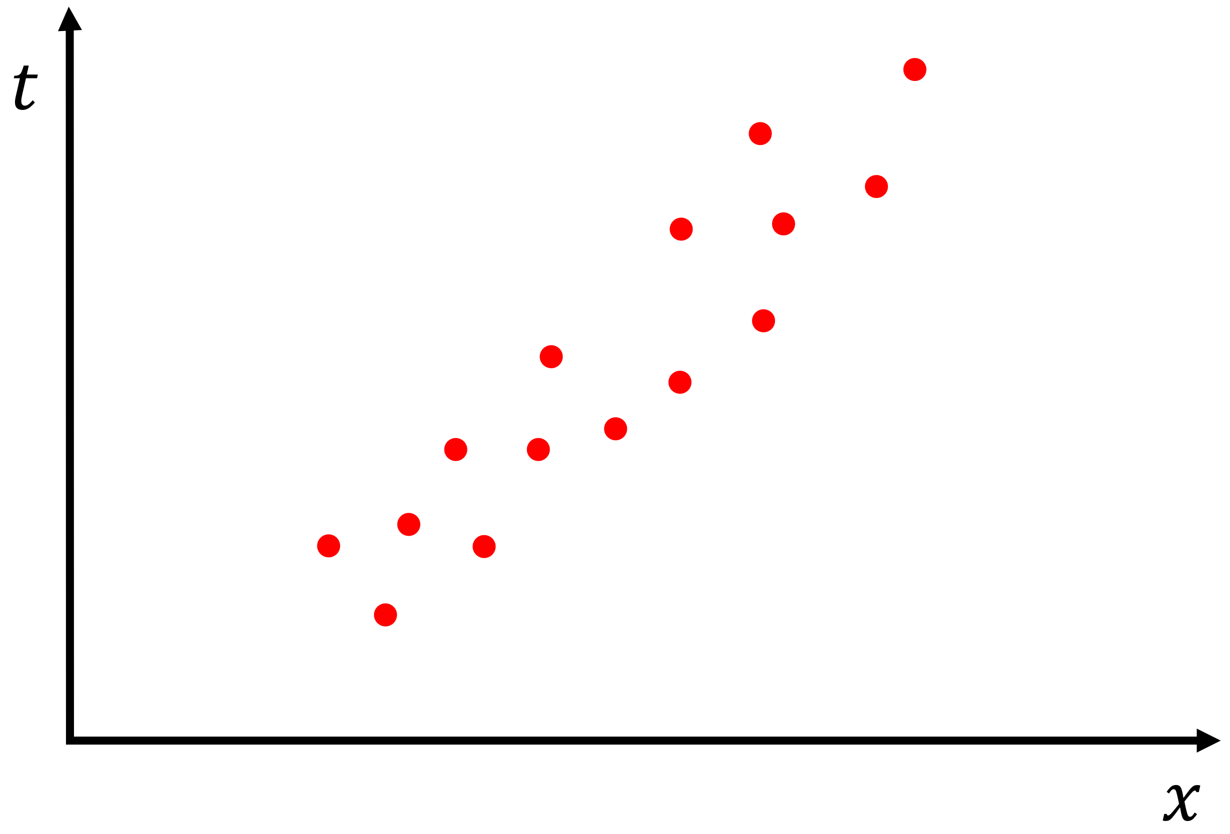


Linear regression

- We assume that the target variable t is given by the sum of the deterministic function $y(\mathbf{x}, \mathbf{w})$ and a random noise ϵ

$$t = y(\mathbf{x}, \mathbf{w}) + \epsilon$$

- Our objective is to find the $\mathbf{w}$ that minimizes the difference between the target and predicted values. What would be a good objective function?



Least squares method

- Given N datapoints, find $\mathbf{w}$ that minimizes the sum-of-squares:

$$L(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}\left(t_n - \mathbf{w}^{\boldsymbol{T}}\boldsymbol{\phi}(\mathbf{x}_n)\right)^2$$

