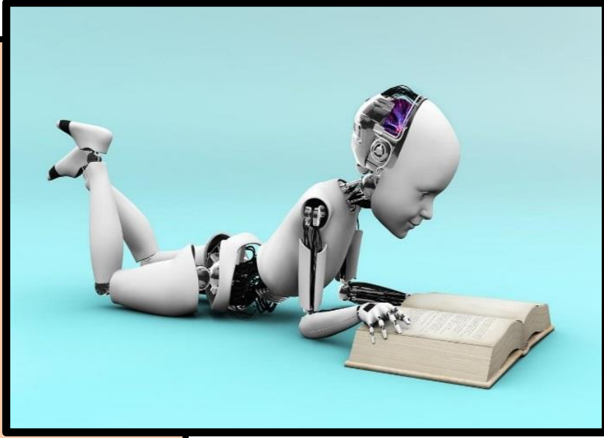
Machine learning from data
Class1: Linear Regression



Zohar Yakhini

IDC

Example: House Pricing

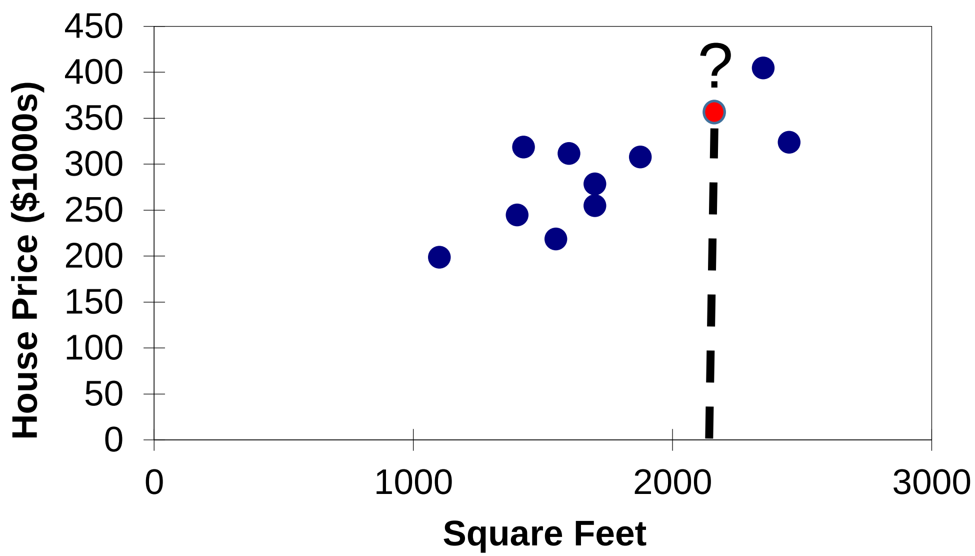
- We want to know the price of a house as a function of its size (in sqft).
- We want to learn a function from the size of the house x to the price $y = f(x)$ so we would be able to answer the above question
- Training set: 10 house <u>instances</u> with <u>feature</u> values and <u>labels</u>

Square Feet (x)	House Price in $1000s (y)
1400	245
1600	312
1700	279
1875	308
1100	199
1550	219
2350	405
2450	324
1425	319
1700	255

Statistics:
Dependent variable (y) = house price
Explaining variable (x) = square footage

Graphical Representation

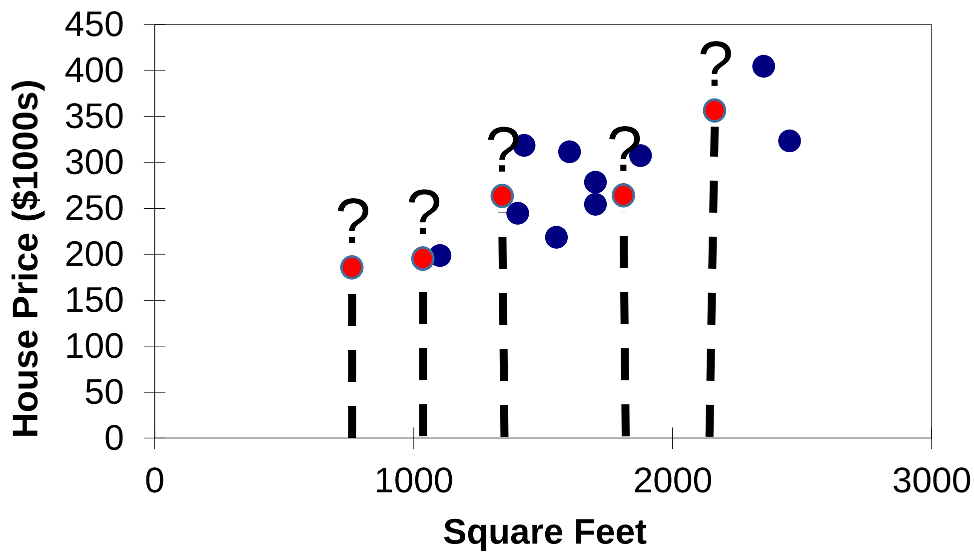
Scatter plot of House Price (y) vs House Size (x)



Prediction:

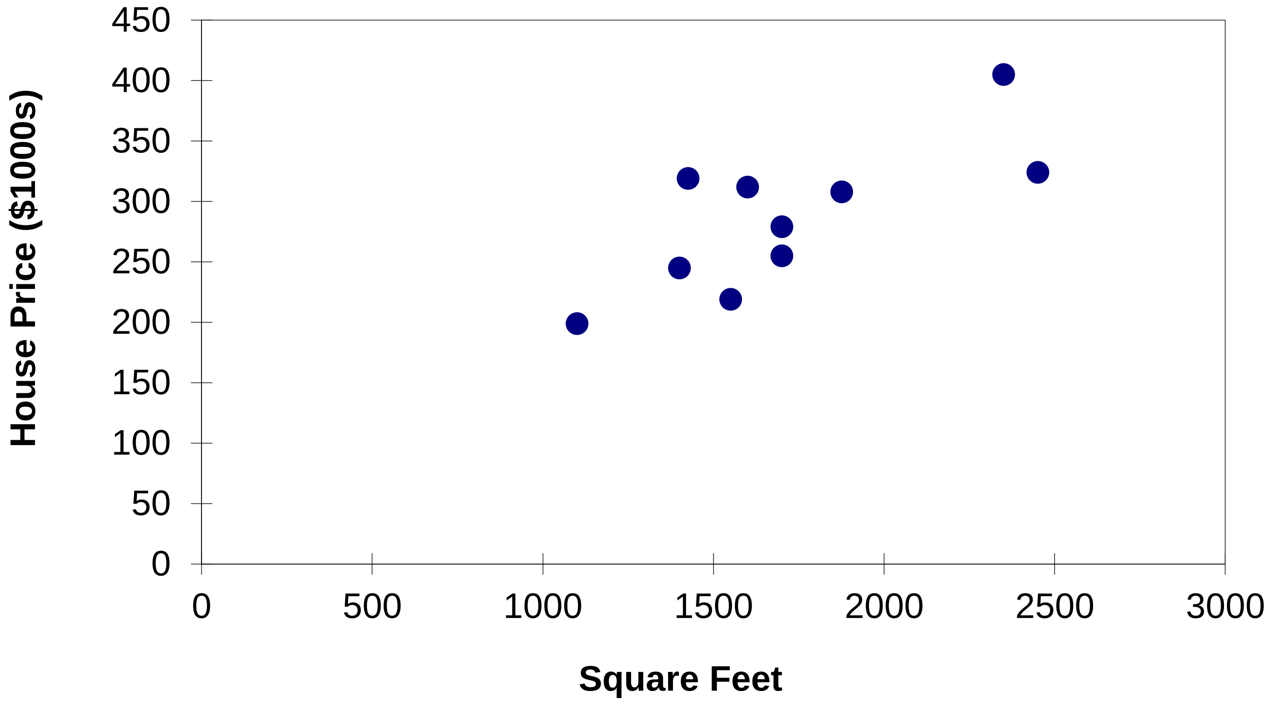
Given house of size x, what would be its price $y = f(x)$?

Memorization?

