Lecutre 2: ML Basics from Representation Learning view

Today:

00:00) Supervised learning formalized

00:10) Linear regression & normal equations

00:20) Gradient descent

00:45) ---- Break ----

01:00) Tensor operations

01:45) ---- Break ----

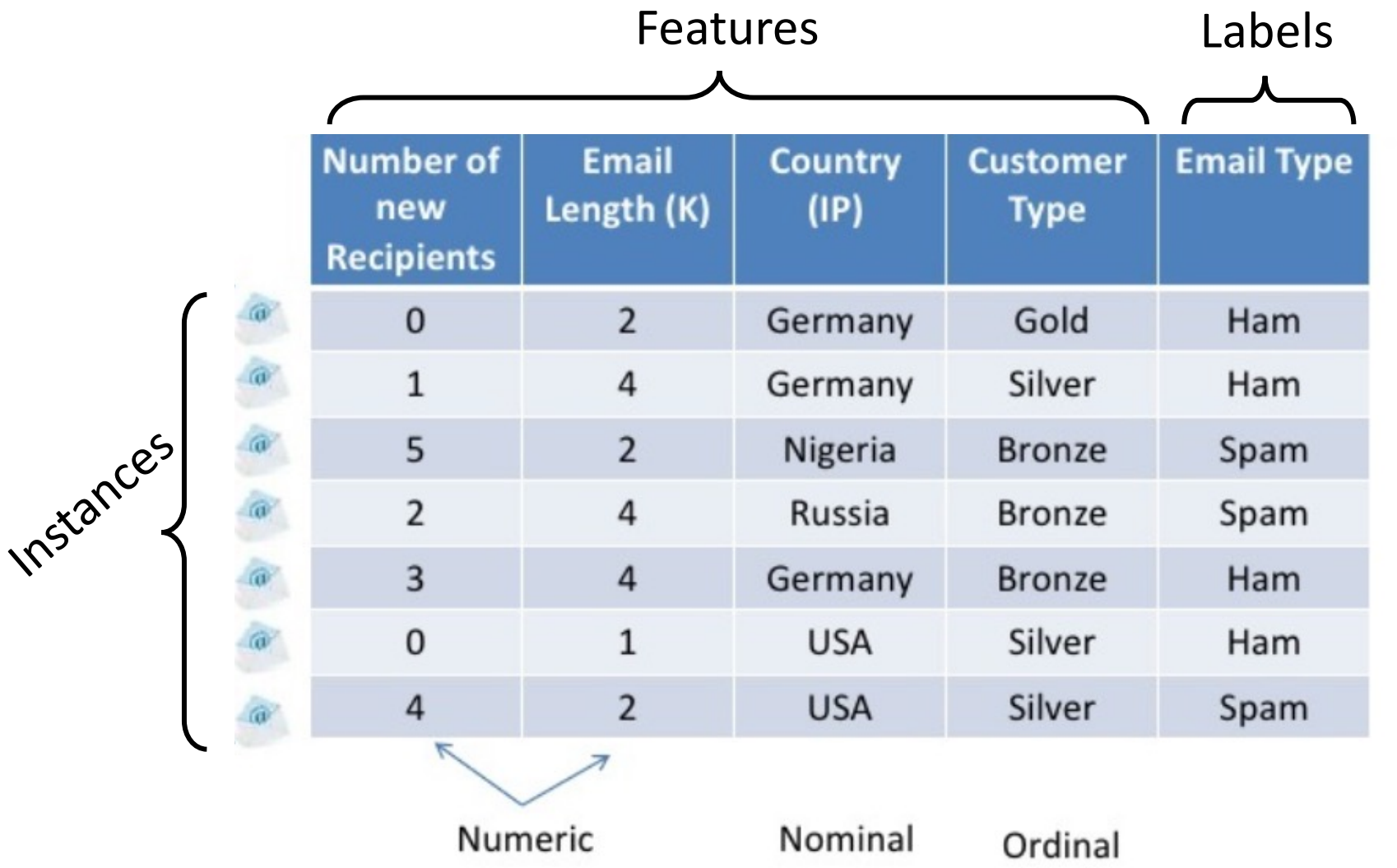
02:00) Feature transform & nonlinearity

02:20) Error decomposition & fitting

02:45) ---- Break ----

03:00) Tutorial (**Oren**): Linear Classification

Supervised Learning



Supervised Learning

Features

Labels

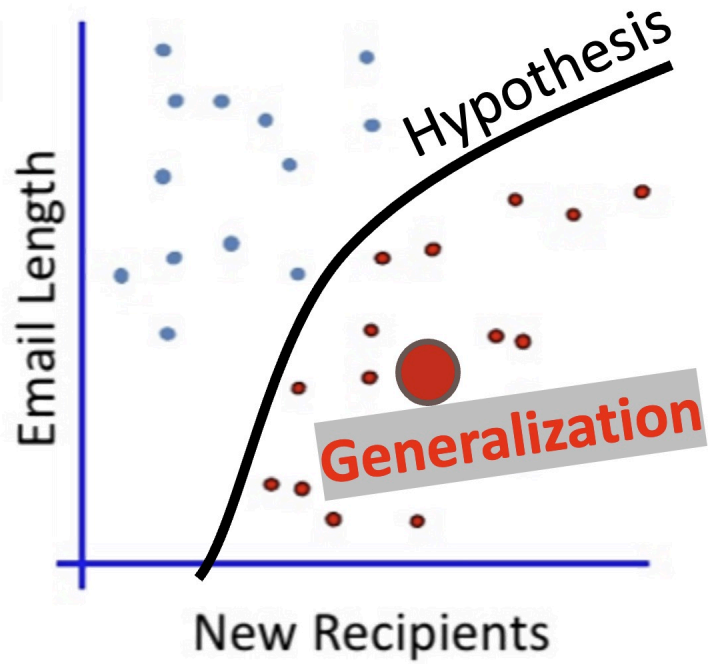
Instances

Number of new Recipients	Email Length (K)	Country (IP)	Customer Type	Email Type
0	2	Germany	Gold	Ham
1	4	Germany	Silver	Ham
5	2	Nigeria	Bronze	Spam
2	4	Russia	Bronze	Spam
3	4	Germany	Bronze	Ham
0	1	USA	Silver	Ham
4	2	USA	Silver	Spam

Numeric

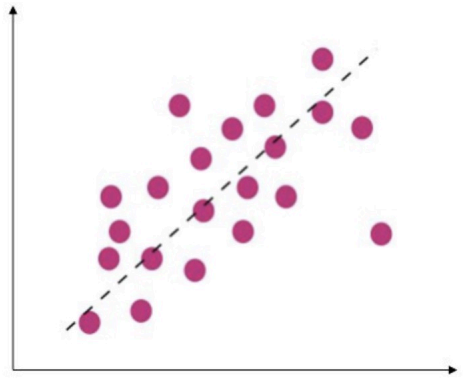
Nominal

Ordinal

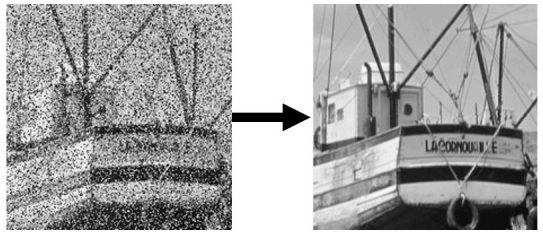


Supervised Learning

Regression



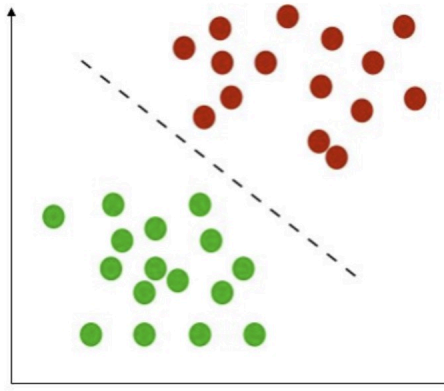
E.g. Image denoising



E.g. Object localization



Classification



E.g. Image classification



Supervised Learning

Learning Algorithm

Training set

Hypothesis

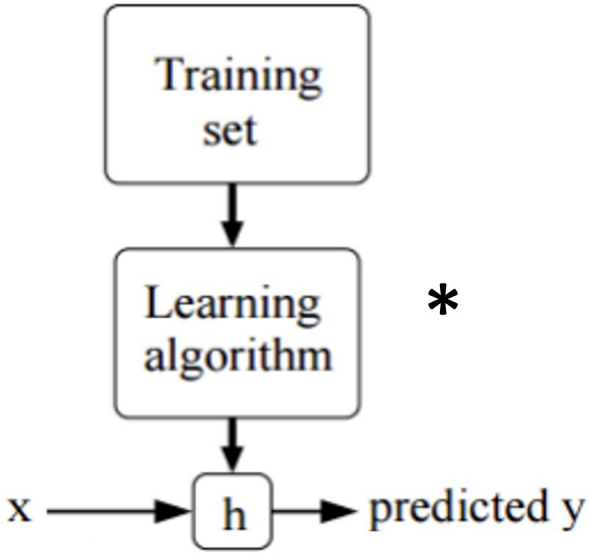
$$A(S) = h$$

Hypothesis class
$\mathcal{H} = \{h_1, h_2 \dots\}$

$h(x) \approx y$

Loss
$\mathcal{L}$

Optimization method



*

Inputs
X

$$\begin{bmatrix} \text{---} \ \mathbf{x_1}^T \ \text{---} \\ \text{---} \ \mathbf{x_2}^T \ \text{---} \\ \vdots \\ \text{---} \ \mathbf{x_M}^T \ \text{---} \end{bmatrix}$$

Labels
Y

$$\begin{bmatrix} \mathbf{y_1}^T \\ \mathbf{y_2}^T \\ \vdots \\ \mathbf{y_M}^T \end{bmatrix}$$

,

* Diagram by Andrew NG

Linear Regression

Hypothesis class: Linear →

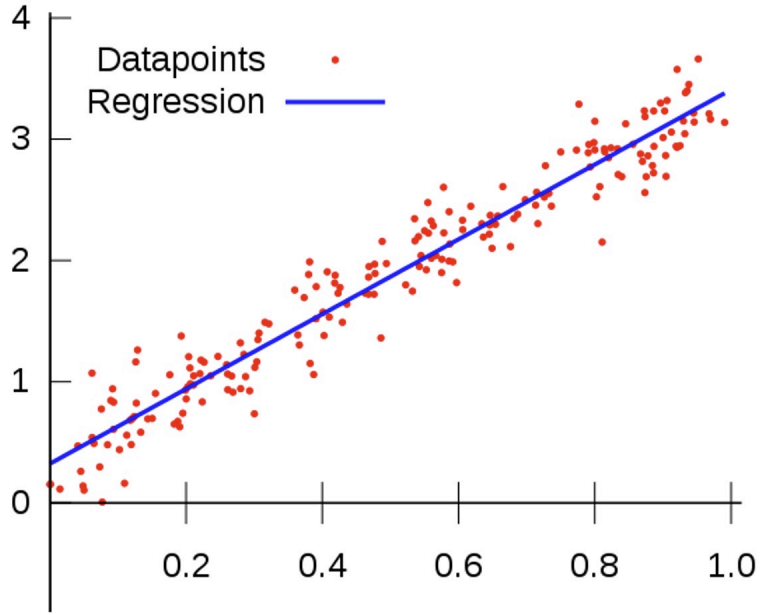
$$\mathcal{H} = \{h_{\boldsymbol{\theta}} | \, \boldsymbol{\theta} \epsilon \, \mathbb{R}^{N+1}\} \;, \quad h_{\boldsymbol{\theta}}(\boldsymbol{x}) = \theta_0 + \sum_{j=1}^{N} \theta_j x_j = \theta_0 + \tilde{\boldsymbol{\theta}}^T \boldsymbol{x} = \boldsymbol{\theta}^T \begin{pmatrix} 1 \\ | \\ \boldsymbol{x} \\ | \end{pmatrix}$$