- Good old trick: set the gradient of the objective function wrt $\mathbf{w}$ to zero:

$$\frac{\partial L(\mathbf{w})}{\partial \mathbf{w}} = \sum_{n=1}^{N}\left(t_n - \mathbf{w}^{\boldsymbol{T}}\boldsymbol{\phi}(\mathbf{x}_n)\right)\boldsymbol{\phi}(\mathbf{x}_n)^T$$

$$0 = \sum_{n=1}^{N}(t_n\boldsymbol{\phi}(\mathbf{x}_n)^T) - \mathbf{w}^{\boldsymbol{T}}\left(\sum_{n=1}^{N}\boldsymbol{\phi}(\mathbf{x}_n)\boldsymbol{\phi}(\mathbf{x}_n)^T\right)$$

$$\mathbf{w} = (\boldsymbol{\Phi}^T\boldsymbol{\Phi})^{-1}\boldsymbol{\Phi}^T\mathbf{t}$$ (Normal equations)

$$\boldsymbol{\Phi}^{+} = (\boldsymbol{\Phi}^T\boldsymbol{\Phi})^{-1}\boldsymbol{\Phi}^T$$ (Moore-Penrose inverse)

Least squares method

- The design matrix:

$$\mathbf{\Phi} = \begin{pmatrix} \phi_0(\mathbf{x}_1) & \phi_1(\mathbf{x}_1) & \cdots & \phi_{M-1}(\mathbf{x}_1) \\ \phi_0(\mathbf{x}_2) & \phi_1(\mathbf{x}_2) & \cdots & \phi_{M-1}(\mathbf{x}_2) \\ \vdots & \vdots & \ddots & \vdots \\ \phi_0(\mathbf{x}_N) & \phi_1(\mathbf{x}_N) & \cdots & \phi_{M-1}(\mathbf{x}_N) \end{pmatrix}_{N \times M}$$

- If our basis function $\boldsymbol{\phi}(\mathbf{x}_n)$ just maps the features with a leading 1, the design matrix becomes:

$$\mathbf{\Phi} = \begin{pmatrix} 1 & x_{11} & \cdots & x_{1D} \\ 1 & x_{21} & \cdots & x_{2D} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{N1} & \cdots & x_{ND} \end{pmatrix}_{N \times (D+1)}$$

- If our basis function is a polynomial $\phi_m(x) = x^m$, of degree $M - 1$ and a data point $\mathbf{x}_n = x_n$ (scalar), the design matrix becomes:

$$\mathbf{\Phi} = \begin{pmatrix} 1 & x_1 & \cdots & x_1^{M-1} \\ 1 & x_2 & \cdots & x_2^{M-1} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_N & \cdots & x_N^{M-1} \end{pmatrix}_{N \times M}$$

Least squares method

- Example: $\mathbf{X} = \begin{bmatrix} 3 & 1{,}500 & 4 \\ 5 & 2{,}830 & 8 \\ 4 & 2{,}420 & 6 \\ 3 & 1{,}870 & 4 \end{bmatrix}$ with simple mapping $\boldsymbol{\phi}(\mathbf{x}_n)$ with a leading 1:

$$\boldsymbol{\Phi} = \begin{pmatrix} 1 & 3 & 1{,}500 & 4 \\ 1 & 5 & 2{,}830 & 8 \\ 1 & 4 & 2{,}420 & 6 \\ 1 & 3 & 1{,}870 & 4 \end{pmatrix}_{N \times (D+1)}$$