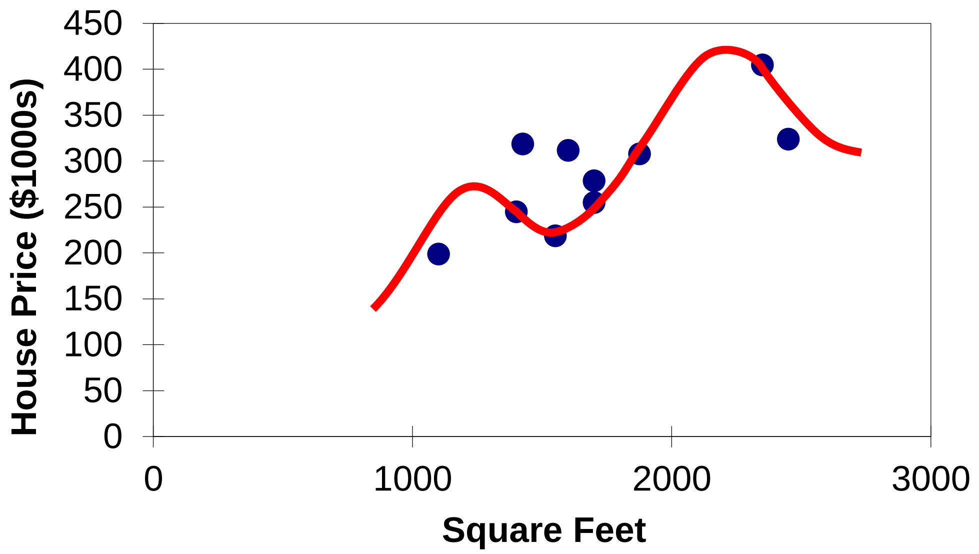


- Store all sizes?
- Our data doesn’t cover all sizes … What shall we do w a house of size 1750 sqft?

OK … a function?

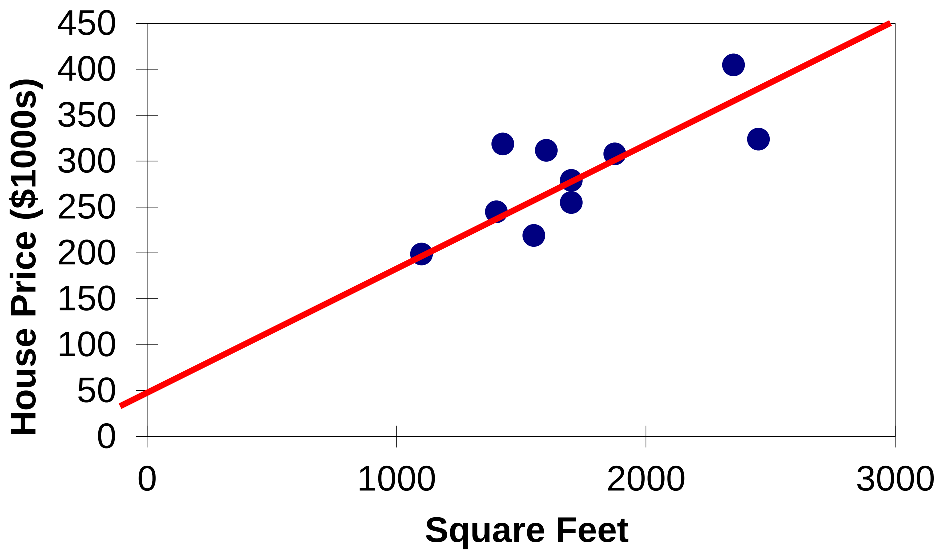


Generalization: Learn a Function



But which function y=f(x)?

Simplest: Linear Model



- We assume/hypothesize that the relationship between the observed and the independent/explained variable is linear and thereby conduct our search
- This is our **Hypotheses Space –** all linear functions

Linear Function Hypothesis

- How do we represent a hypothesis h in this space?

$$y = \theta_0 + \theta_1 x$$

- What if we have many features and not just house size?
- Such as:
  - Number of rooms
  - Distance to shopping center
  - Neighborhood crime rate
  - Distance to IDC
  - More…

Multiple Features; Higher Dimension

- Let $$\mathbf{x}^{(\mathbf{i})} = \left(\mathrm{x}_1^{(i)}, \mathrm{x}_2^{(i)}, \ldots, \mathrm{x}_n^{(i)}\right)$$
  be the vector of feature values for each instance i
- For simplicity we add another "constant" feature for every i
  $$\mathrm{x}_0^{(i)} = 1$$
- For n features our linear hypothesis will be represented by the parameters $$\boldsymbol{\theta} = \left(\theta_0, \theta_1, \theta_2, \ldots, \theta_n\right)$$
  with which we construct:
  $$\boxed{\mathrm{y} = \theta_0 + \theta_1 \mathrm{x}_1 + \theta_2 \mathrm{x}_2 + \cdots + \theta_\mathrm{n} \mathrm{x}_\mathrm{n}}$$
- We say that $$\boldsymbol{\theta} = \left(\theta_0, \theta_1, \theta_2, \ldots, \theta_n\right)$$
  is the vector of parameters that defines our function (or our **model/execution-algorithm/hypothesis**)

The Hypothesis (or model) is the execution algorithm

- How does a specific set of values,

  $$\boldsymbol{\theta} = \left(\theta_0, \theta_1, \theta_2, \ldots, \theta_n\right)$$

  which defines a specific hypothesis (model) help us?
- How can we use it?
- Given a new house instance $\mathbf{x}' = \left(\mathrm{x}'_0, \mathrm{x}'_1, \ldots, \mathrm{x}'_\mathrm{n}\right)$ we can estimate its price by an inner product with the vector $\boldsymbol{\theta}$:

$$\boxed{\text{value of house} = \mathrm{y} = \theta_0 + \theta_1 \mathrm{x}'_1 + \theta_2 \mathrm{x}'_2 + \cdots + \theta_\mathrm{n} \mathrm{x}'_\mathrm{n}}$$

Finding the Best Hypothesis/Model

- There are many possible parameter vectors

$$\boldsymbol{\theta} = \left(\theta_0, \theta_1, \theta_2, \ldots, \theta_n\right)$$

  and each one defines a different hypothesis in our hypotheses space

- How do we find the best one?
- What can we use to help us find it?

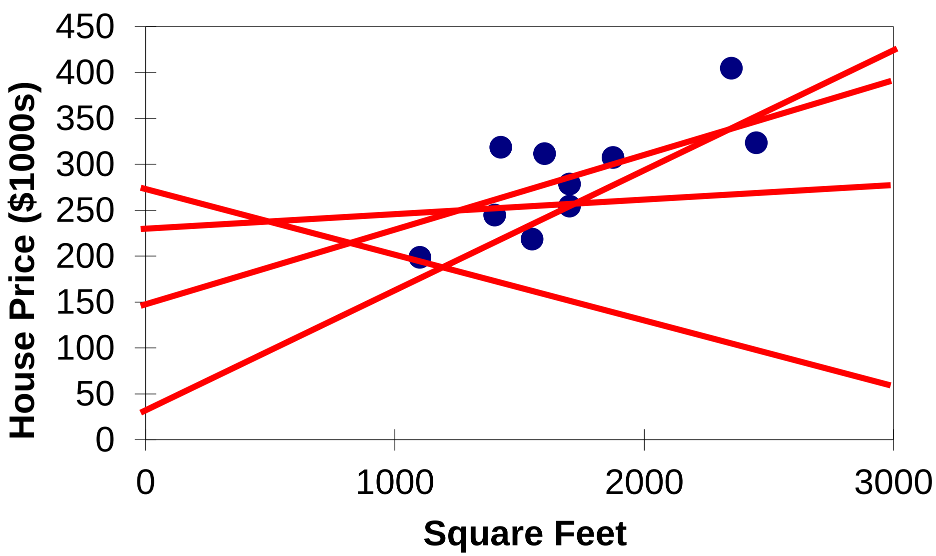
- The training set: m instances where, for each, we know the feature values $\mathbf{x}^{(\mathbf{i})} = \left(\mathrm{x}_0^{(i)}, \mathrm{x}_1^{(i)}, \ldots, \mathrm{x}_n^{(i)}\right)$

  as well as the label value $\mathrm{y}^{(i)}$

The Hypotheses Space

- For the simple case of a single feature we get different possible straight lines when we change θ_0 and θ_1 in the hypothesis (model)