*Bias= Just add **1** at top of the input vec!*

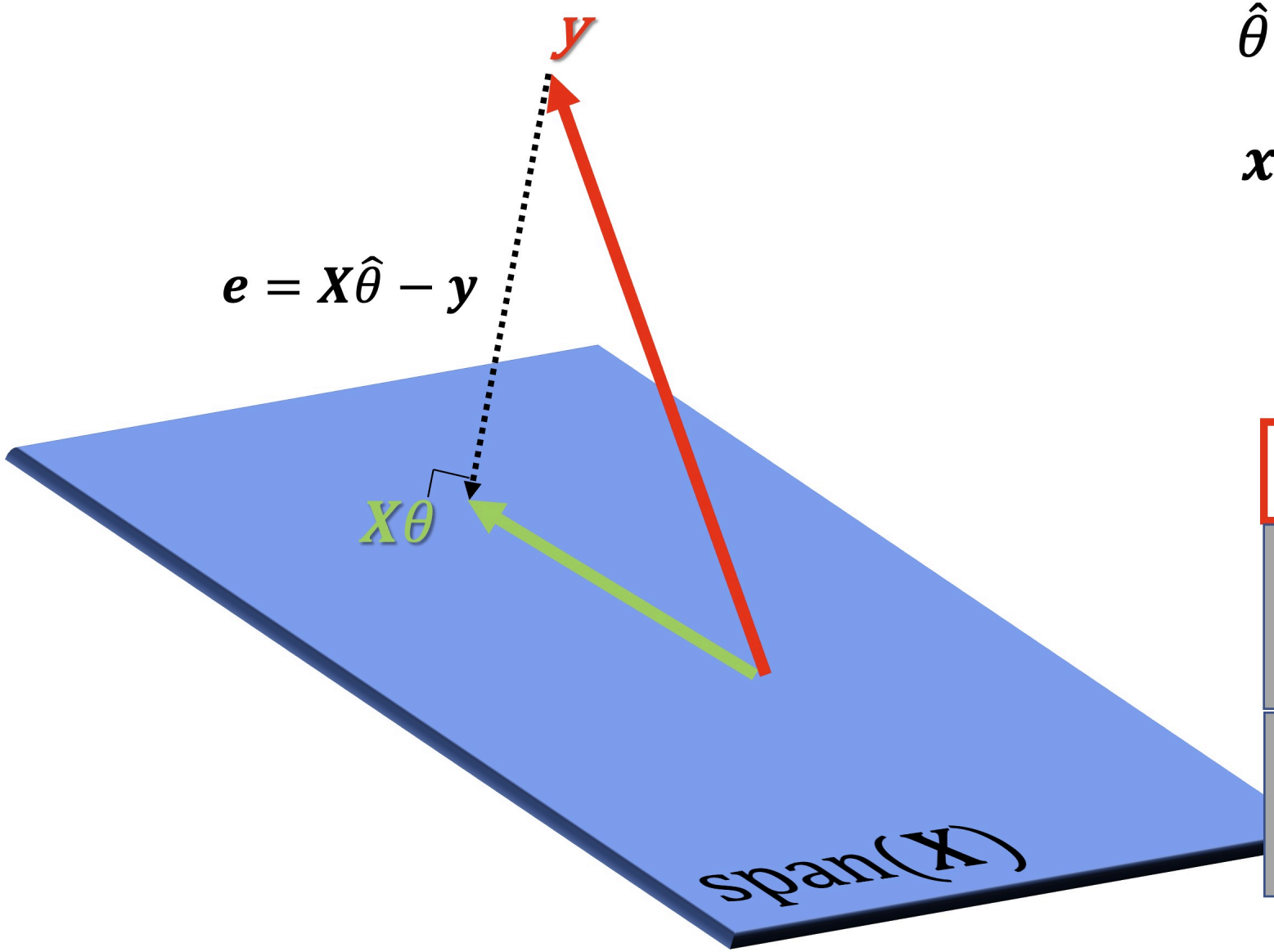
Loss: Mean Squared Error →

$$\mathcal{L} = \frac{1}{2M} \sum_{i=1}^{M} (h_\theta(\boldsymbol{x}_i) - y_i)^2 = \frac{1}{2M} \|\boldsymbol{X}\boldsymbol{\theta} - \boldsymbol{y}\|^2$$

Optimization method: Normal equations / Gradient Descent



Normal Equations (intuition)



$$\hat{\theta} = argmin_{\theta} \|y - X\theta\|^2$$

$$x \perp e \quad \forall\ x \in span(X)$$

$$\Downarrow$$

$$X^T(X\hat{\theta} - y) = 0$$

$$\Downarrow$$

$$\hat{\theta} = (X^T X)^{-1} X^T y \quad *$$

Formal proof: Try

Also think about: is $X^T X$ invertible?

$* \; if \; X^T X \; invertible$

Normal Equations

Q: Will normal equations always be practical?

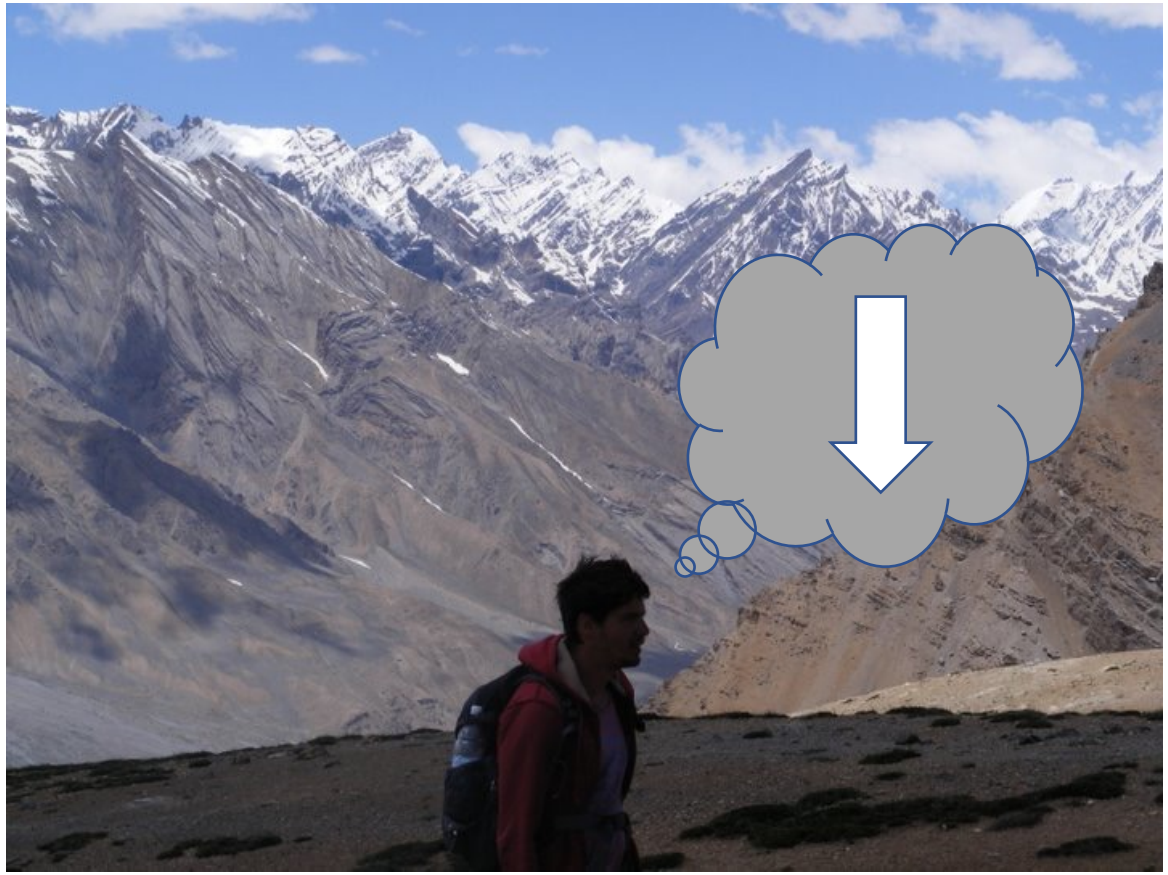
A: No;

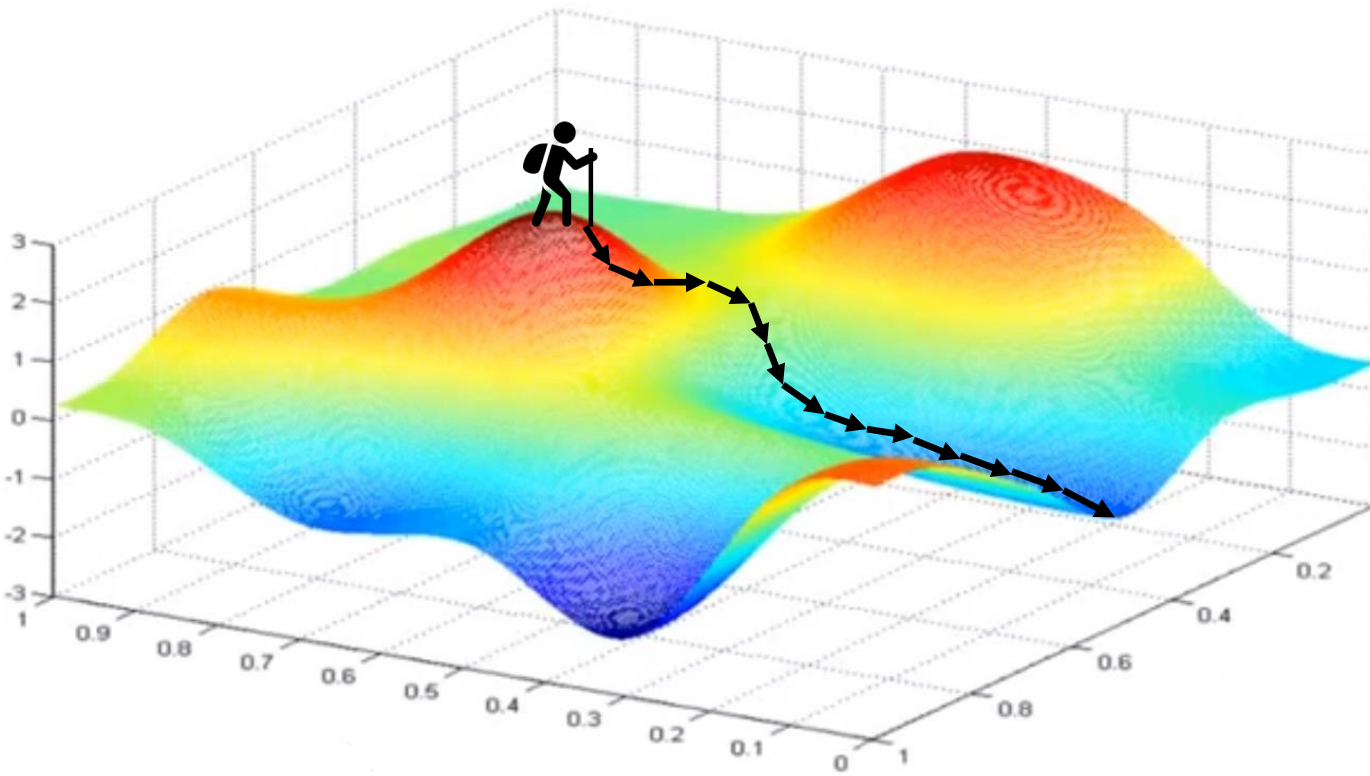
1. Inverting $\boldsymbol{X}^T\boldsymbol{X}$ may cost unreasonable memory / time
2. Sometimes not applicable: Regularization? Different loss? Non-linearity?



Gradient descent

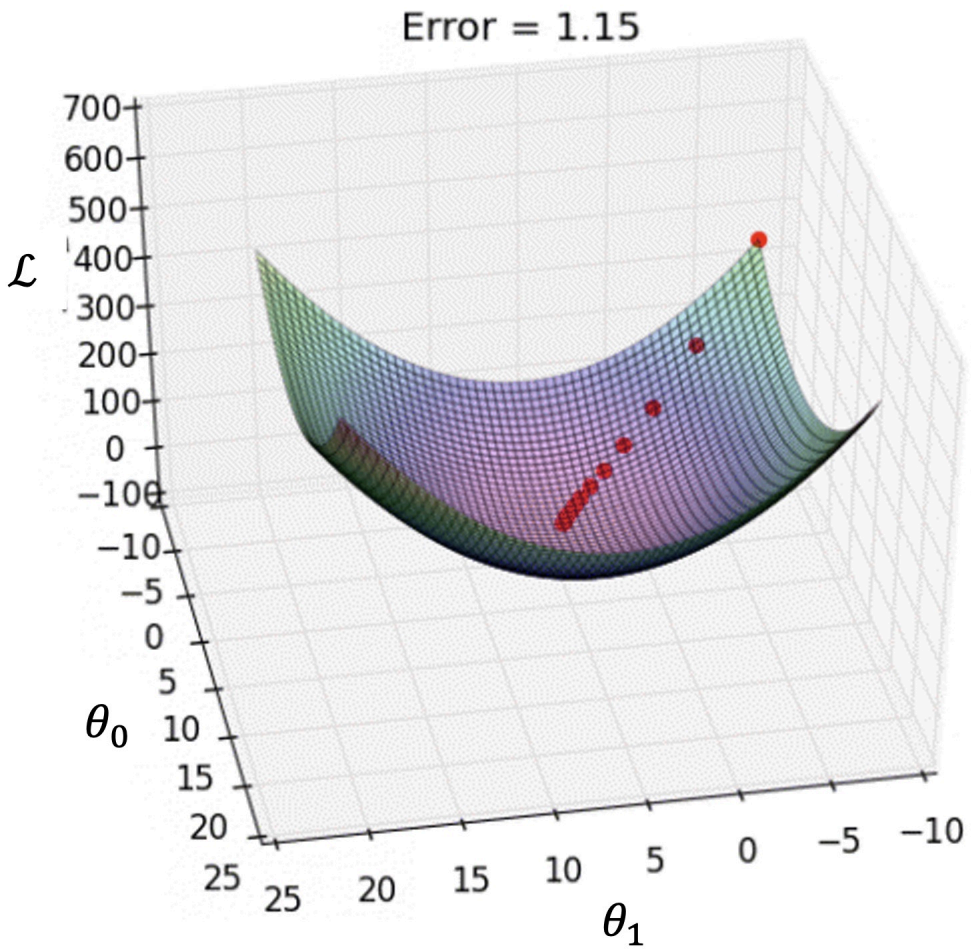
Possible solution: Iteratively reduce loss