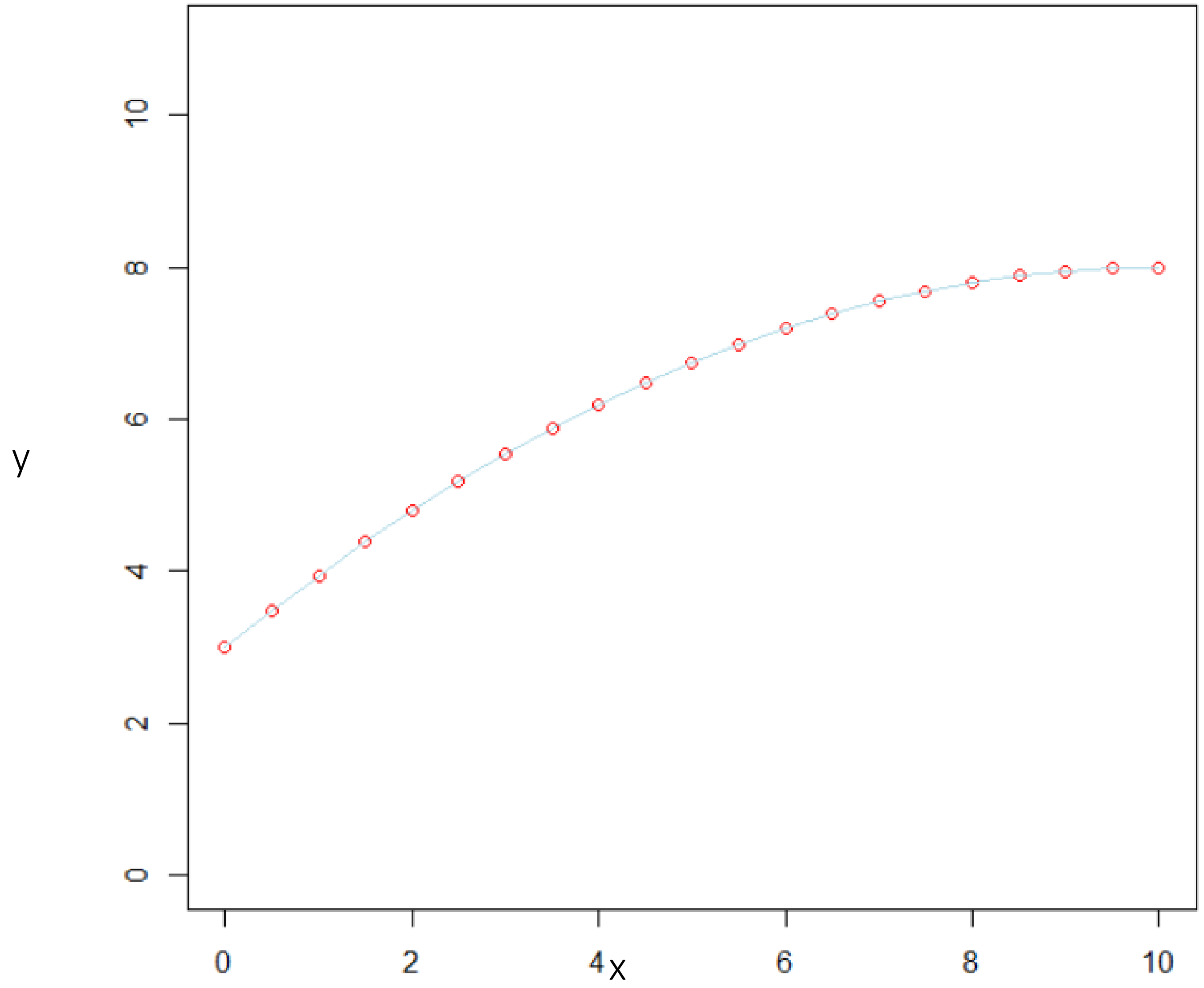
- Example: $\mathbf{X} = \begin{bmatrix} 1{,}500 \\ 2{,}830 \\ 2{,}420 \\ 1{,}870 \end{bmatrix}$ with polynomial $\phi_m(x) = x^m$, of degree $M - 1 = 3$:

$$\boldsymbol{\Phi} = \begin{pmatrix} 1 & 1{,}500 & 1{,}500^2 & 1{,}500^3 \\ 1 & 2{,}830 & 2{,}830^2 & 2{,}830^3 \\ 1 & 2{,}420 & 2{,}420^2 & 2{,}420^3 \\ 1 & 1{,}870 & 1{,}870^2 & 1{,}870^3 \end{pmatrix}_{N \times M}$$

What is happening in polynomial regression?