$$y = \theta_0 + \theta_1 x$$

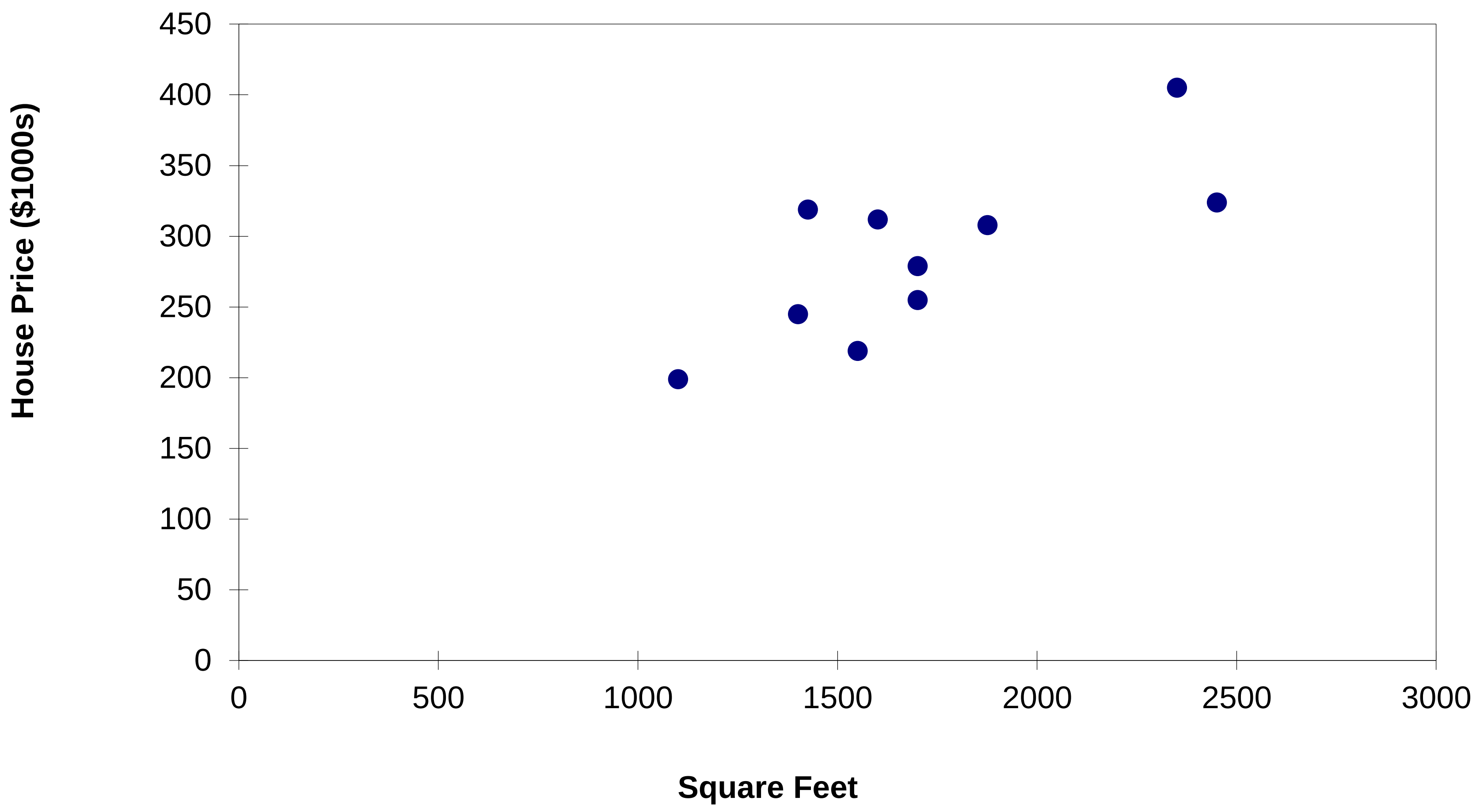


"Training" or "Learning". ERM

- We require that on our training data the values of our prediction function (f) would be similar to the known value of the house.
- So, we want to find $\boldsymbol{\theta}$ such that for all instances i in the training set we will have:

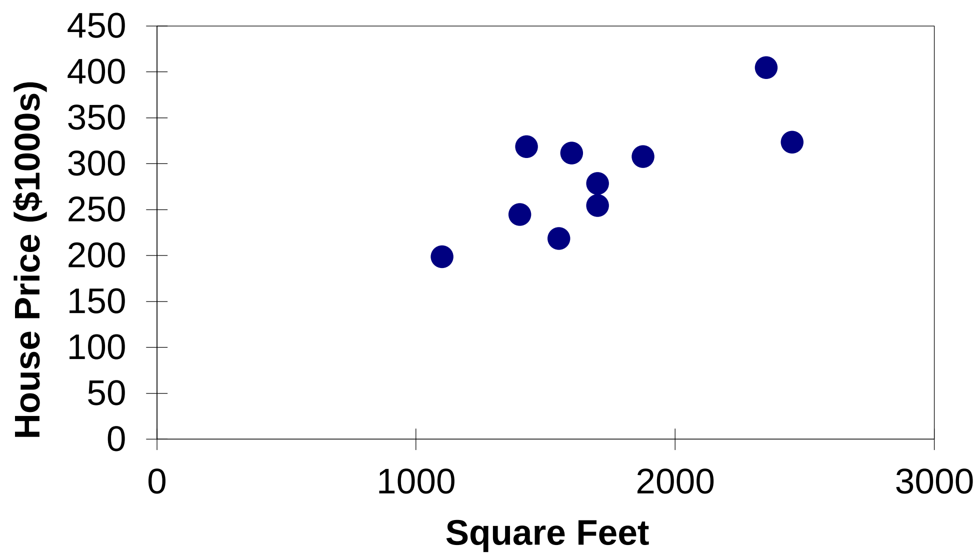
$$\mathrm{y}^{(\mathrm{i})} = \theta_0 \mathrm{x}_0^{(\mathrm{i})} + \theta_1 \mathrm{x}_1^{(\mathrm{i})} + \theta_2 \mathrm{x}_2^{(\mathrm{i})} + \cdots + \theta_\mathrm{n} \mathrm{x}_\mathrm{n}^{(\mathrm{i})} = \boldsymbol{\theta} \cdot \mathbf{x}^{(\mathbf{i})}$$

- However, this may not always be possible – (why?)


House Price ($1000s)
450
400
350
300
250
200
150
100
50
0
0
500
1000
1500
2000
2500
3000
Square Feet

Consistent Learners

- A learning algorithm that can achieve 0 error on the training set is called a "consistent learner"
- This can not be done here.



Cost Function of a Model θ

- As consistent learning may be impossible.
- Still, we can try to reduce the error.
  Per instance the error is:

Prediction

Actual value

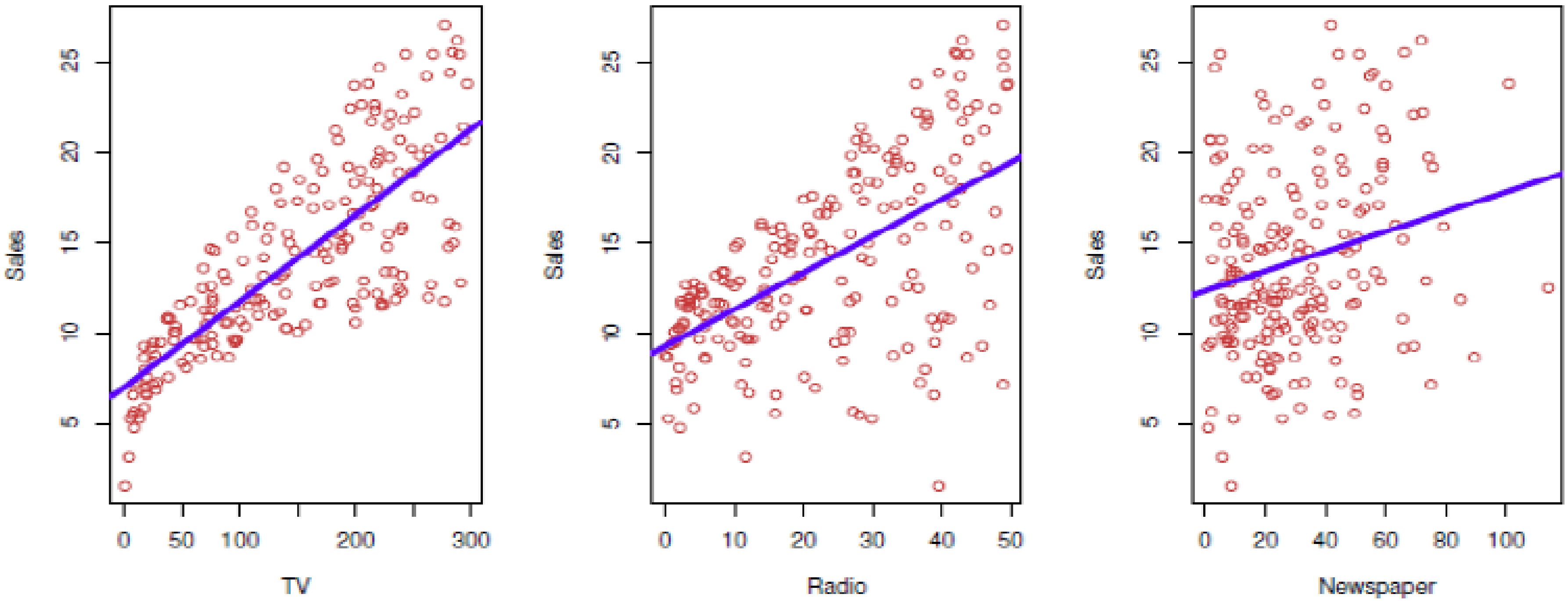
$$(\theta_0 x_0^{(i)} + \theta_1 x_1^{(i)} + \theta_2 x_2^{(i)} + \cdots + \theta_n x_n^{(i)} - y^{(i)}) = \boxed{\boldsymbol{\theta} \cdot \mathbf{x}^{(\mathbf{i})}} - \boxed{y^{(i)}}$$