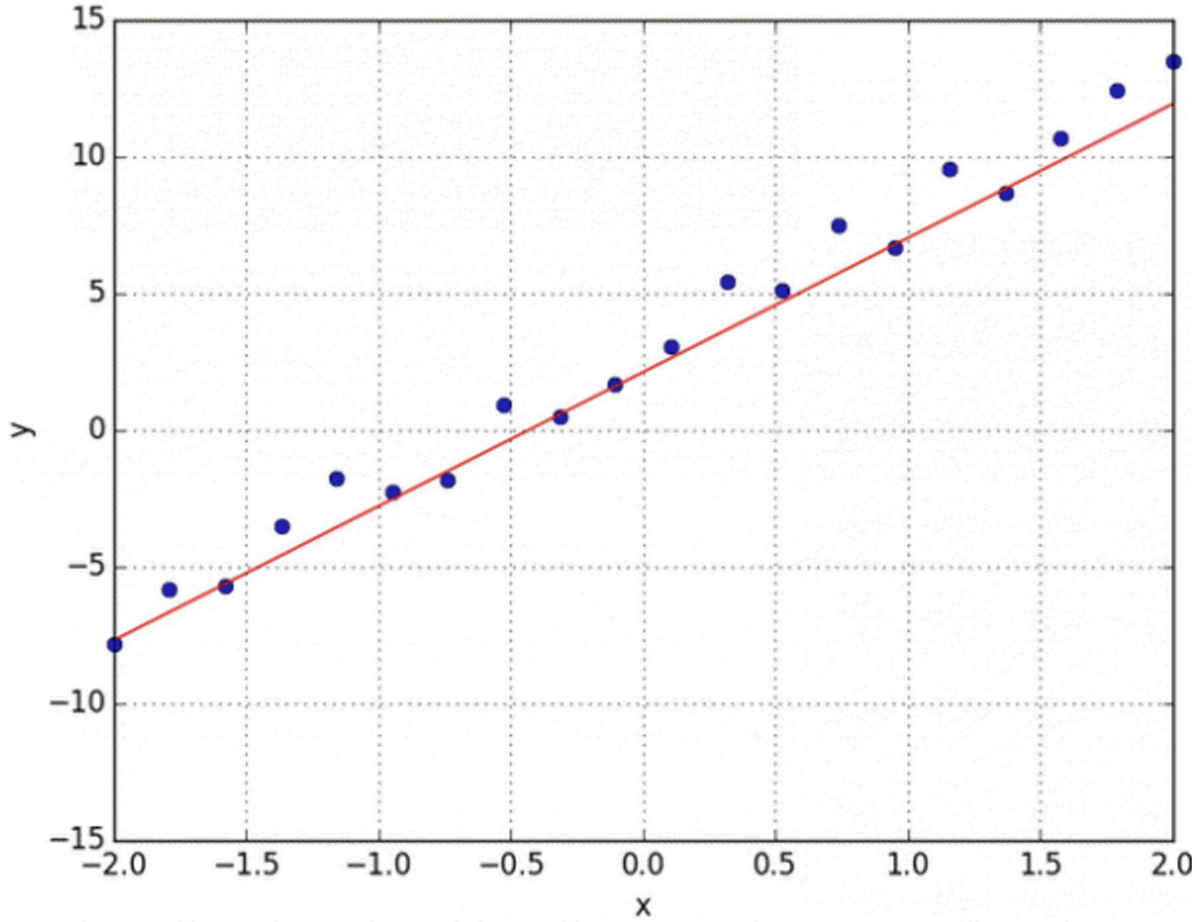




Can we guarantee global min?

Gradient descent





Parameter space: $\mathcal{L}(\boldsymbol{\theta}; S)$

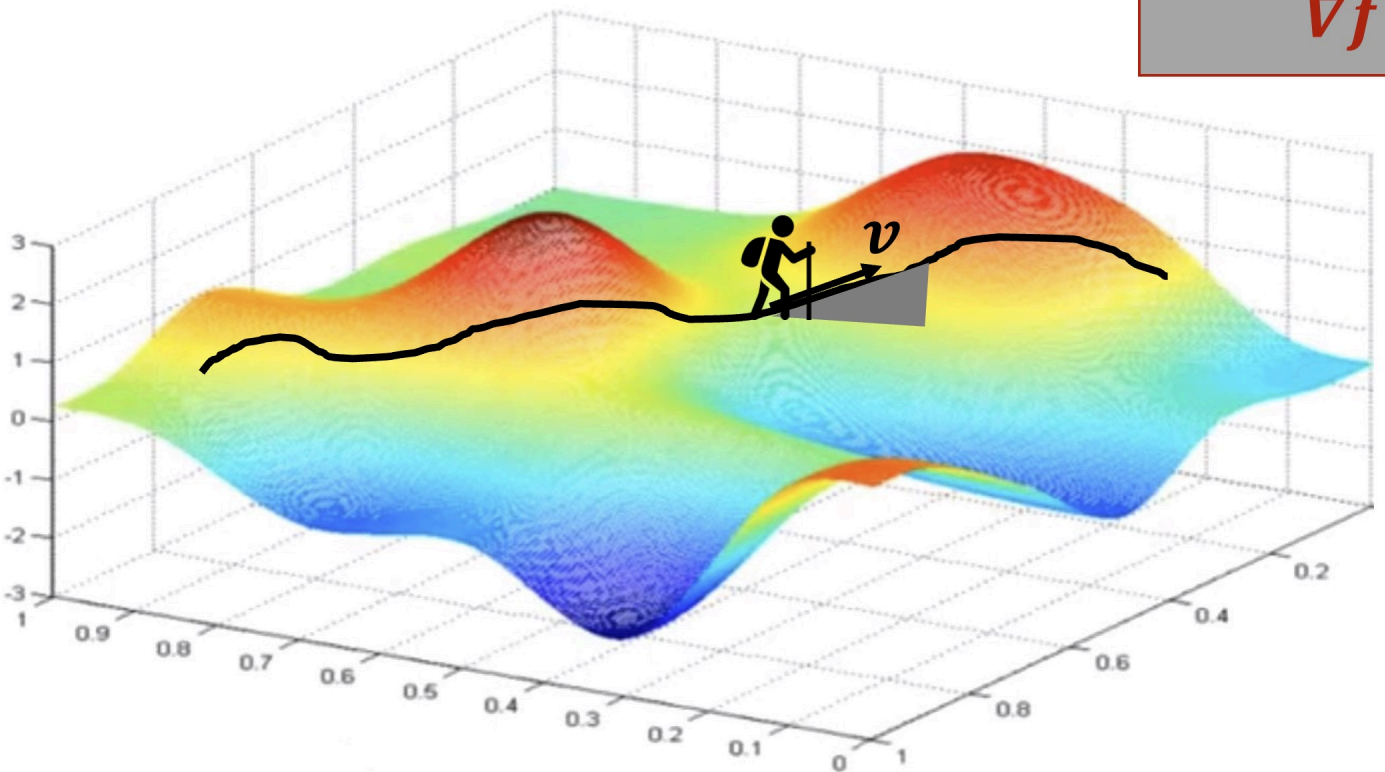
Data space: $h_{\boldsymbol{\theta}}(\boldsymbol{x})$

Calculus reminder: Directional derivative

$$\lim_{\varepsilon \to 0} \frac{f(\boldsymbol{x} + \varepsilon \boldsymbol{v}) - f(\boldsymbol{x})}{\varepsilon \|\boldsymbol{v}\|} = \frac{1}{\|\boldsymbol{v}\|} \sum v_i \frac{\partial f}{\partial x_i} = \frac{\boldsymbol{v}^T}{\|\boldsymbol{v}\|} \begin{pmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \\ \vdots \\ \frac{\partial f}{\partial x_N} \end{pmatrix} = \frac{1}{\|\boldsymbol{v}\|} \langle \boldsymbol{v}, \nabla f \rangle$$

If differentiable

Gradient!
$\vec{\nabla} f$

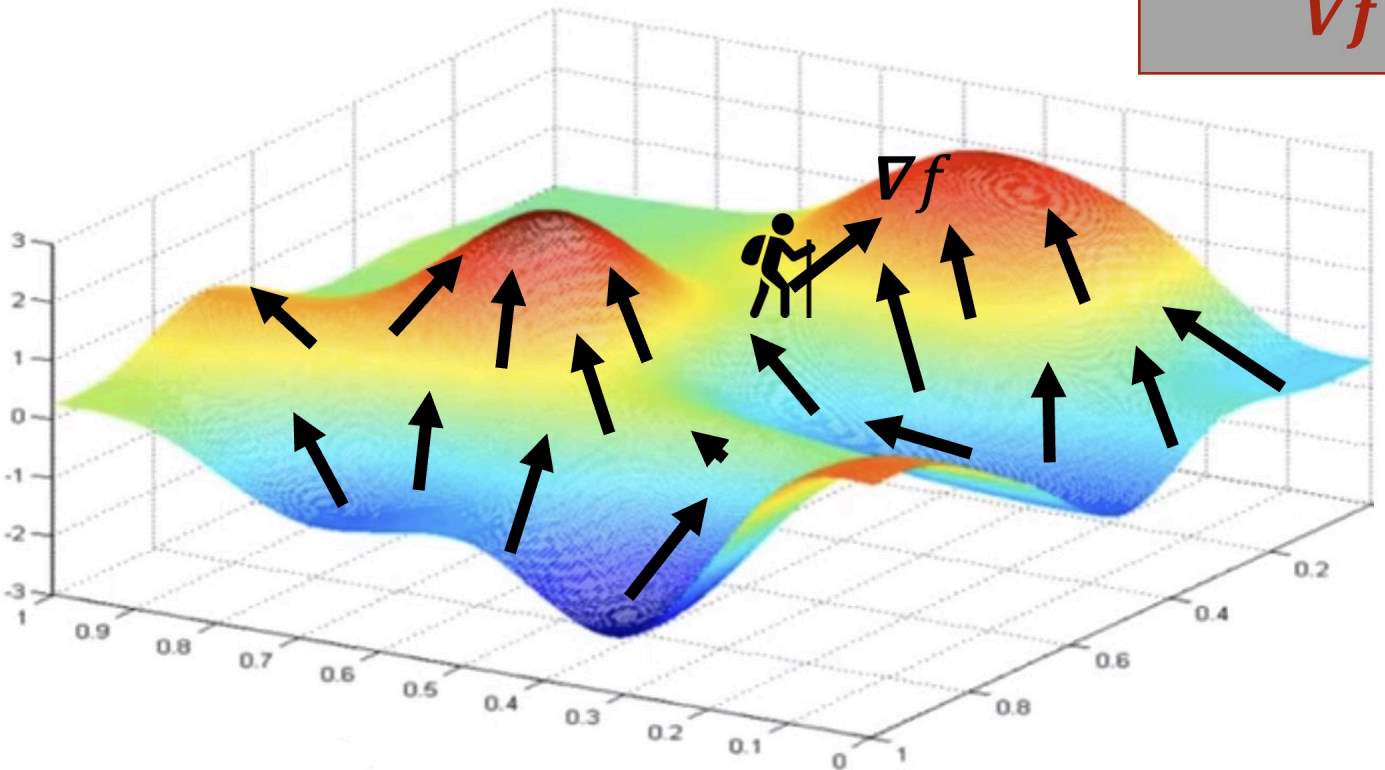


Calculus reminder: Directional derivative

$$\lim_{\varepsilon \to 0} \frac{f(\boldsymbol{x} + \varepsilon \boldsymbol{v}) - f(\boldsymbol{x})}{\varepsilon \|\boldsymbol{v}\|} = \frac{1}{\|\boldsymbol{v}\|} \sum v_i \frac{\partial f}{\partial x_i} = \frac{\boldsymbol{v}^T}{\|\boldsymbol{v}\|} \begin{pmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \\ \vdots \\ \frac{\partial f}{\partial x_N} \end{pmatrix} = \frac{1}{\|\boldsymbol{v}\|} \langle \boldsymbol{v}, \nabla f \rangle$$

If differentiable

Gradient!
$\vec{\nabla} f$



<u>According to Cauchy-Schwarz inequality</u>:

- Max value is $\|\nabla f\|$
- Obtained when $\boldsymbol{v}$ is parallel to ∇f

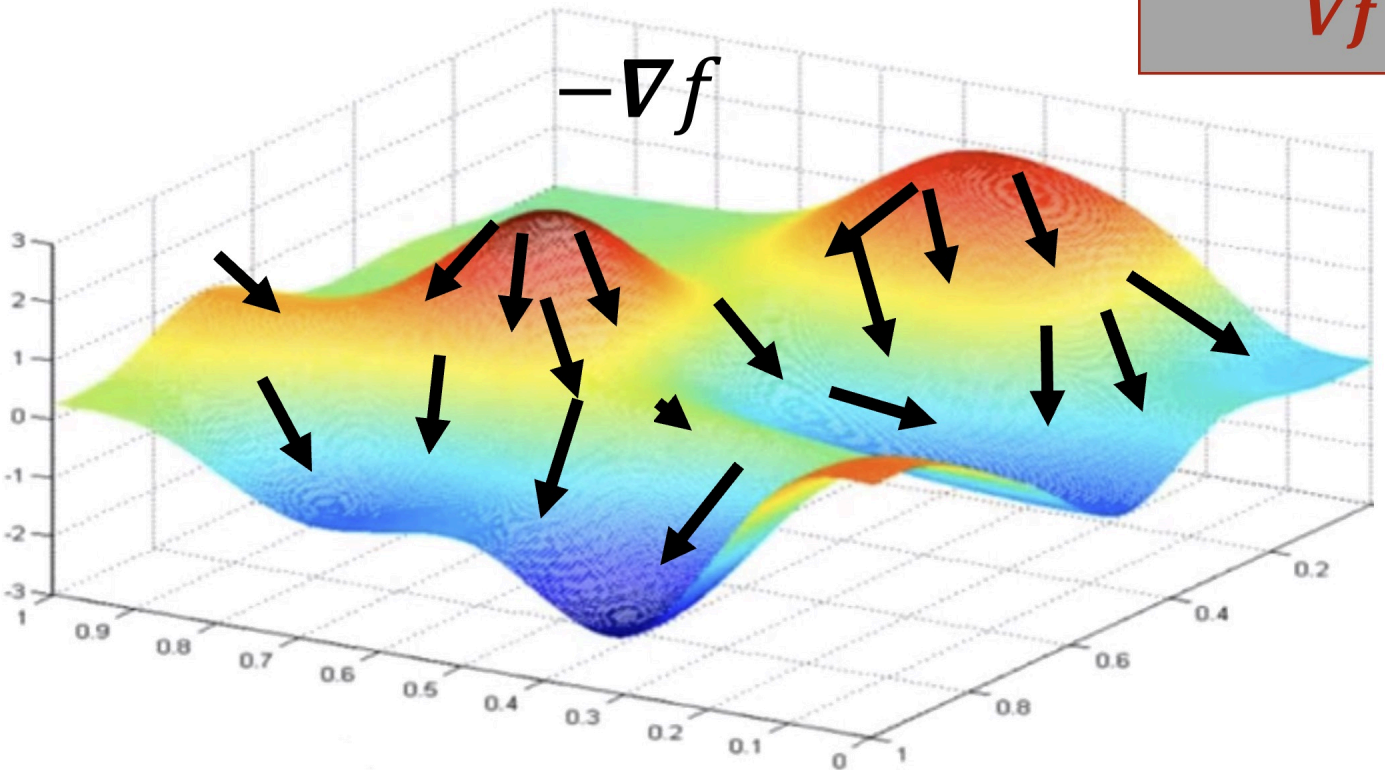
- Gradient directs to steepest ascent.
- It's size is the max steepness.

Calculus reminder: Directional derivative

$$\lim_{\varepsilon \to 0} \frac{f(\boldsymbol{x} + \varepsilon \boldsymbol{v}) - f(\boldsymbol{x})}{\varepsilon \|\boldsymbol{v}\|} = \frac{1}{\|\boldsymbol{v}\|} \sum v_i \frac{\partial f}{\partial x_i} = \frac{\boldsymbol{v}^T}{\|\boldsymbol{v}\|} \begin{pmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \\ \vdots \\ \frac{\partial f}{\partial x_N} \end{pmatrix} = \frac{1}{\|\boldsymbol{v}\|} \langle \boldsymbol{v}, \nabla f \rangle$$

If differentiable

Gradient!
$\vec{\nabla} f$



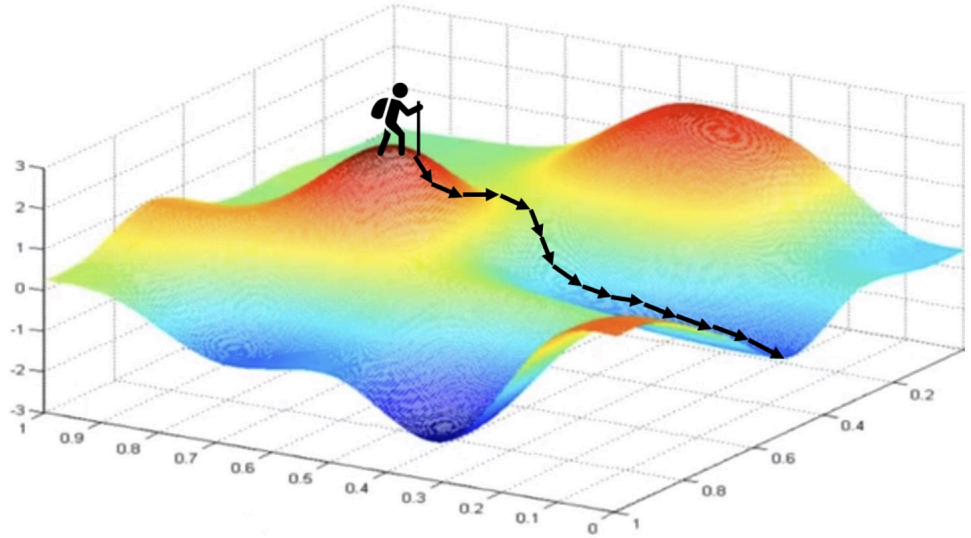
<u>According to Cauchy-Schwarz inequality</u>:

- Max value is $\|\nabla f\|$
- Obtained when $\boldsymbol{v}$ is parallel to ∇f

- Gradient directs to steepest ascent.
- It's size is the max steepness.

Gradient descent

$$\nabla\mathcal{L}(\theta_0, \theta_1 \dots \theta_N) = \begin{pmatrix} \frac{\partial\mathcal{L}}{\partial\theta_0} \\ \frac{\partial\mathcal{L}}{\partial\theta_1} \\ \vdots \\ \frac{\partial\mathcal{L}}{\partial\theta_N} \end{pmatrix}$$





Augustin Louis Cauchy

1. Initialize $\theta \sim$ Random
2. Repeat until convergence:

{

$$\boldsymbol{\theta} := \boldsymbol{\theta} - \alpha\nabla\mathcal{L}(\boldsymbol{\theta}; S)$$

}

α: Learning rate

Gradient descent



Full batch Gradient Descent

$$\boldsymbol{\theta} := \boldsymbol{\theta} - \alpha \boldsymbol{\nabla} \mathcal{L}(\boldsymbol{\theta}; S)$$

Stochastic Gradient Descent

$$\boldsymbol{\theta} := \boldsymbol{\theta} - \alpha \boldsymbol{\nabla} \mathcal{L}(\boldsymbol{\theta};\ x_i, y_i)$$

$$i \sim U[1, M]$$

Stochastic Mini-Batch Gradient Descent

$$\boldsymbol{\theta} := \boldsymbol{\theta} - \alpha \boldsymbol{\nabla} \mathcal{L}(\boldsymbol{\theta}; \boldsymbol{B})$$

$$\boldsymbol{B} = \{(x_i, y_i)\ ;\quad \boldsymbol{i} \sim U^m[1, M]\}$$

Gradient descent



Full batch Gradient Descent

Stochastic Gradient Descent

Stochastic Mini-Batch Gradient Descent

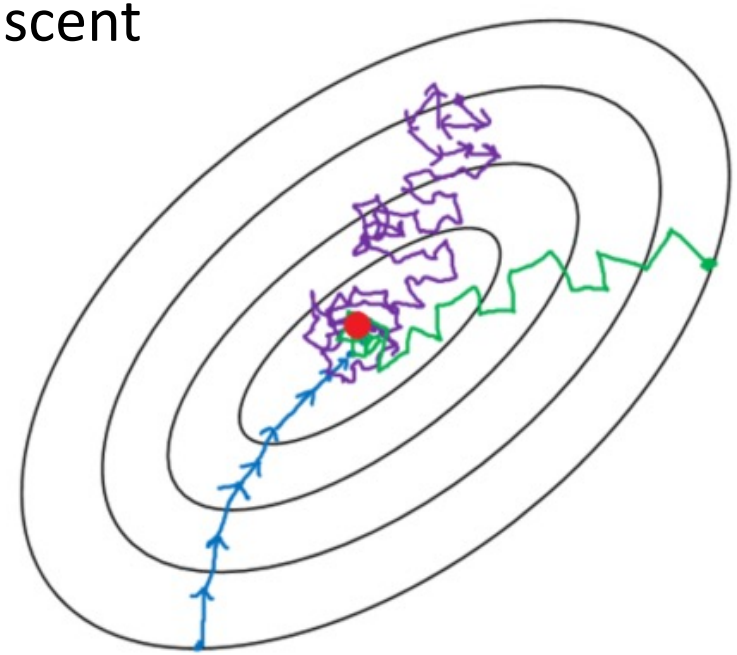


Figure by Z² Little on Medium

Gradient descent for Linear Regression

$$\mathcal{L} = \frac{1}{2m}\sum_{i=1}^{m}(\boldsymbol{\theta}^T \boldsymbol{x}_i - y_i)^2$$

$$\frac{\partial}{\partial \boldsymbol{\theta}}\mathcal{L} = \frac{1}{m}\sum_{i=1}^{m}(\boldsymbol{\theta}^T \boldsymbol{x}_i - y_i)\boldsymbol{x}_i = \frac{1}{m}\sum_{i=1}^{m}\boldsymbol{x}_i(\boldsymbol{X\theta} - \boldsymbol{y})_{\boldsymbol{i}} = \boldsymbol{X}^T(\boldsymbol{X\theta} - \boldsymbol{y})$$

$\boldsymbol{e}$

Repeat until convergence:

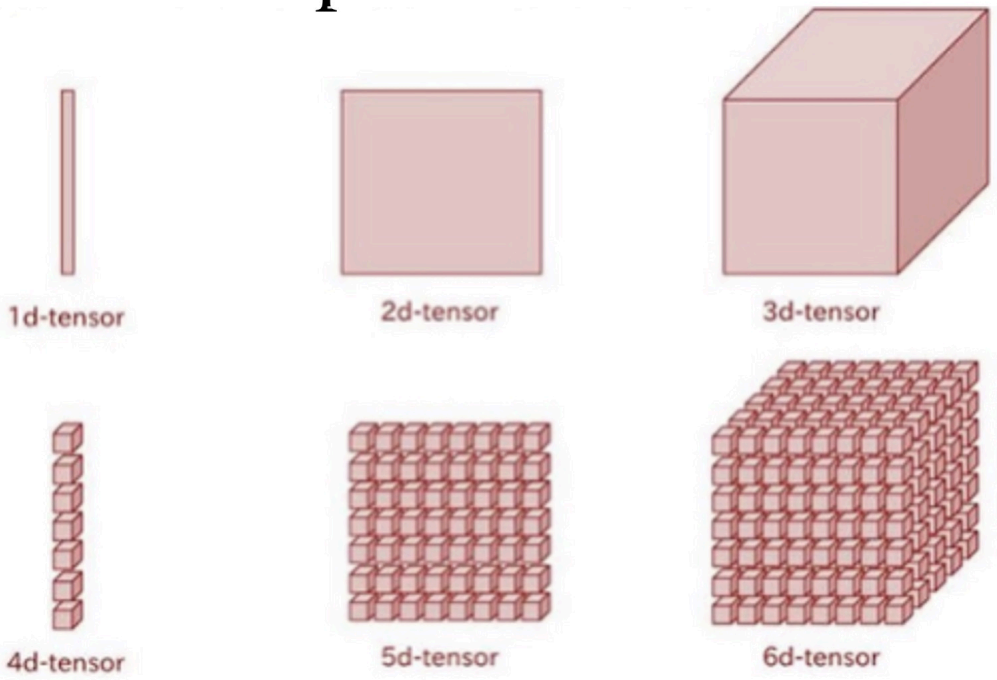
{

$$\boldsymbol{\theta} := \boldsymbol{\theta} - \frac{\alpha}{m}\boldsymbol{X}^T\boldsymbol{e}$$