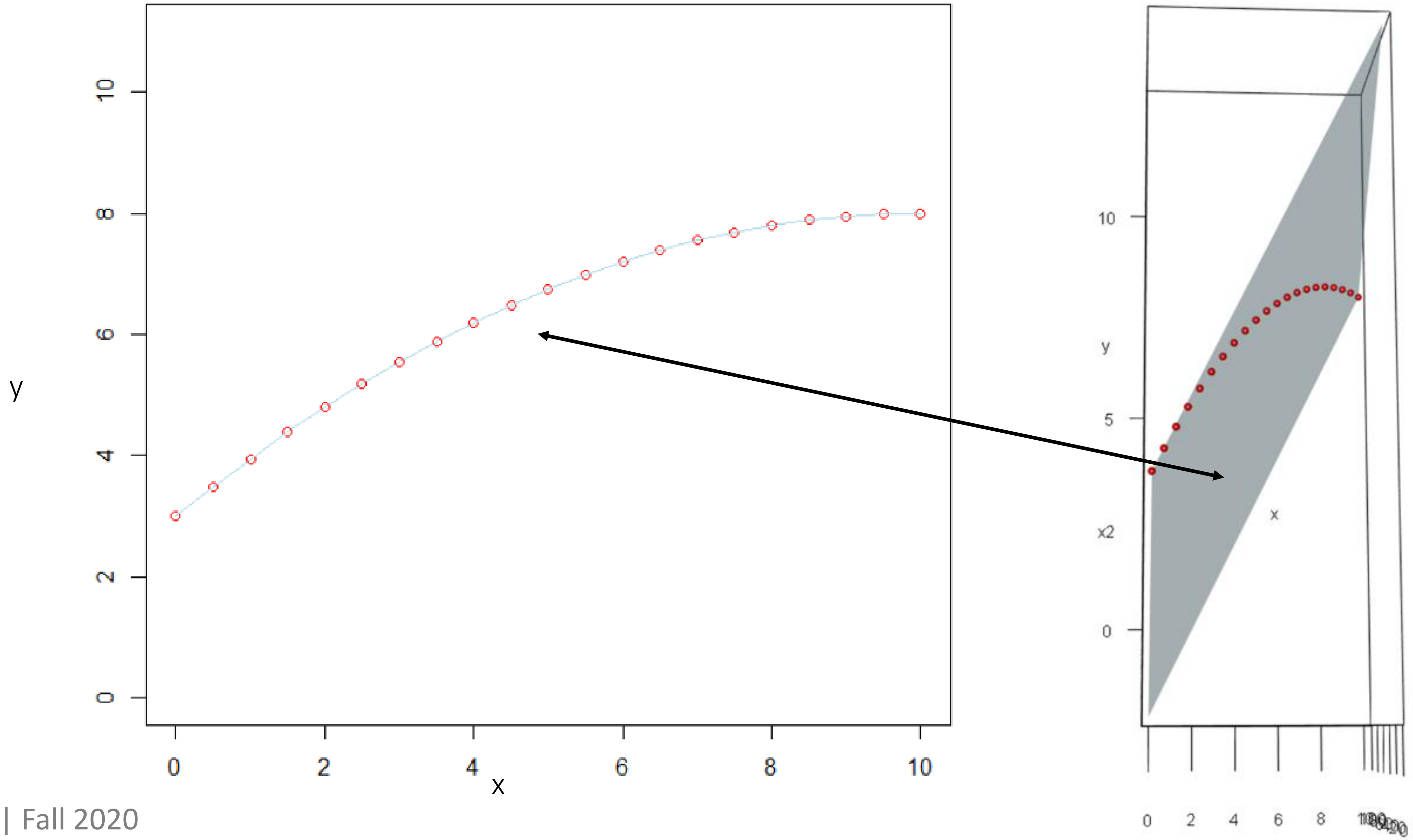
$$\mathbf{X} = [0, 0.5, 1, \ldots, 9.5, 10]^T$$
$$\boldsymbol{t} = [3, 3.4875, 3.95, \ldots, 7.98, 8]^T$$