- And we now average on ALL m training instances to get our
  cost function:

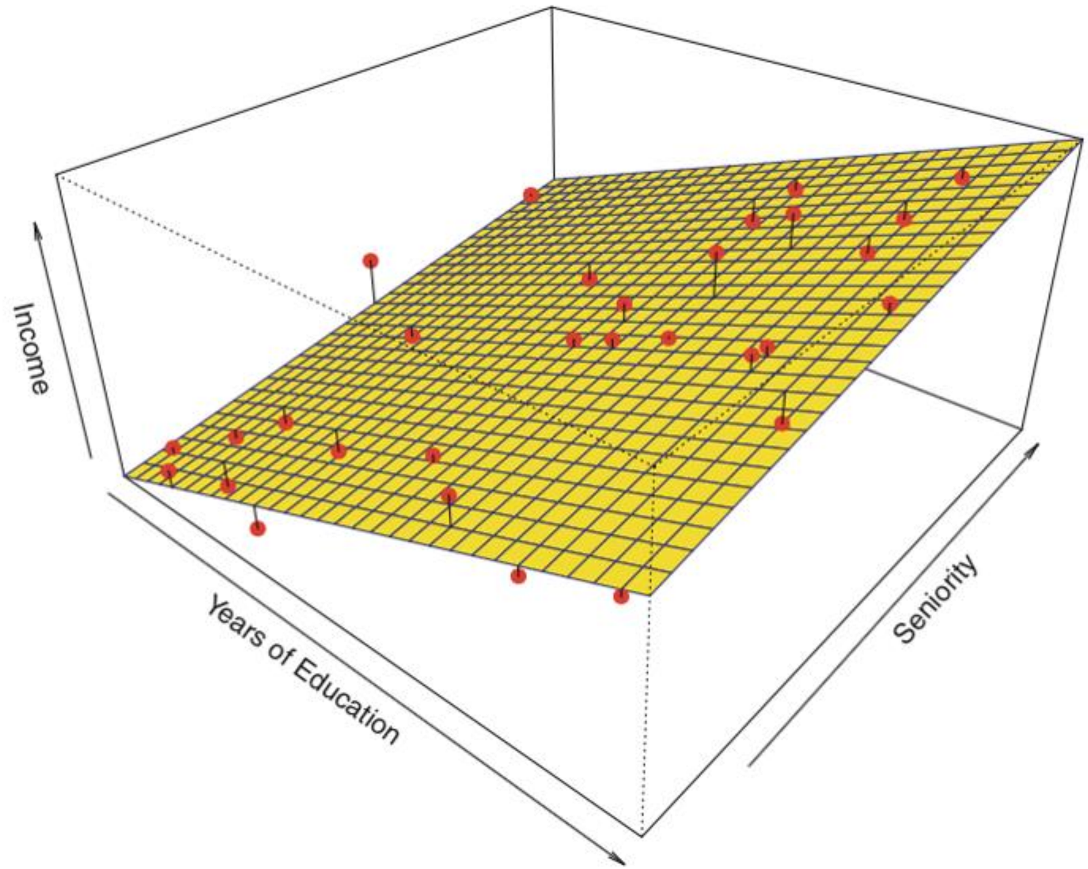
$$J(\boldsymbol{\theta}) = \frac{1}{2}\frac{1}{m}\sum_{i=1}^{m}\left(\boldsymbol{\theta} \cdot \mathbf{x}^{(\mathbf{i})} - y^{(i)}\right)^2$$

- Square errors are used so that errors in different directions don't cancel out …
  Its also a smoother function compared to $|x|$

Example: advertising and sales



In higher dimensions



Minimizing the Cost Function

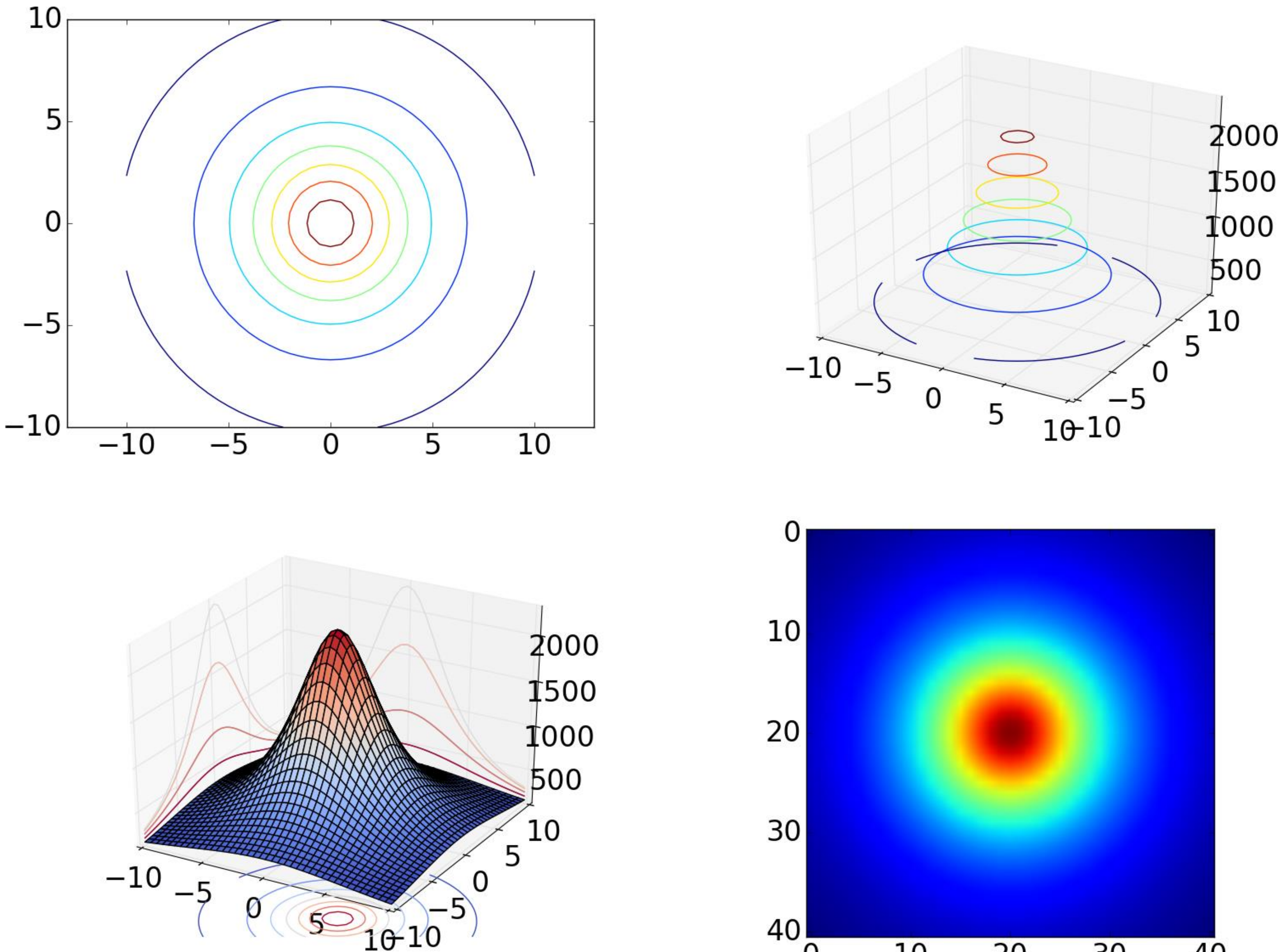
- In the ERM approach, our best hypothesis $\boldsymbol{\theta}^*$ would be the one that <u>minimizes the cost function</u>. Formally:

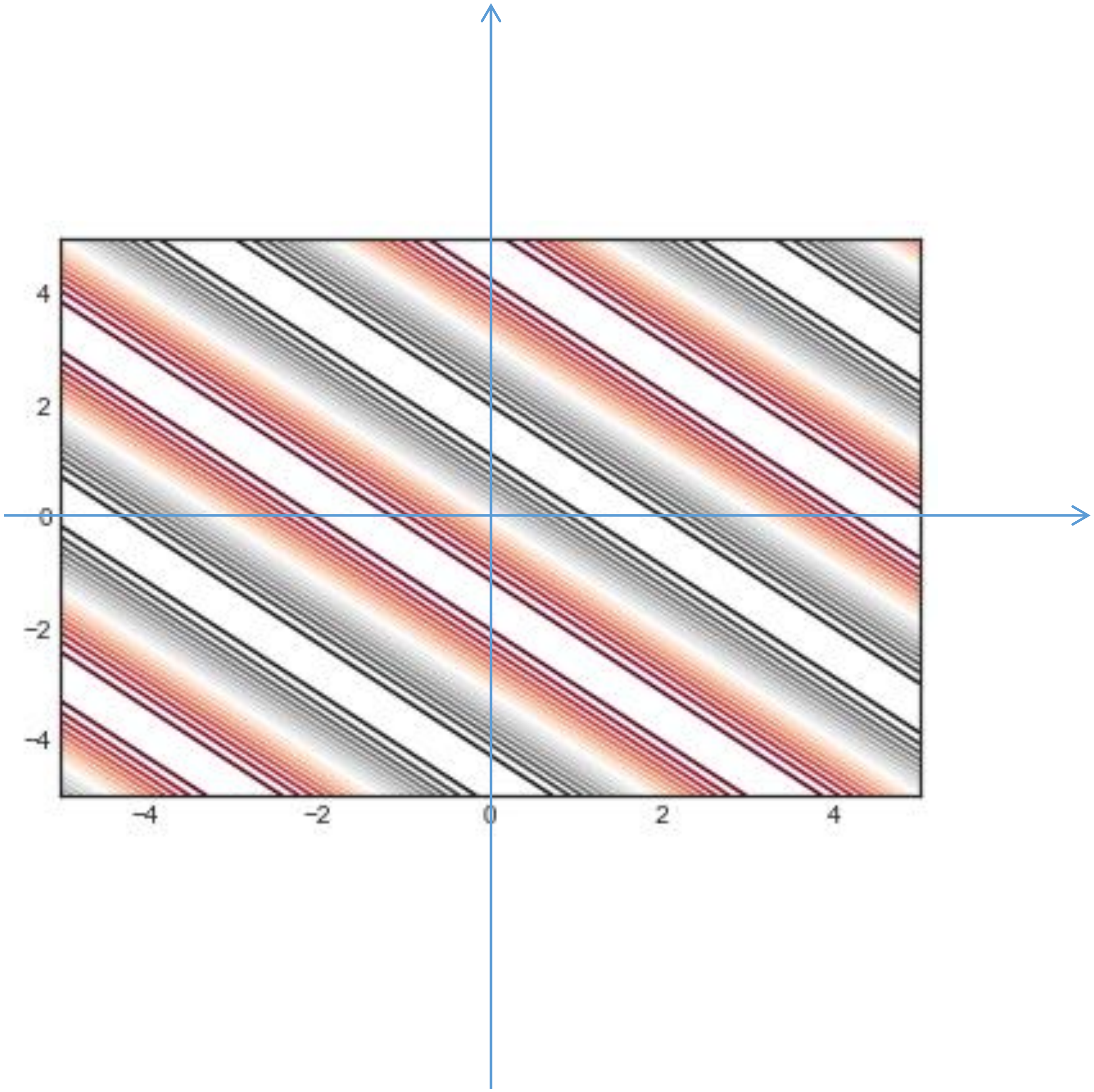
$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \left[\mathrm{J}(\boldsymbol{\theta}) \right] = \arg\min_{\boldsymbol{\theta}} \left[\frac{1}{2m} \sum_{i=1}^{m} \left(\boldsymbol{\theta} \cdot \mathbf{x}^{(\mathbf{i})} - \mathrm{y}^{(\mathrm{i})} \right)^2 \right]$$

- How can we find it?

The Cost Function

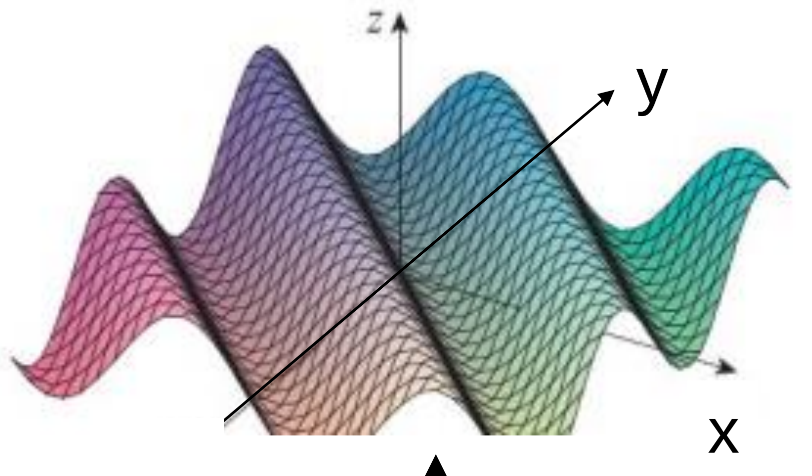
- In our simple case (house prices) we have 2 parameters: θ_0, θ_1
- For each value of these two parameters we can calculate the cost $J(\theta_0, \theta_1)$ over the entire training data (the training error).
- This is a function from $\mathbb{R}^2$ to $\mathbb{R}$.


10
5
0
−5
−10
−10
−5
0
5
10
2000
1500
1000
500
10
5
0
−5
−10
−10
−5
0
5
10
2000
1500
1000
500
10
5
0
−5
−10
−10
−5
0
5
10
0
10
20
30
40
0
10
20
30
40