}

Q: Find the relation between convergence and Normal Equations

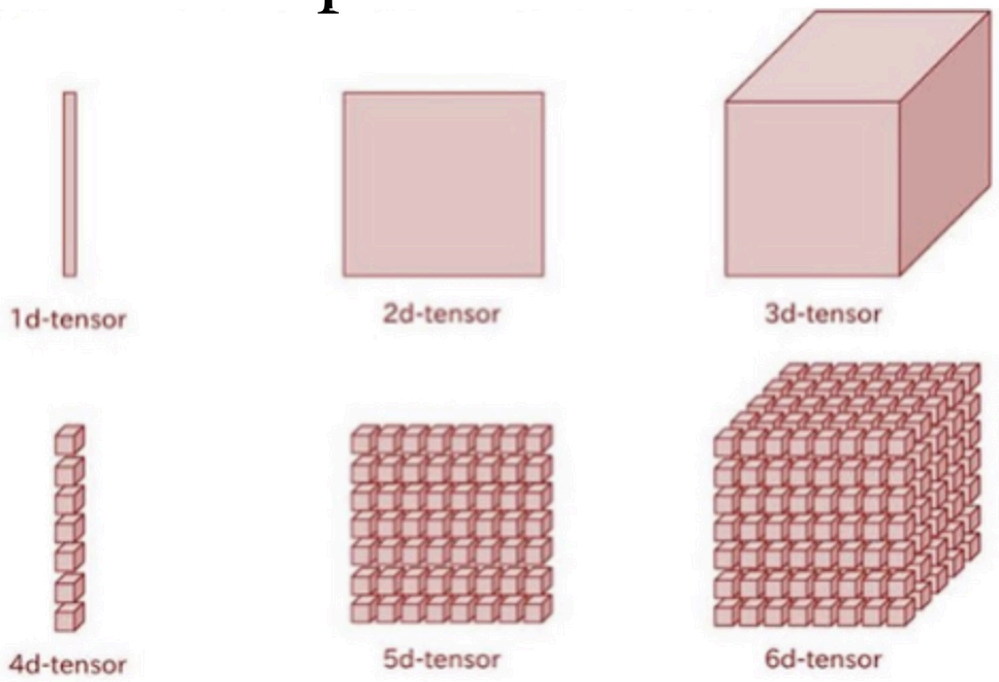
Tensor operations



Tensor multiplication $C_{ik} = \Sigma_j A_{ij} B_{jk}$



Tensor operations



Tensor multiplication

$$C_{ik} = \Sigma_j A_{ij} B_{jk}$$



j appears twice → summed away (contraction)
i, k appear once → survive in output

Einstein summation

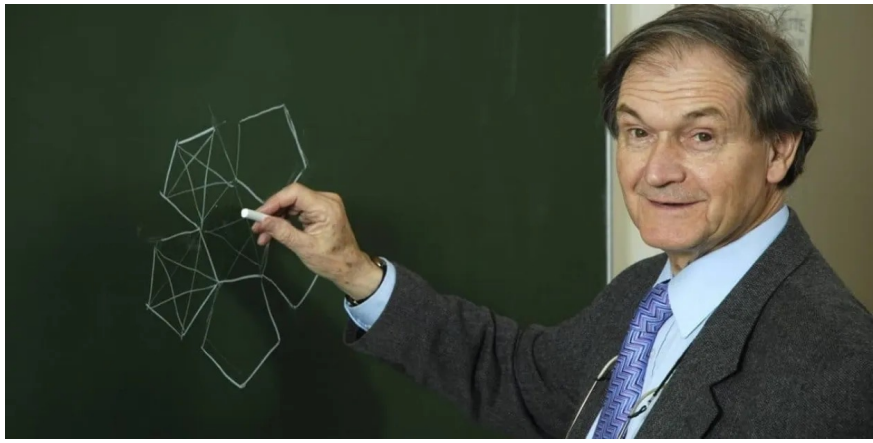
shared index → contract
free index → expand

Note: trivial case included

C_{ik} ∘ ○ ○

This is a tensor
(not a term)

Penrose graphical notation









tensor = box

index = leg

wire = contraction



scalar (order 0)



vector (order 1)

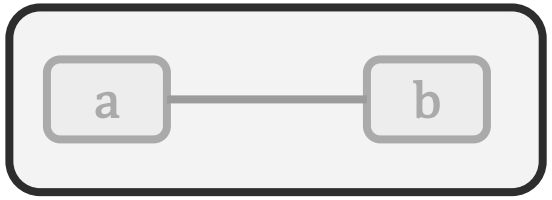


matrix (order 2)



tensor (order 4)

inner product



$a_i\, b_i$

→ *scalar (no free legs)*



order 0

outer product



$a_i\, b_j$

→ *matrix (2 free legs)*



order 2

Penrose graphical notation



mat. multiply

$A_{ij}\, B_{jk}$



mat. outer product

$A_{ij}\, B_{kl}$



mat. scalar product (Forbenious)

$A_{ij}\, B_{ij}$



mat. trace

A_{ii}



mat-vec product

$A_{ij}\, v_j$



mat. triple chain

$A_{ij}\, B_{jk}\, C_{kl}$

Penrose graphical notation



mat. multiply

$A_{ij}\, B_{jk}$



mat. outer product

$A_{ij}\, B_{kl}$



mat. scalar product (Forbenious)

$A_{ij}\, B_{ij}$



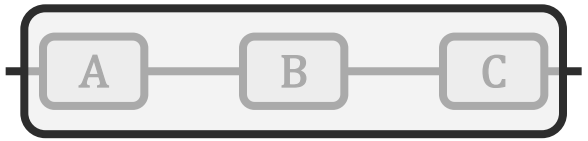
mat. trace

A_{ii}



mat-vec product

$A_{ij}\, v_j$



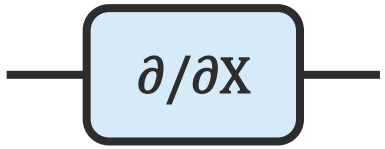
mat. triple chain

$A_{ij}\, B_{jk}\, C_{kl}$

<u>Prove (visually)</u>:

Flatten 2 matrices to be vecs and take scalar product **=** *Multiply the matrices and take trace*

The differentiation operator

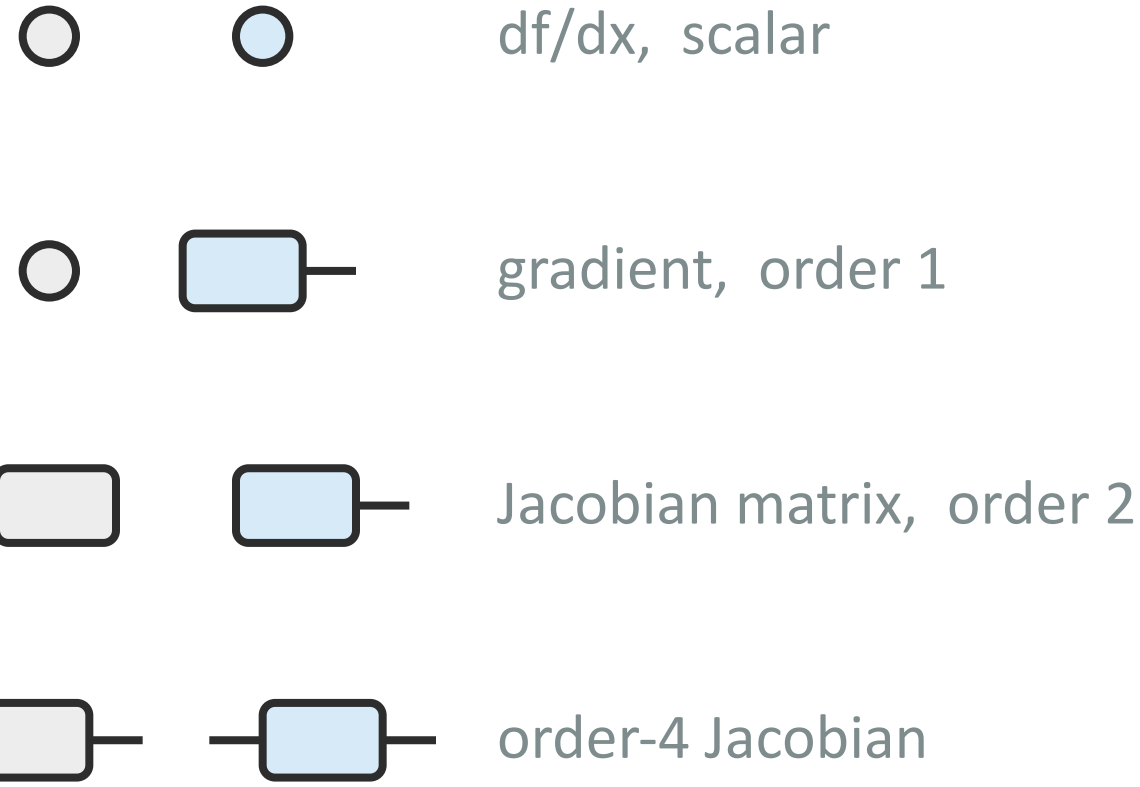


Differentiate = outer prod

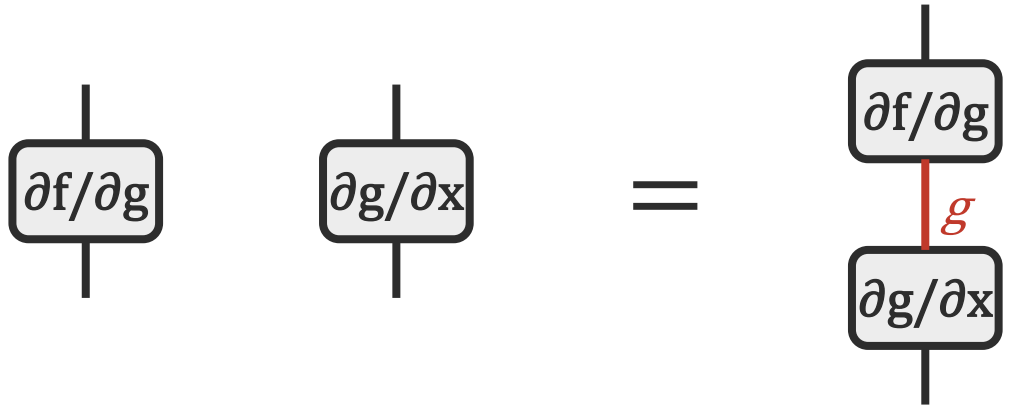
$$\partial f/\partial X = f \otimes \partial/\partial X$$



order(J) = order(f) + order(X)