$$y(x, w) = w_0 + w_1 x + w_2 x^2$$
$$w_0 = 3; w_1 = 1; w_2 = -0.5$$

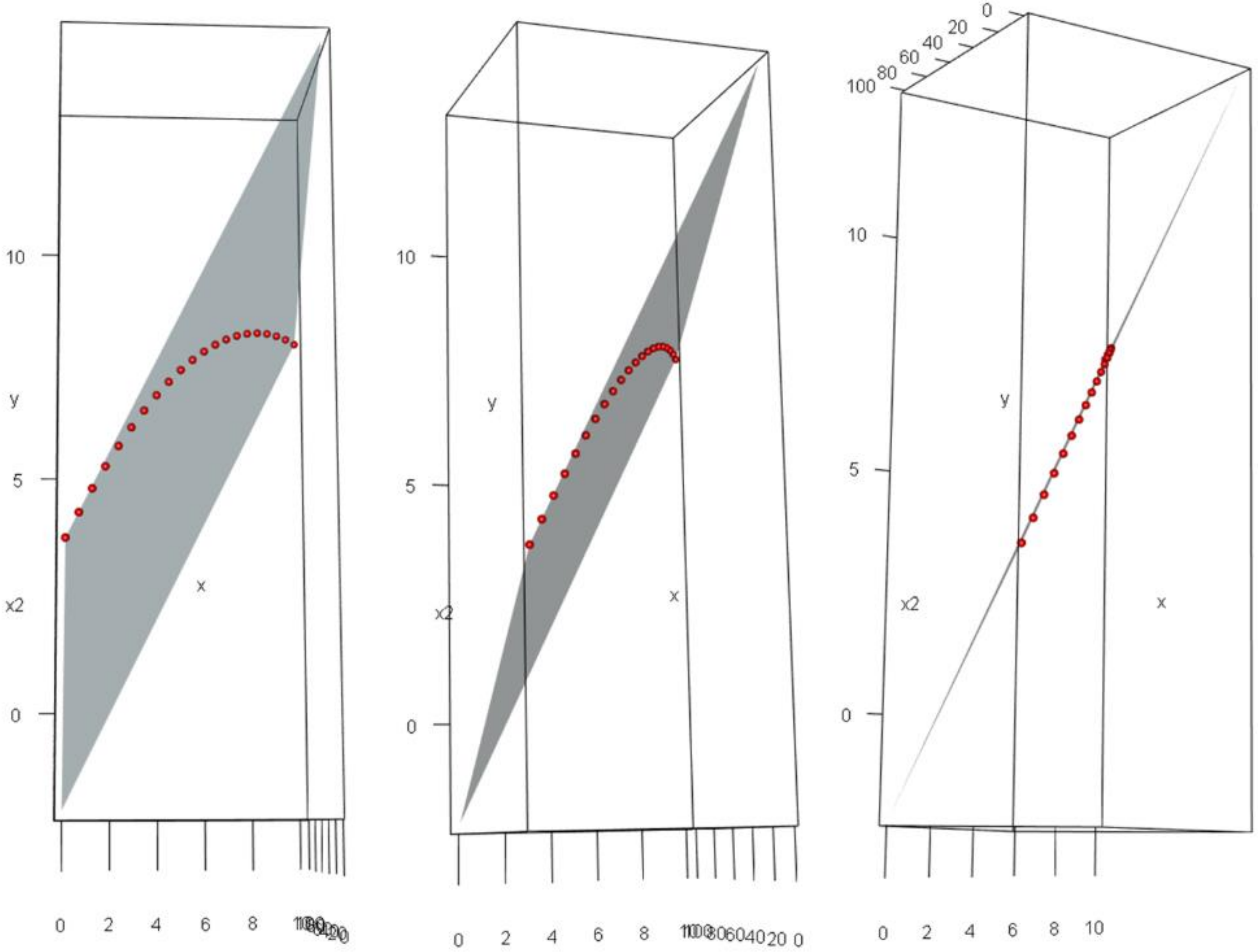


Adding to the feature space

- We are fitting a D-dimensional hyperplane in a $D + 1$ dimensional hyperspace (in this example a 2D plane in a 3D space). That hyperplane really is “flat” / “linear” in 3D. It can be seen a non-linear regression (a curvy line) in our 2D example in fact it is a flat surface in 3D.
- So the fact that it is mentioned that the model is linear in parameters, it is shown here