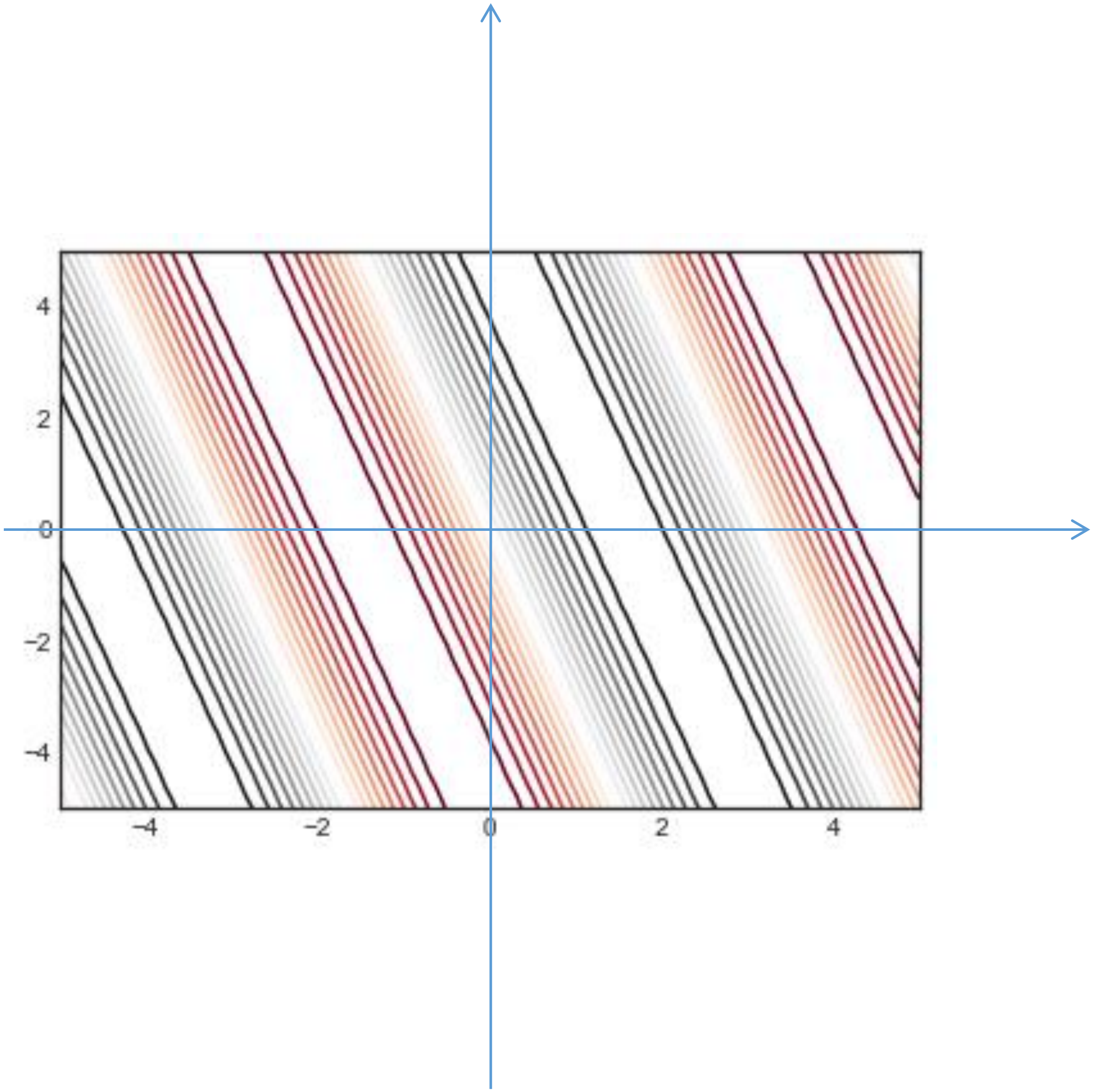

4
2
0
−2
−4
−4
−2
0
2
4

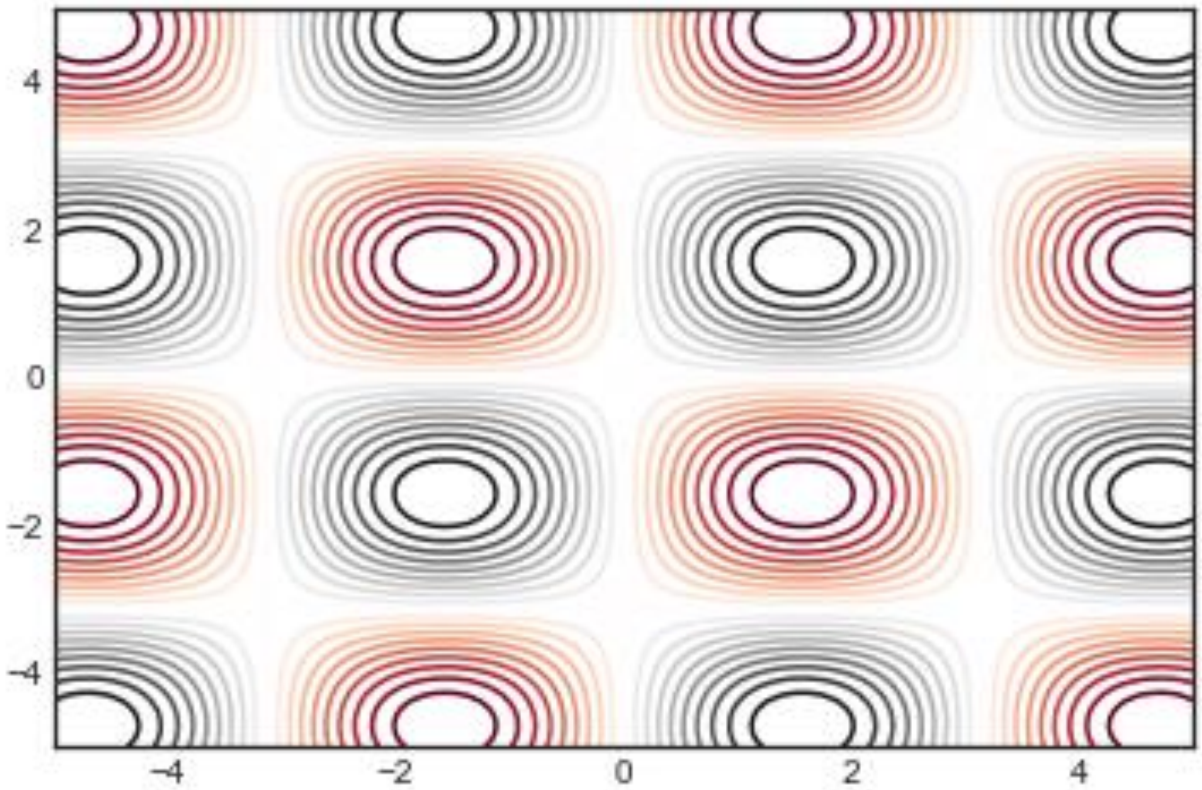
Code to generate the contour plot

```
def f(x, y):
    return np.sin(x+y)

x = np.linspace(-5, 5, 50)
y = np.linspace(-5, 5, 50)

mx, my = np.meshgrid(x, y)
z = f(mx, my)

plt.contour(mx, my, z, 20, cmap='RdGy');
```