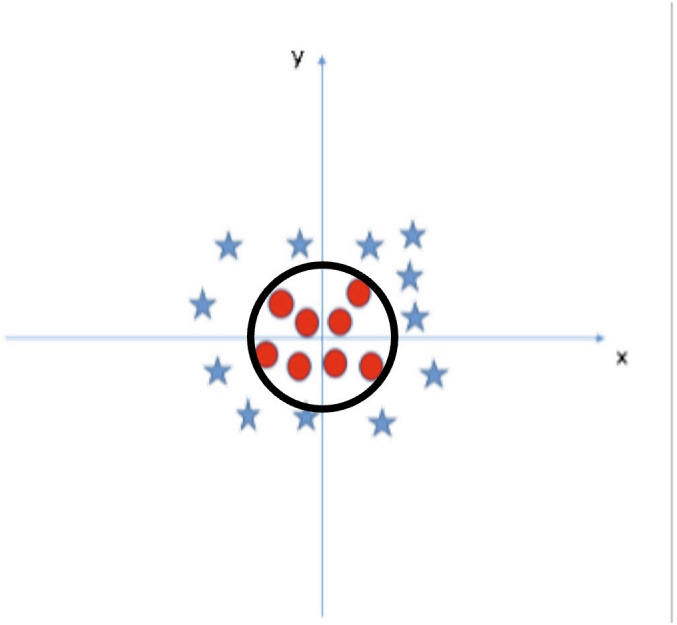


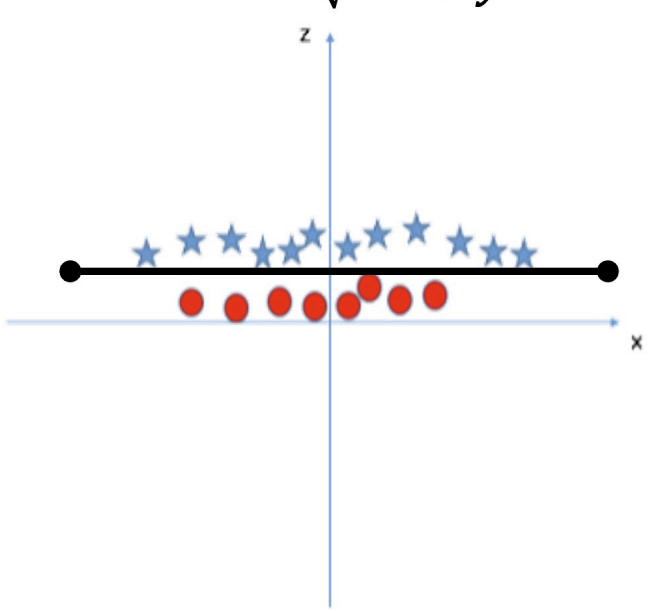
Chain rule = inner prod

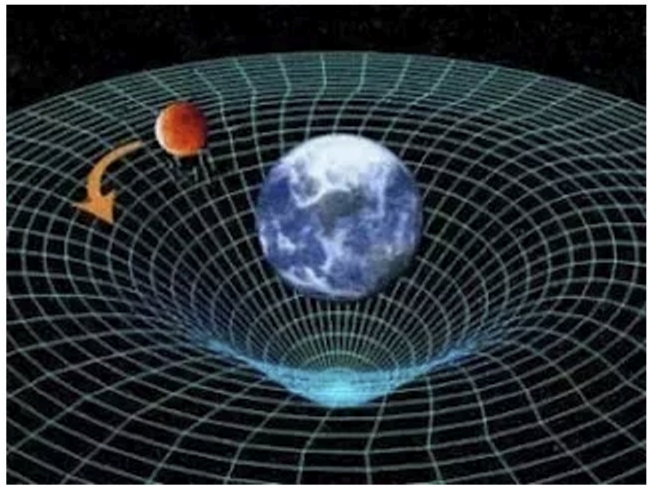


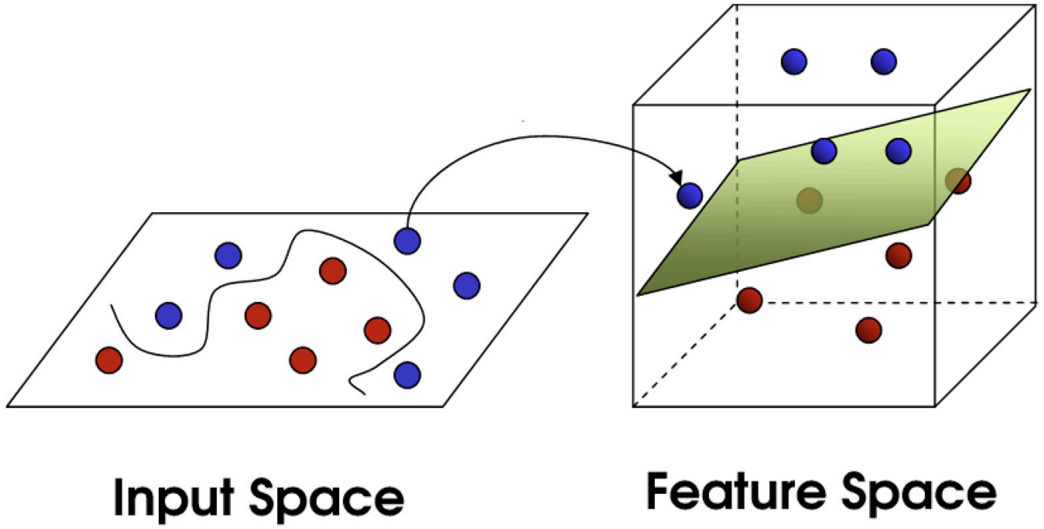
Feature transform

$$z = \sqrt{x^2 + y^2}$$

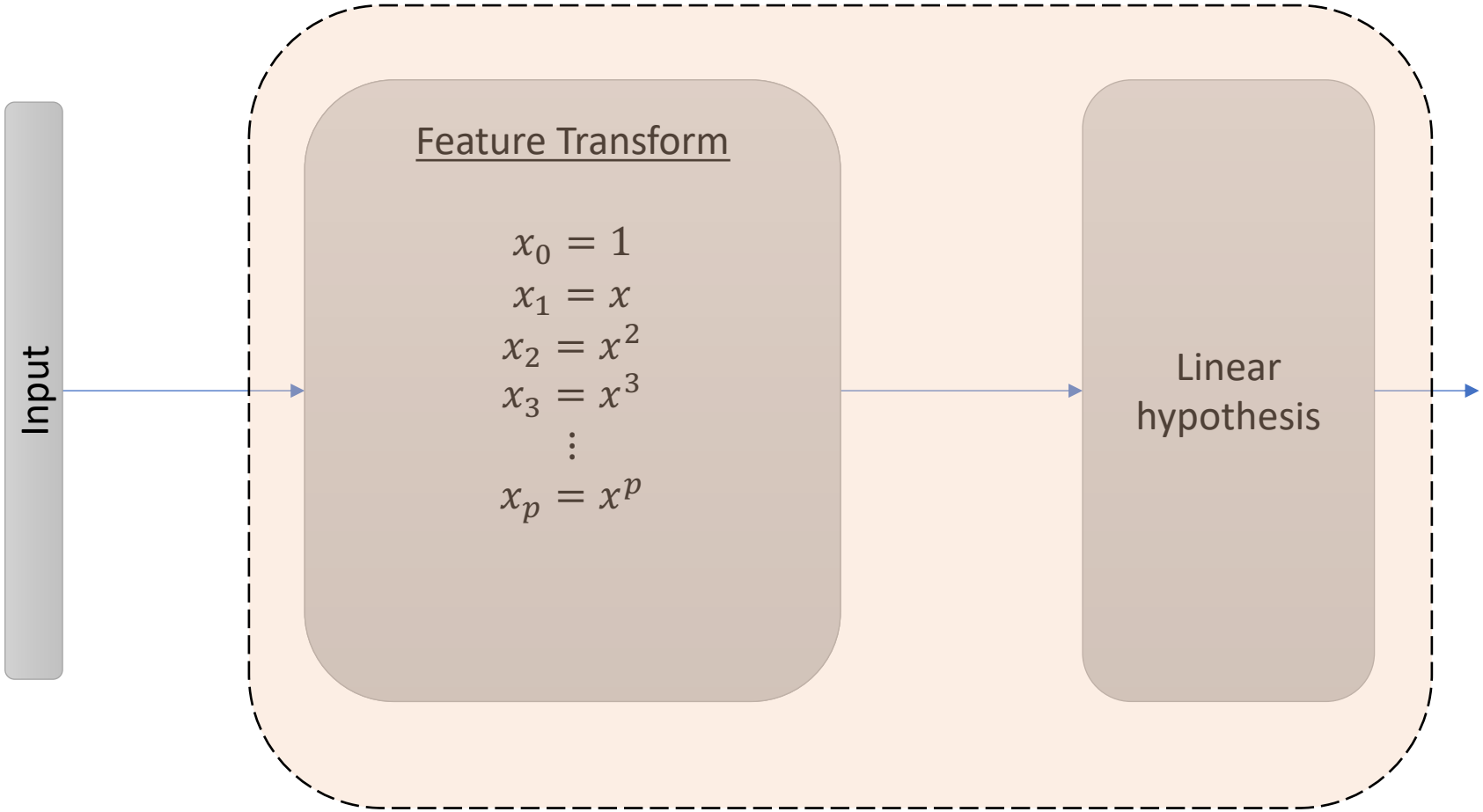








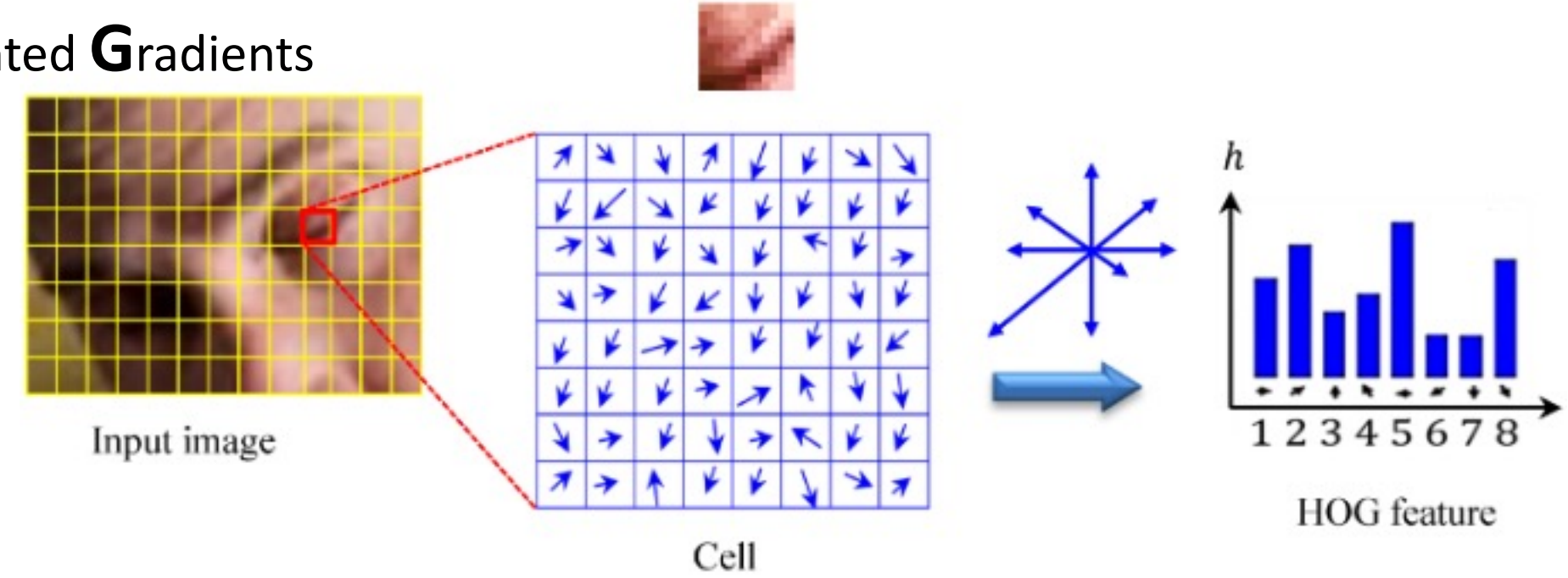
Feature transform



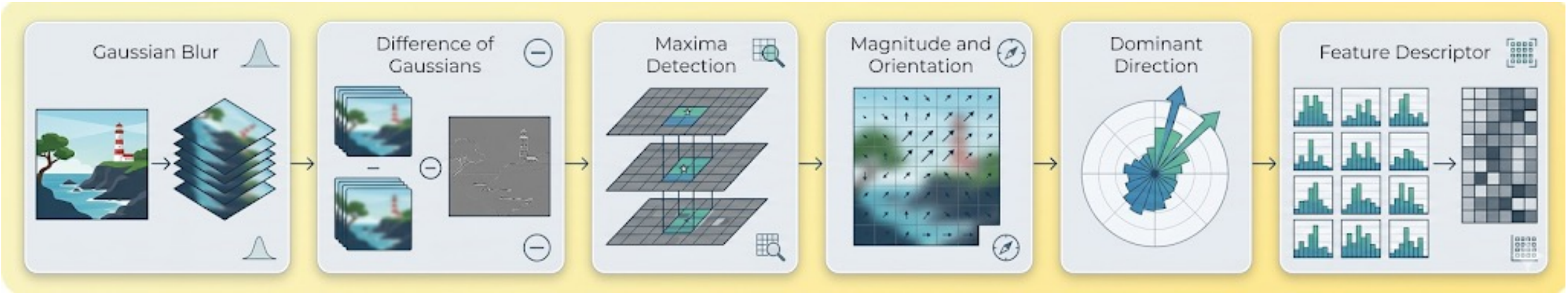
Non-linear hypothesis!
(Polynomial)