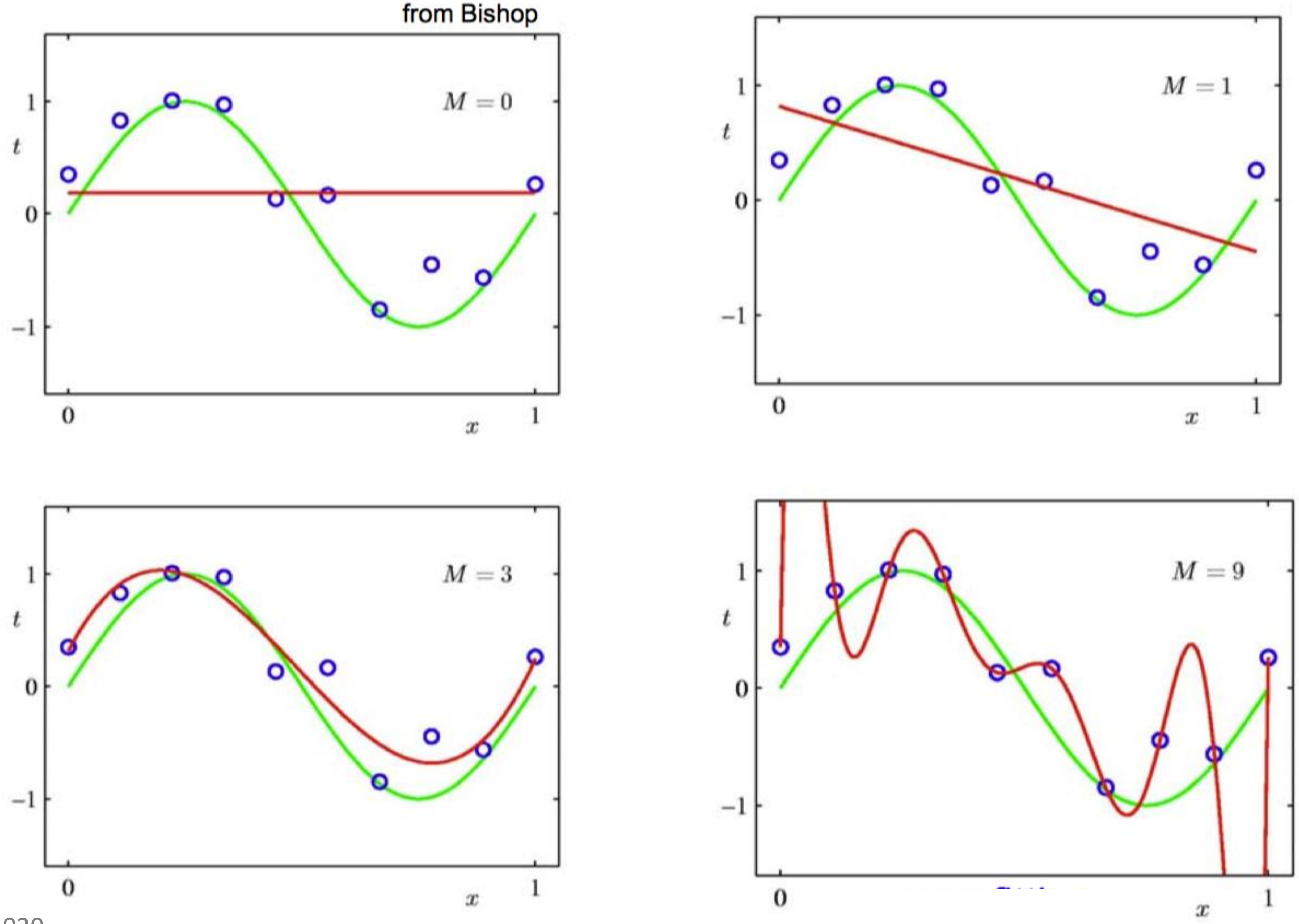
Adding to the feature space



$$\mathbf{\Phi} = \begin{pmatrix} 1 & 0 & 0 \\ 1 & 0.5 & 0.25 \\ \vdots & \vdots & \vdots \\ 1 & 10 & 100 \end{pmatrix}_{N\times\mathbf{3}}$$

$$\boldsymbol{y} = \begin{bmatrix} 3.0 \\ 3.4875 \\ \vdots \\ 8 \end{bmatrix}_{N\times 1}$$