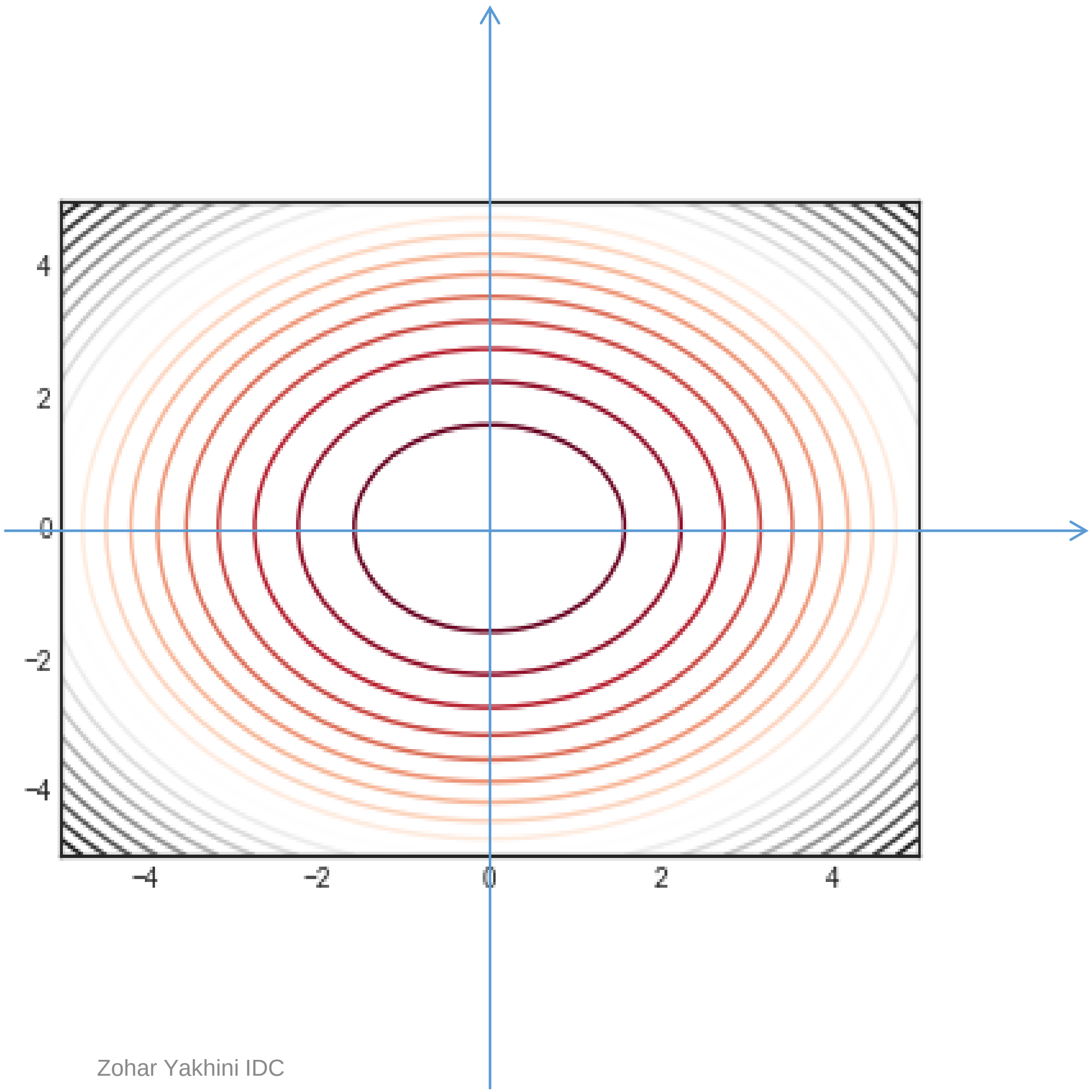


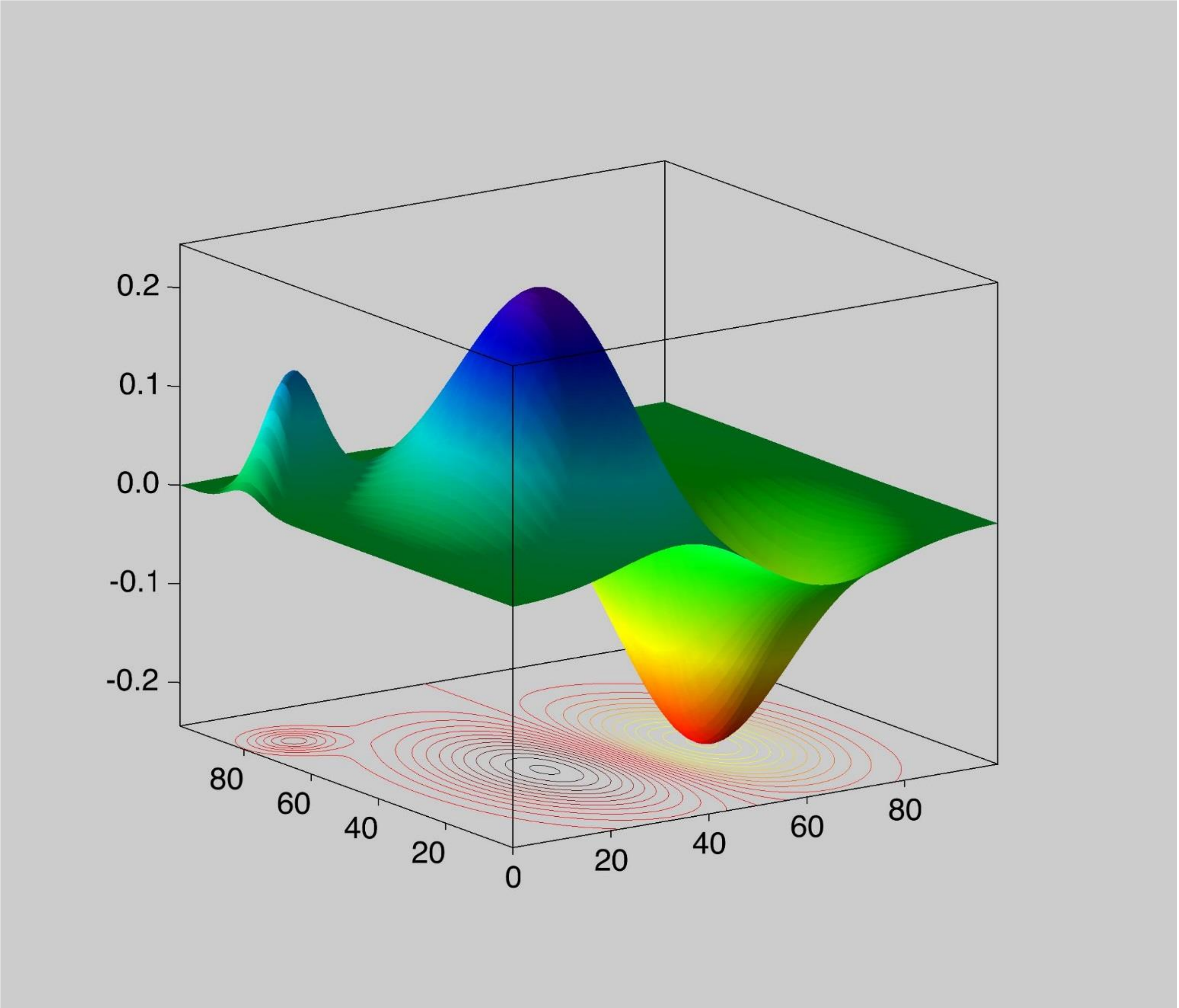
3D plot.
Not a contour plot


4
2
0
−2
−4
−4
−2
0
2
4


4
2
0
−2
−4
−4
−2
0
2
4

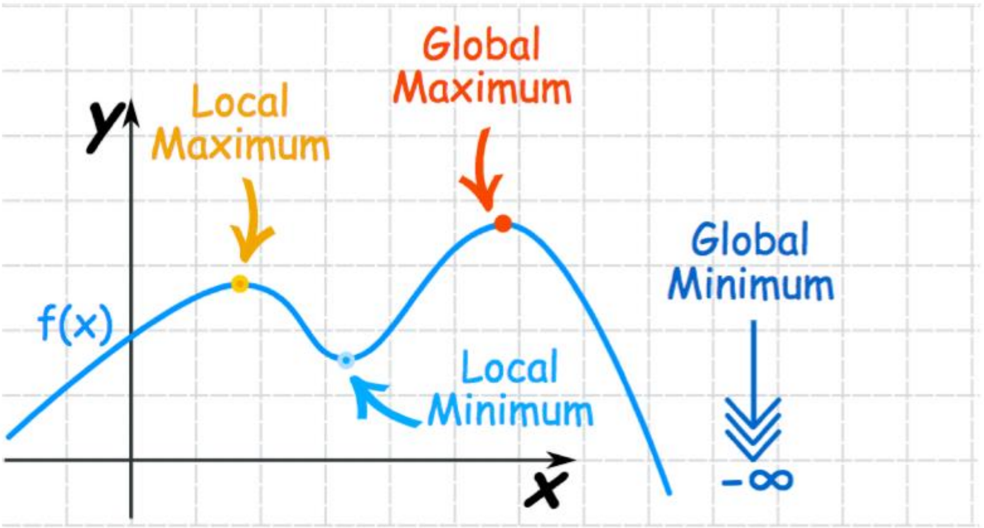
Recall:
We want to find the argmin of the cost function $J(\theta)$




0.2
0.1
0.0
-0.1
-0.2
80
60
40
20
0
20
40
60
80

How to Find the Minimum of a 1D function?

- In high school:
  derivative at the point where an extremum is attained should be 0
- We can also follow the “downward” direction.
- How is this done?
  1. Find the derivative
  2. Move against its sign



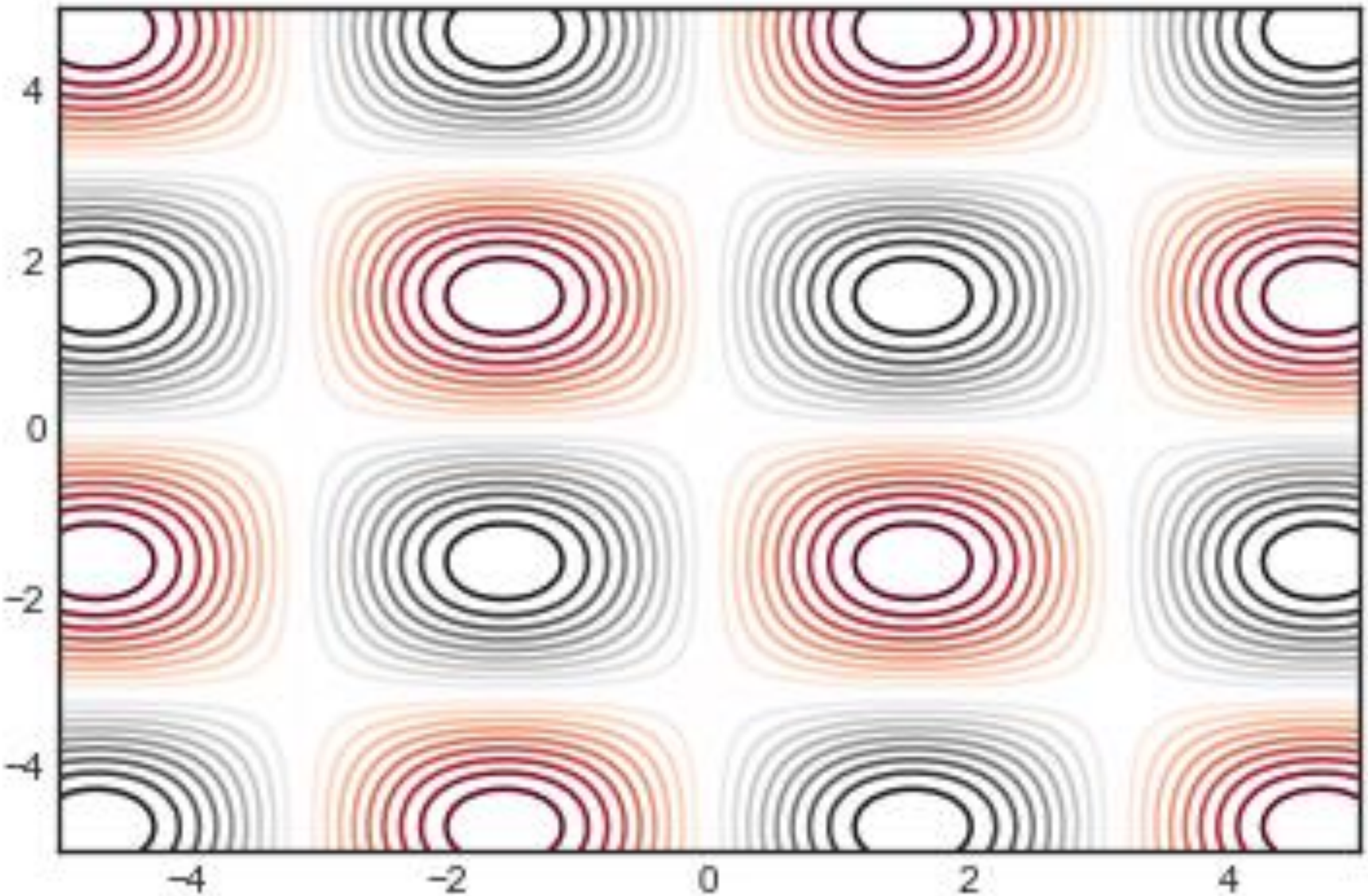
Directional Derivatives

Consider a differentiable 2D function $f(x_1, x_2)$ The derivative in a <u>general direction</u> $u = (u_1, u_2)$ (unit 2D vector) is called the directional derivative $D_u f$ and is defined as:

$$D_u f(x_1, x_2) = \lim_{s \to 0} \frac{f(x_1 + su_1, x_2 + su_2) - f(x_1, x_2)}{s} = \left(\frac{df}{ds}\right)_u$$