Classical image features

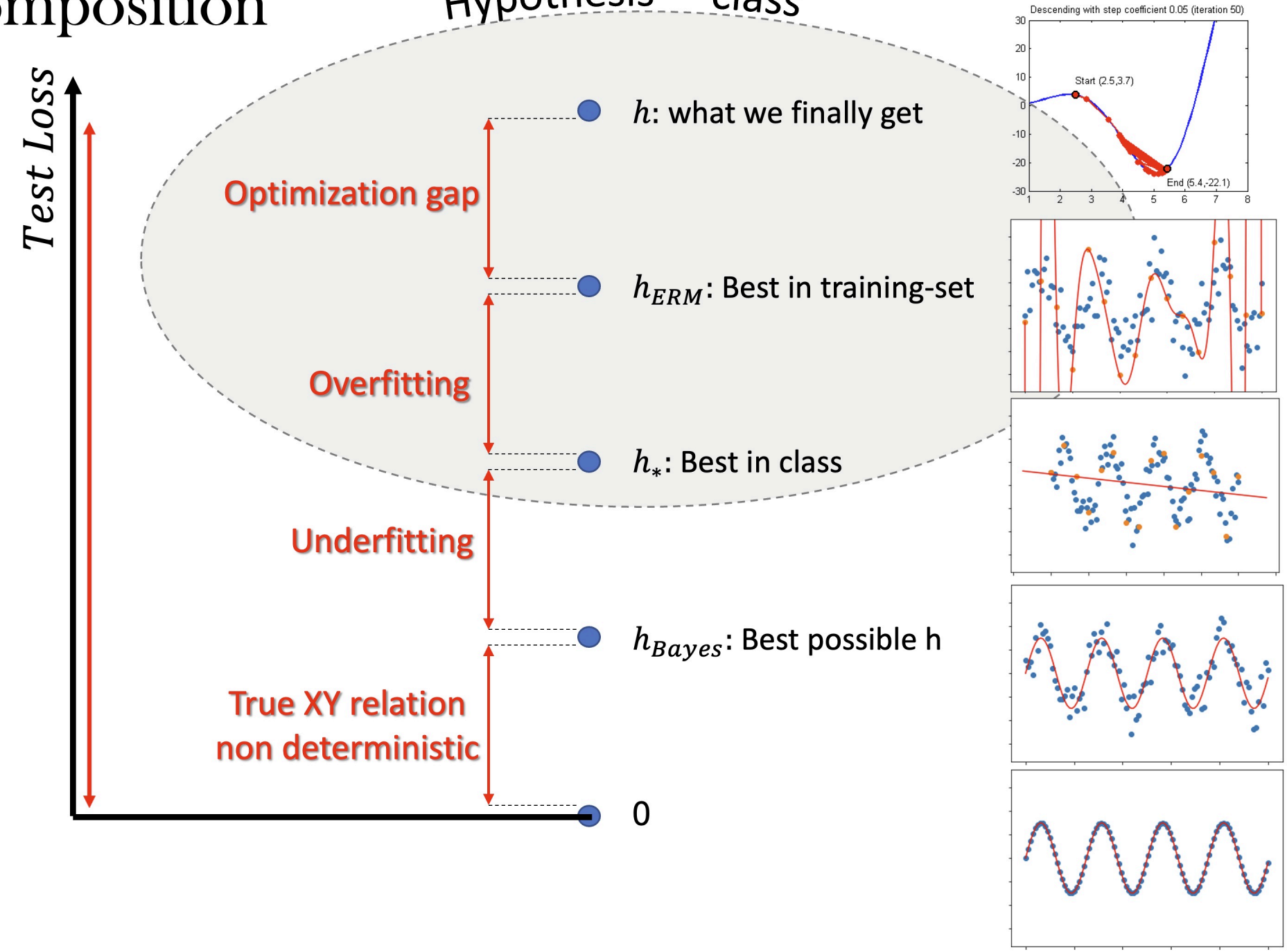
Histogram of **O**riented **G**radients



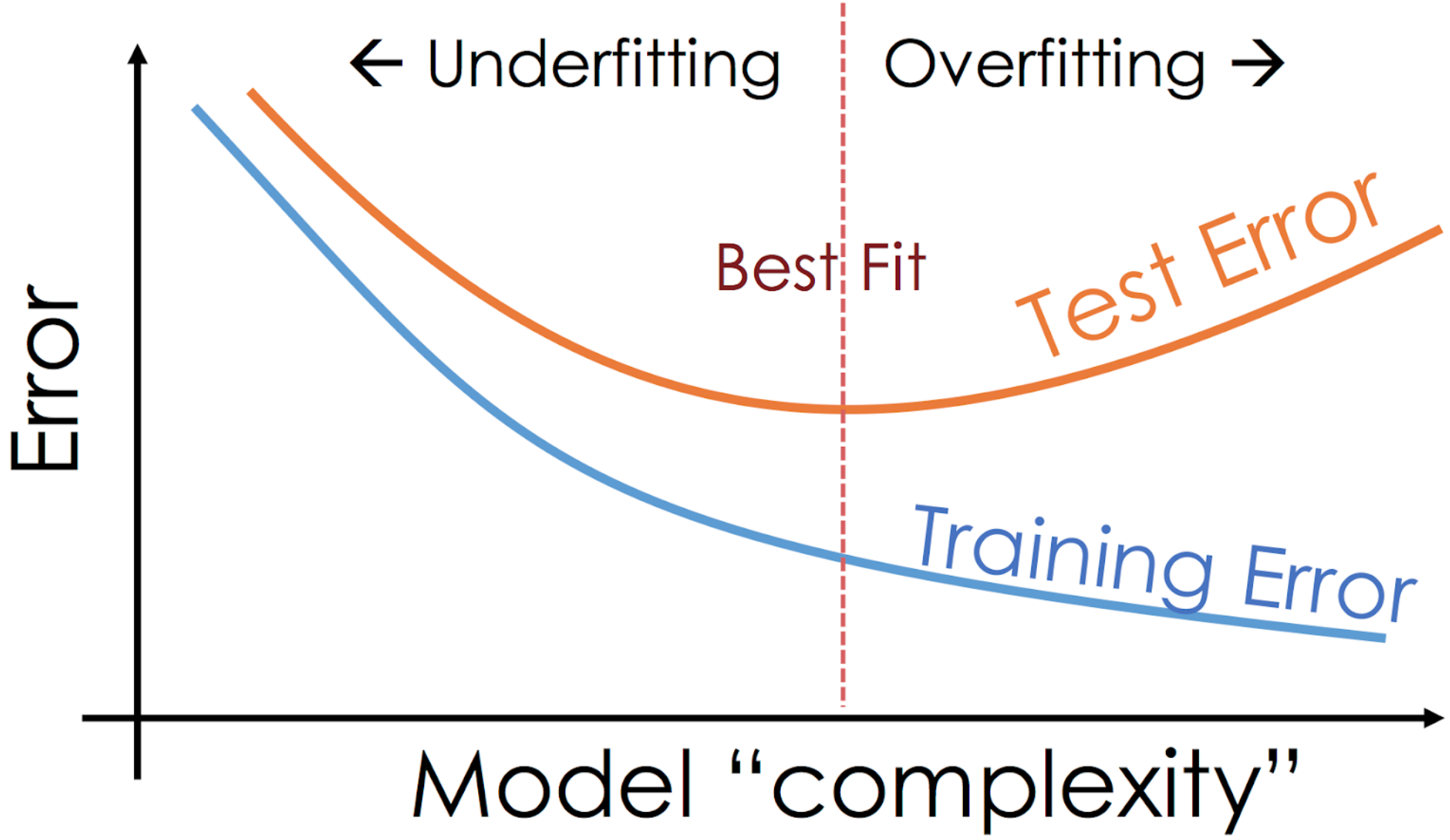
Scale **I**nvariant **F**eature **T**ransform