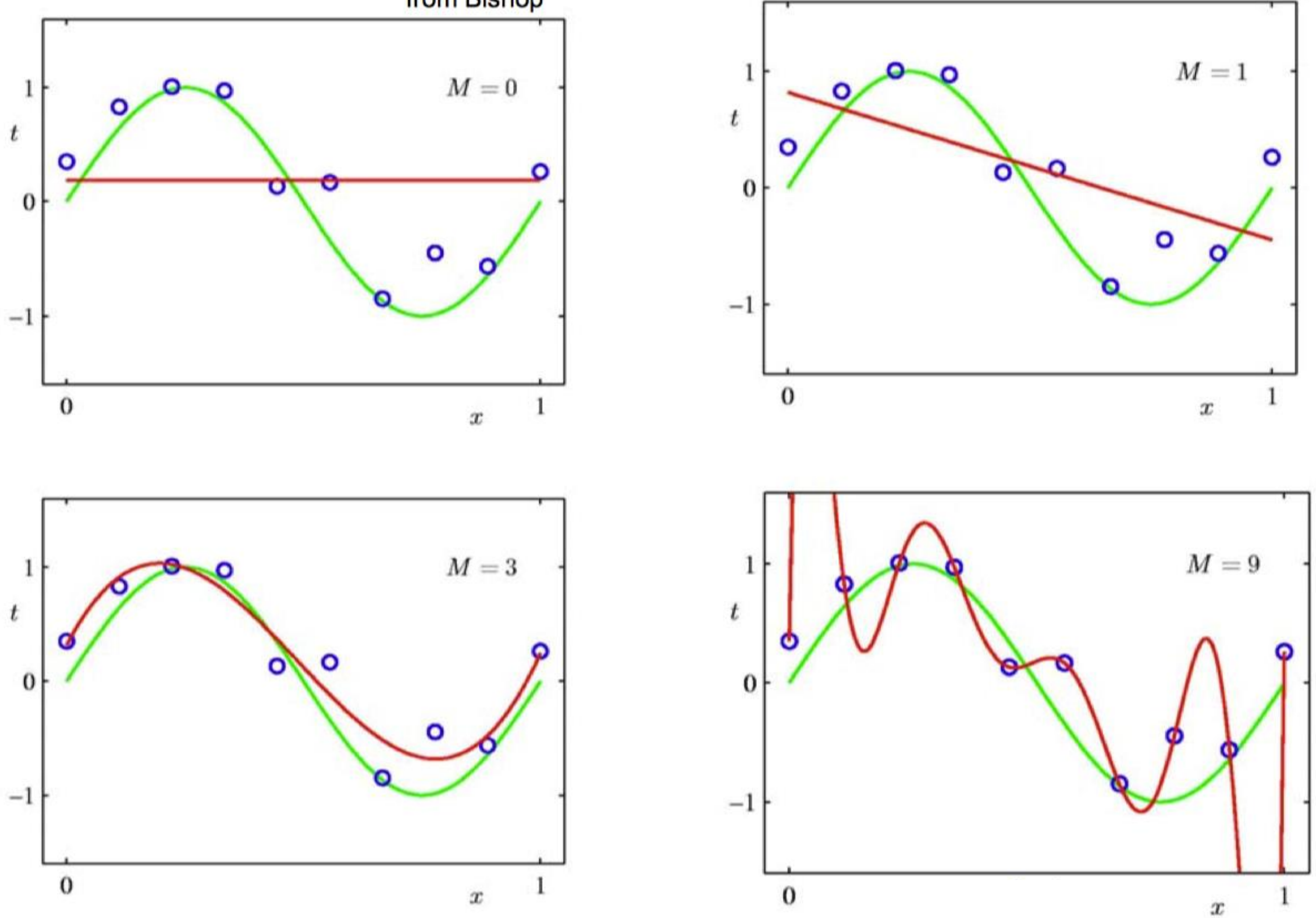
Increasing the polynomial degree



Which one is better?

- Can we increase the maximal polynomial degree such that the curve passes through all training points?

from Bishop



Least squares method

- Let us assume that our basis function is simply mapping the points in the vector $\mathbf{x}_n$ with a leading 1:

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

Dataset: $\mathbf{X}_{N\times D}$ D = dimension, N = datapoints (instances)

$$\mathbf{\Phi}^T\mathbf{\Phi} = \begin{bmatrix} (D+1)\times N \end{bmatrix}\begin{bmatrix} N\times(D+1) \end{bmatrix} = \begin{bmatrix} (D+1)\times(D+1) \end{bmatrix}$$

Not a big matrix because $N \gg D$, this matrix is invertible most of the times. If we are VERY unlucky and columns of $\mathbf{\Phi}^T\mathbf{\Phi}$ are not linearly independent (it's not a full rank matrix), then it is not invertible.

Solving normal equations

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

- Pros: a single-shot algorithm! Easiest to implement.
- Cons: need to compute inverse $(\mathbf{\Phi}^T\mathbf{\Phi})^{-1}$, expensive, numerical issues (e.g. matrix could be singular, etc.)

Outline

- Supervised Learning
- Linear Regression: least squares with normal equations
- Linear Regression: least squares with gradient descent

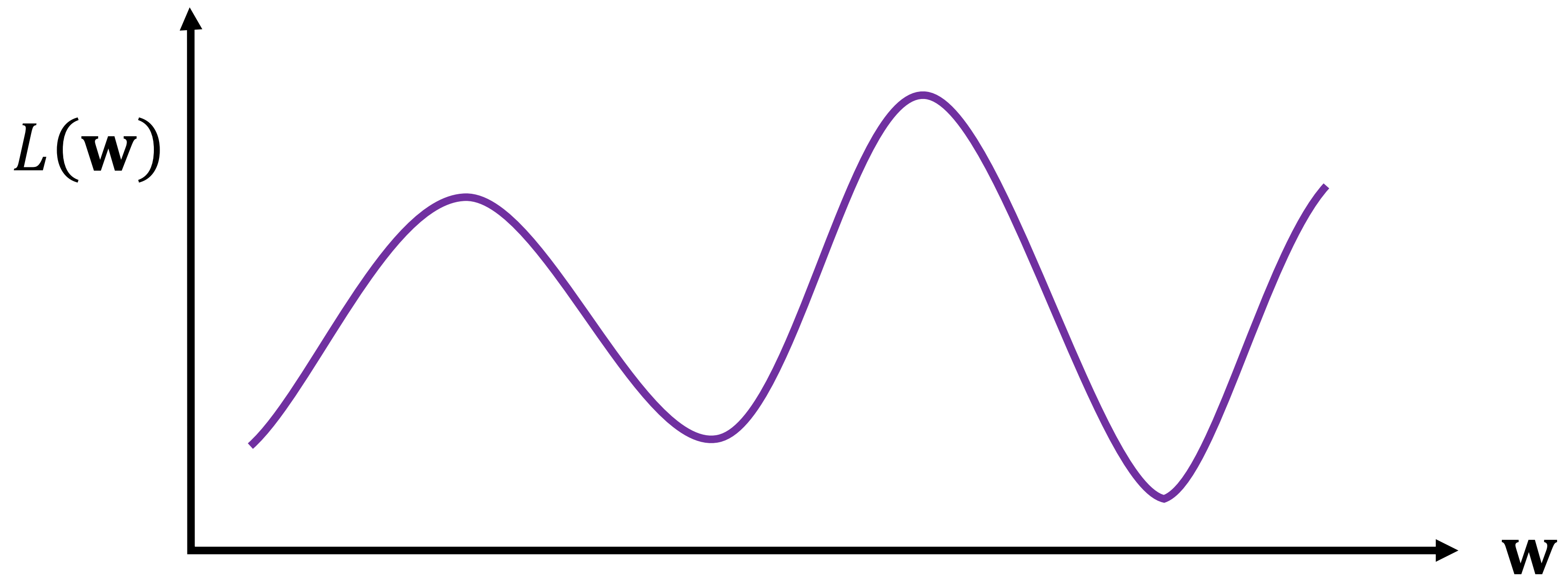
Alternative methods for optimization

- The matrix inversion in $\mathbf{w} = (\boldsymbol{\Phi}^T\boldsymbol{\Phi})^{-1}\boldsymbol{\Phi}^T\mathbf{t}$ can be very expensive to compute. Let's consider the mean of the sum-of-squares error:

$$L(\mathbf{w}) = \frac{1}{N}\sum_{n=1}^{N}\left(t_n - \mathbf{w}^T\boldsymbol{\phi}(\mathbf{x}_n)\right)^2$$

- Calculating the derivative wrt $\mathbf{w}$, we obtain:

$$\frac{\partial L(\mathbf{w})}{\partial \mathbf{w}} = \frac{1}{N}\sum_{n=1}^{N}\left(t_n - \mathbf{w}^T\boldsymbol{\phi}(\mathbf{x}_n)\right)\boldsymbol{\phi}(\mathbf{x}_n)^T$$



Methods for optimization

- Gradient descent

$$\mathbf{w}_{(\tau+1)} = \mathbf{w}_{(\tau)} - \alpha \frac{\partial L(\mathbf{w})}{\partial \mathbf{w}} \rightarrow \mathbf{w}_{(\tau+1)} = \mathbf{w}_{(\tau)} - \frac{\alpha}{N} \sum_{n=1}^{N} \left(t_n - \mathbf{w}_{(\tau)}{}^T \boldsymbol{\phi}(\mathbf{x}_n) \right) \boldsymbol{\phi}(\mathbf{x}_n)^T$$

  - Pros: fast-converging, easy to implement
  - Cons: need to read all data

- Stochastic gradient descent

$$\mathbf{w}_{(\tau+1)} = \mathbf{w}_{(\tau)} - \beta \frac{\partial L(\mathbf{w})}{\partial \mathbf{w}} \rightarrow \mathbf{w}_{(\tau+1)} = \mathbf{w}_{(\tau)} - \beta \left(t_n - \mathbf{w}_{(\tau)}{}^T \boldsymbol{\phi}(\mathbf{x}_n) \right) \boldsymbol{\phi}(\mathbf{x}_n)^T$$

  - Pros: online, low per-step cost
  - Cons: maybe slow-converging

Stochastic gradient descent: example