Error decomposition



Generalization



Overfitting- Data influence



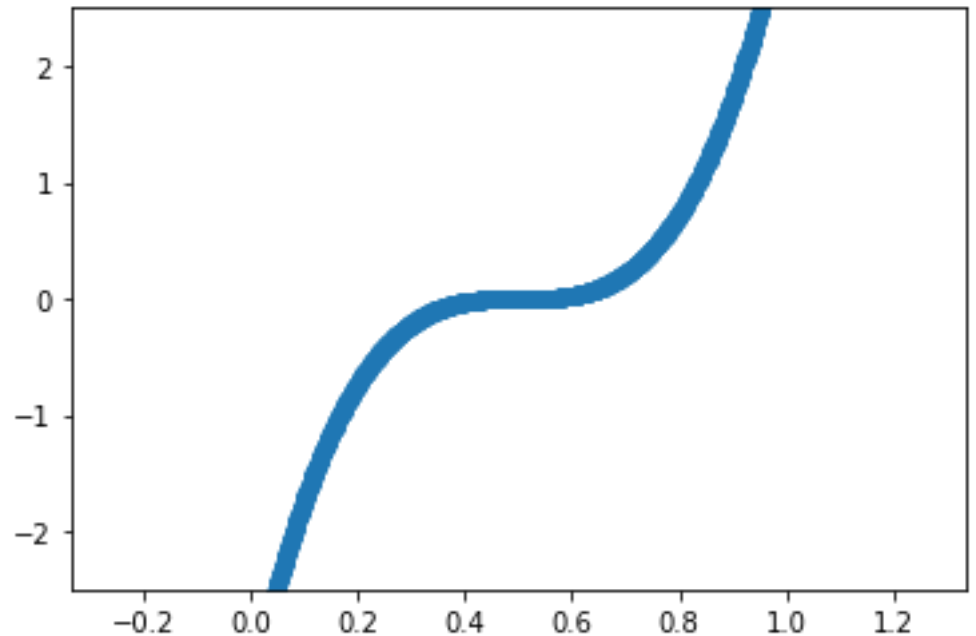
Matrix A

$$\begin{bmatrix} 3 & 4 \\ 6 & 8 \end{bmatrix}$$

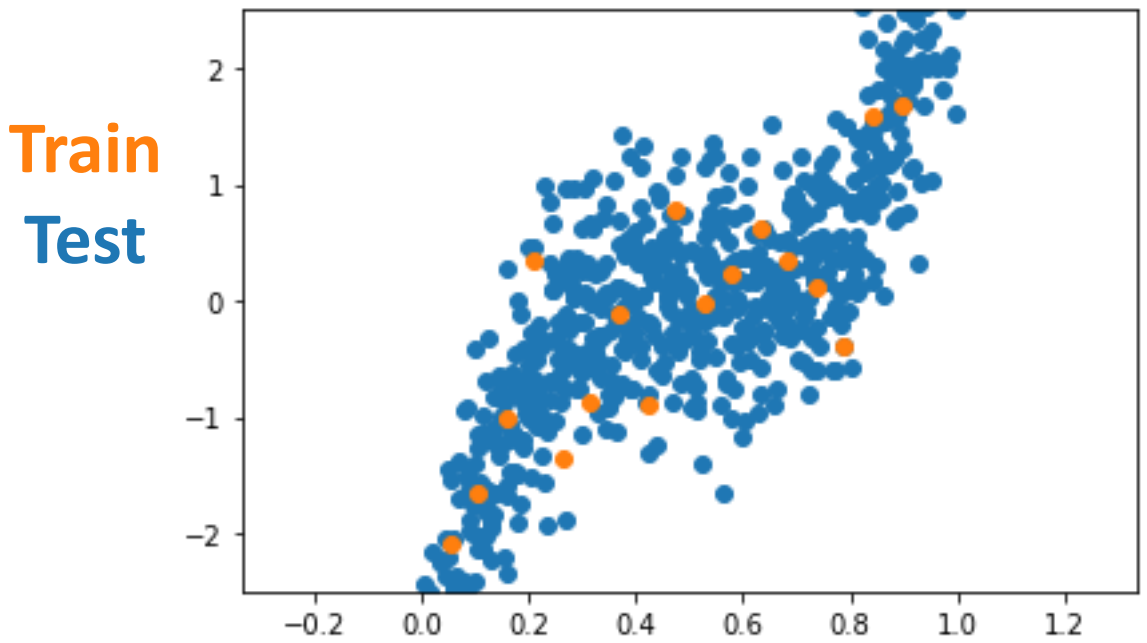
A^2

$$\begin{bmatrix} 33 & 44 \\ 66 & 88 \end{bmatrix}$$

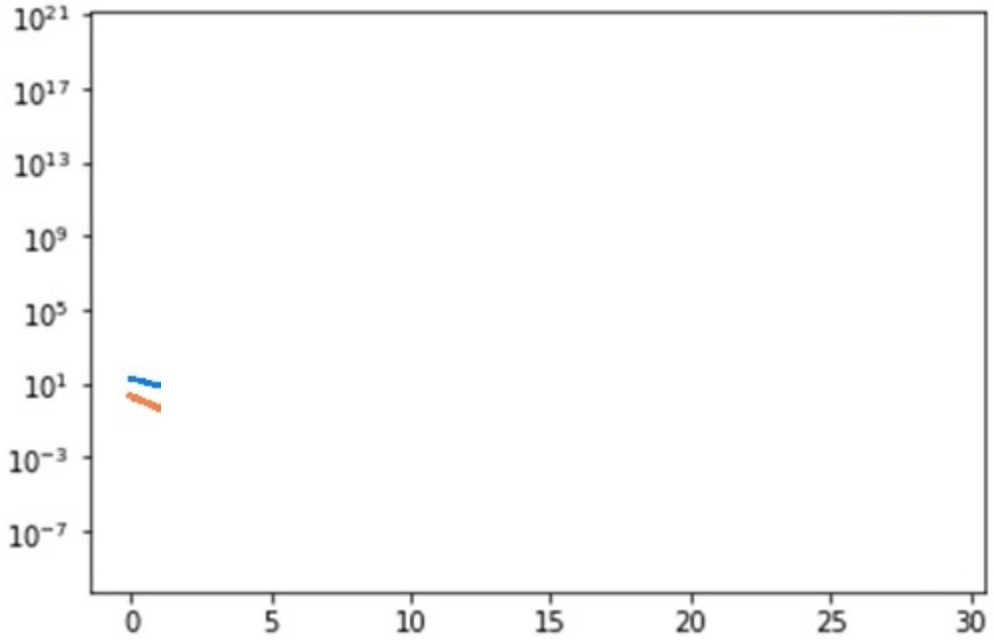
Polynomial fitting



Polynomial fitting

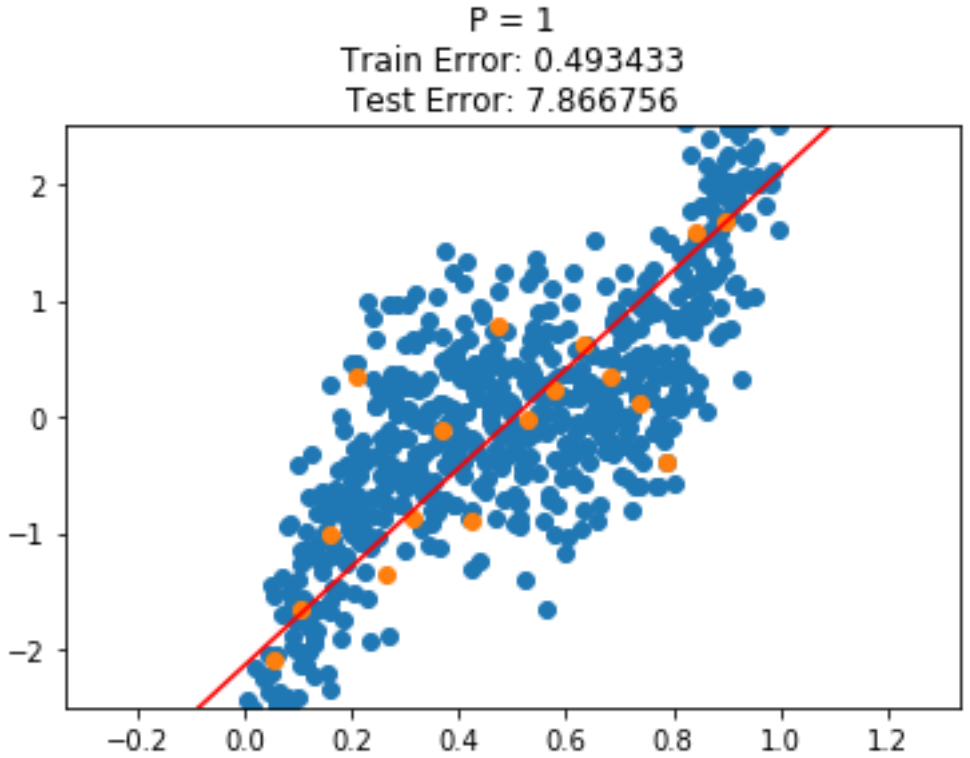


Polynomial fitting

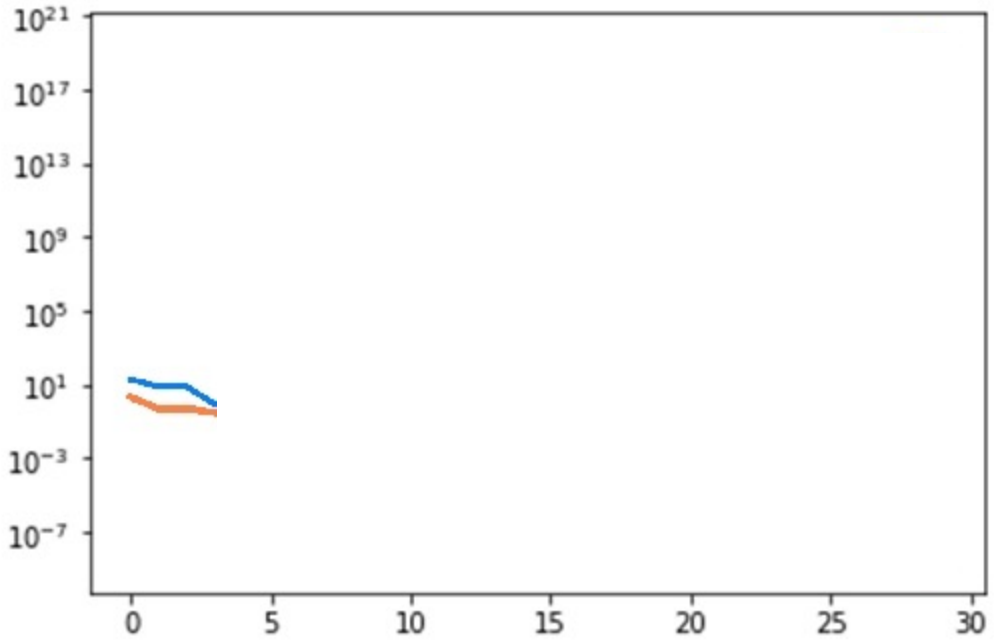


Train

Test



Polynomial fitting

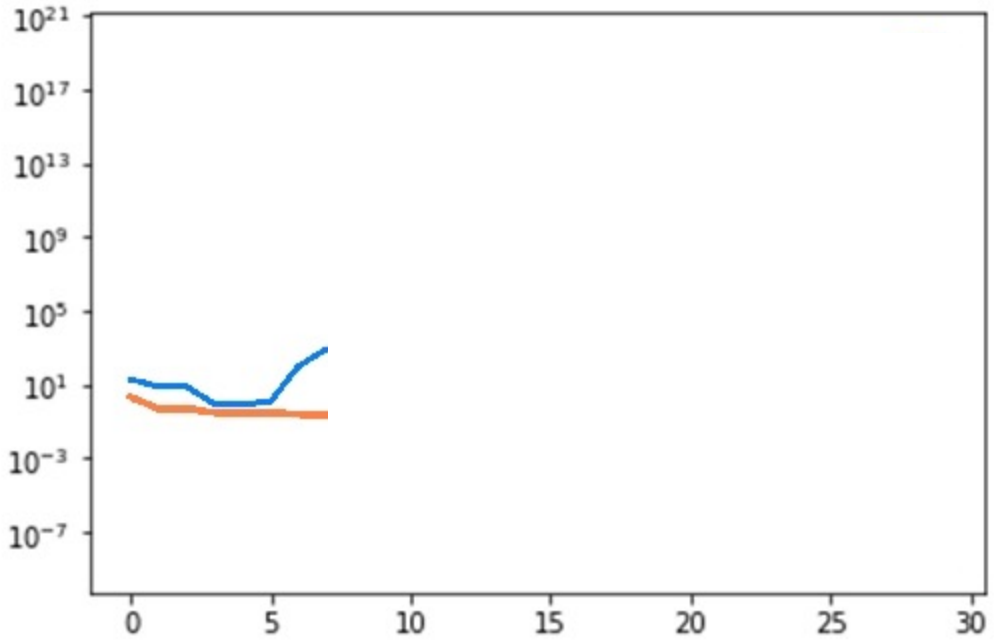


Train

Test

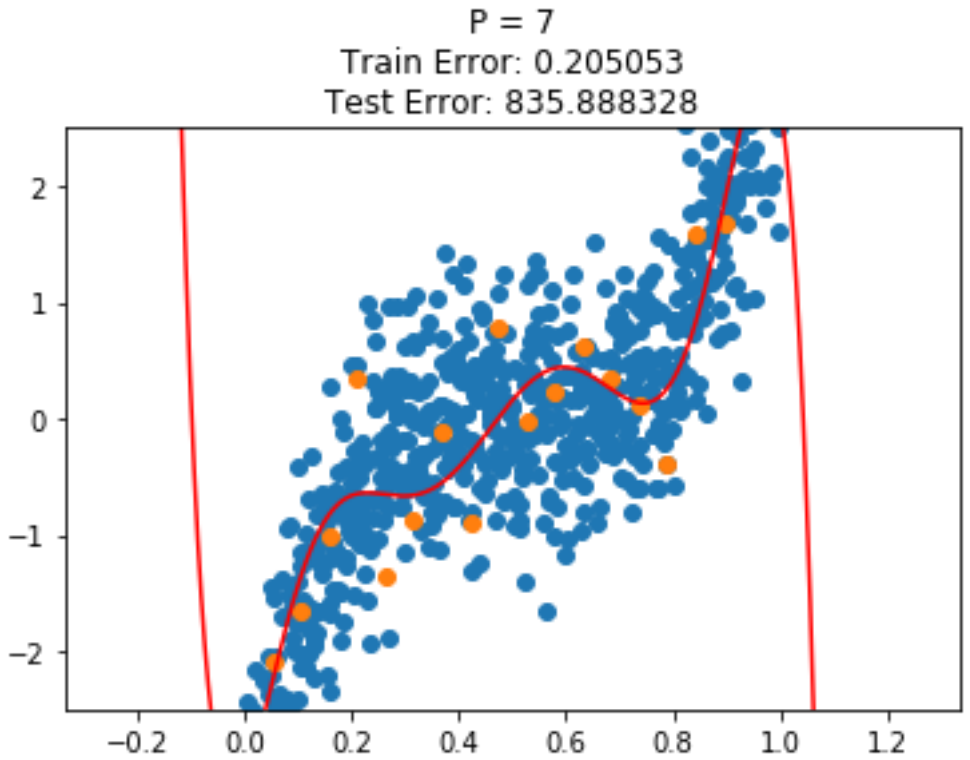


Polynomial fitting

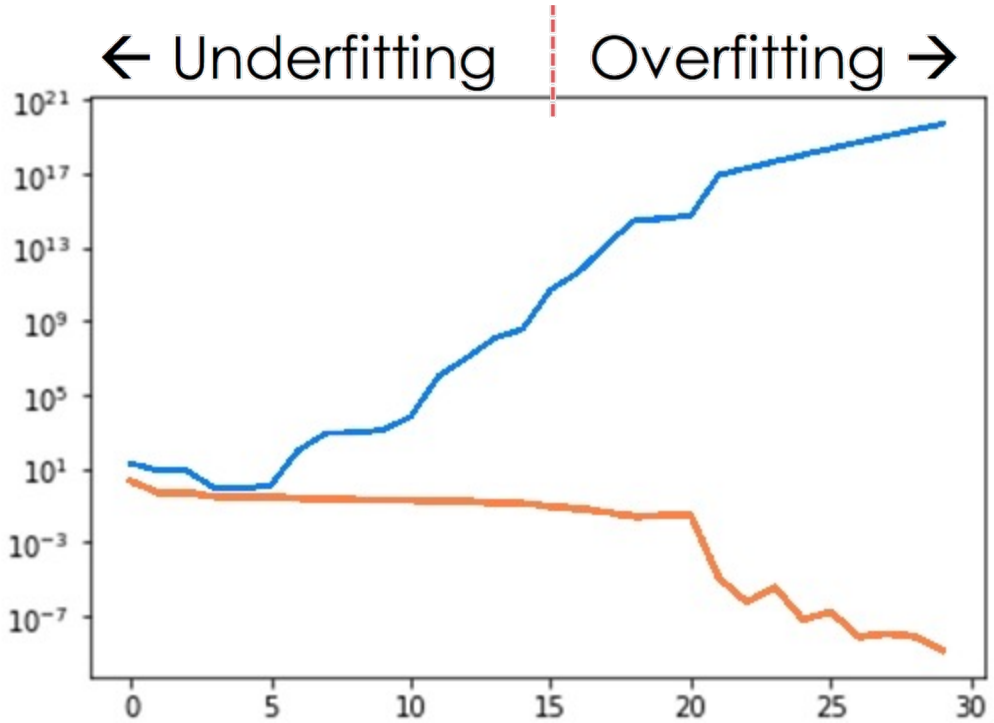


Train

Test

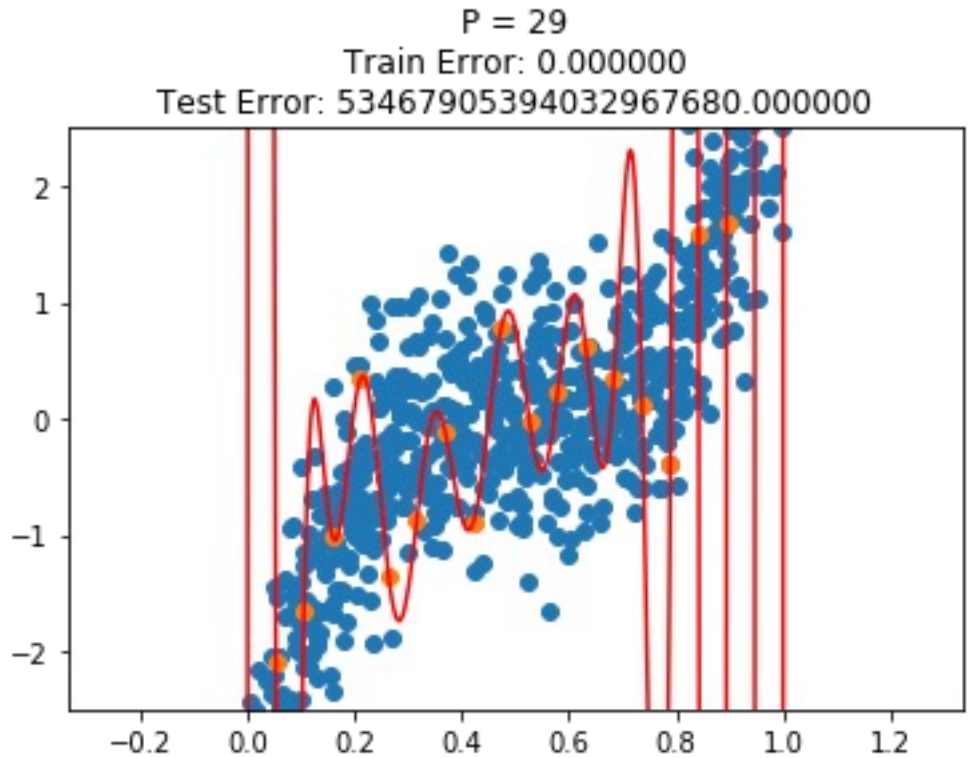


Polynomial fitting



Train

Test



Thank you!



Next week:

Neural Networks