For a differentiable 2D function $f(x_1, x_2)$ the principal partial derivatives, those in direction $u = x_i$ are denoted

$$\frac{\partial f}{\partial x_1}(x_1, x_2) \text{ and } \frac{\partial f}{\partial x_2}(x_1, x_2)$$



4
2
0
−2
−4
−4
−2
0
2
4

Partial Derivatives

- Again, for a differentiable 2D function $f(x, y)$, the (principal) partial derivatives in the directions x and y are denoted

$$\frac{\partial f}{\partial x}(x, y)\ , \frac{\partial f}{\partial y}(x, y)$$

- They can be computed by keeping one variable constant and differentiating by the other.
  For example …

$$f(x, y) \;=\; 2x \;+\; 13y$$
$$f(x, y) \;=\; y\exp(x)$$
$$f(x, y) \;=\; 2x \;+\; 3xy \;+\; 5y^2$$

https://mathinsight.org/directional_derivative_gradient_introduction

$$f(x, y) = y\, e^{x}$$

$$\frac{\partial f}{\partial x}(x, y) = ye^{x}$$

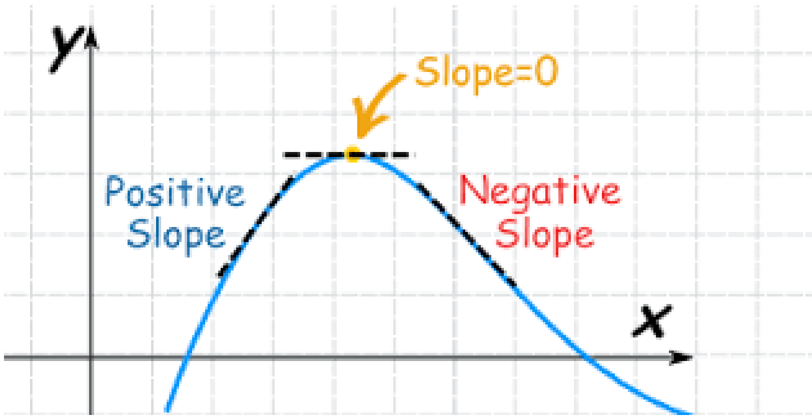
$$\frac{\partial f}{\partial y}(x, y) = e^{x}$$

$$f(x,y) \;=\; y^4\, e^x$$

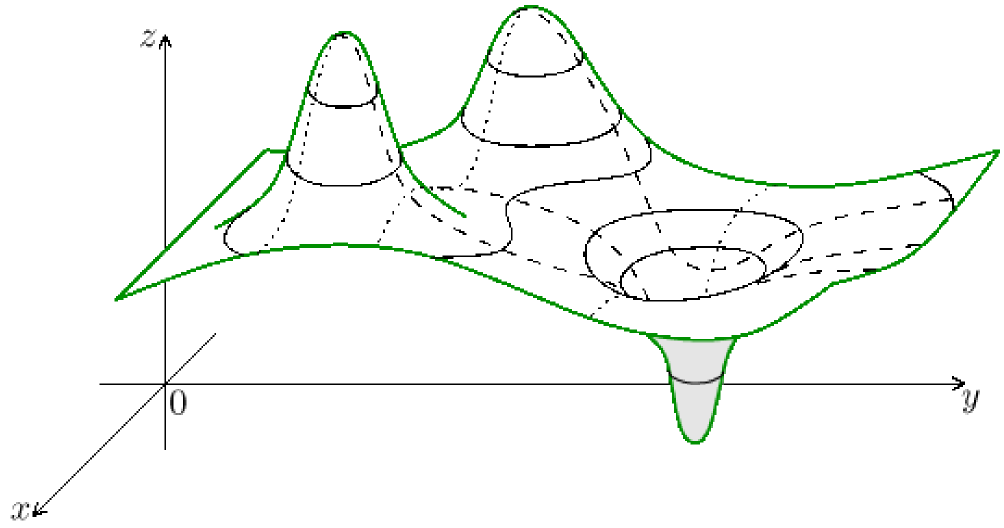
$$\frac{\partial f}{\partial x}\,(x,y) = y^4\, e^x$$

$$\frac{\partial f}{\partial y}\,(x,y) = 4y^3 e^x$$

Extrema and zero derivatives



1D



2D

The Gradient of a function

Define the
GRADIENT of f:

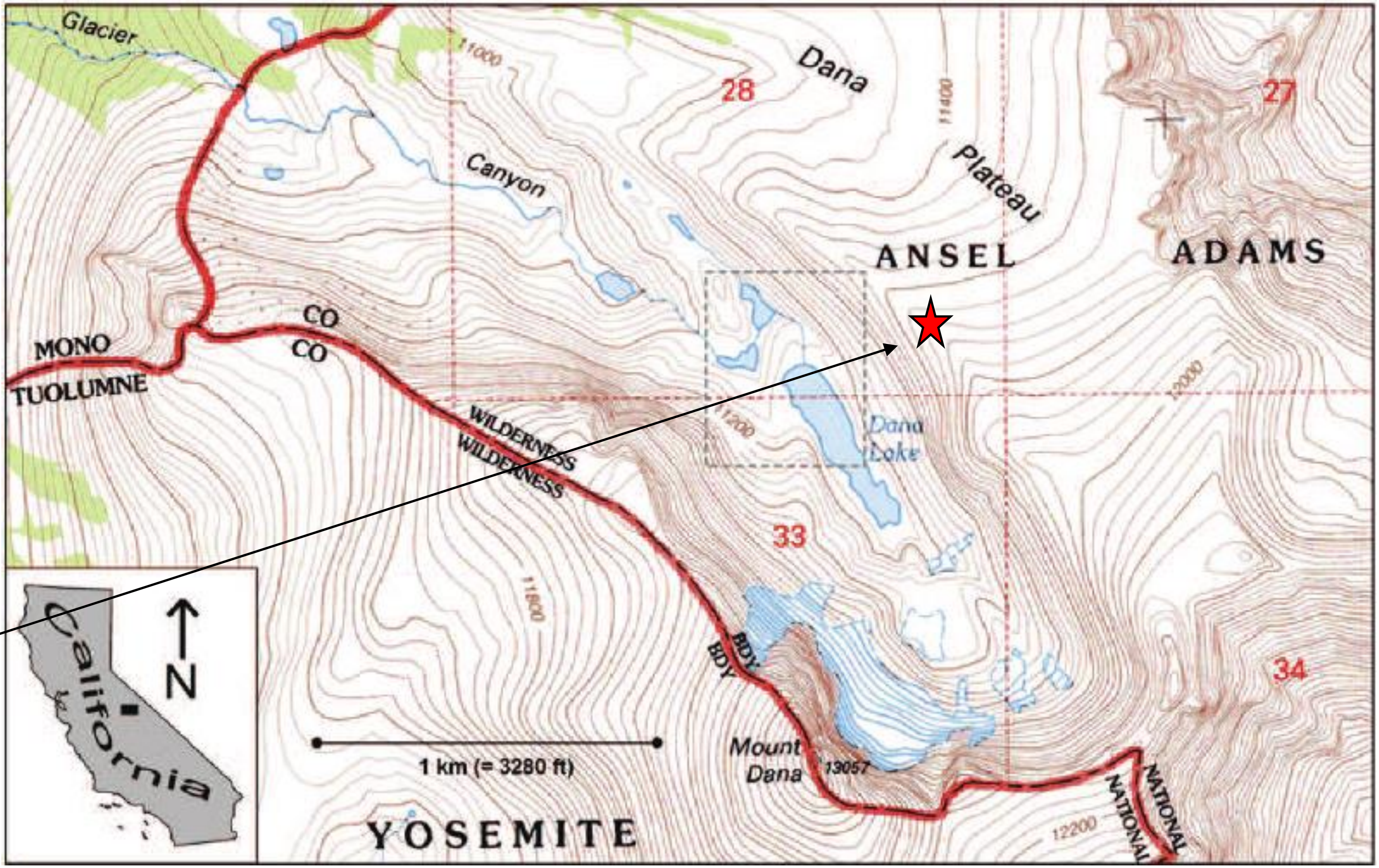
$$\nabla f = \left(\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2} \right)$$

Thm:
For any direction $u = (u_1, u_2)$ and any point $\vec{x} = (x_1, x_2)$ we have:

$$\boxed{D_u f(\vec{x}) = \langle \nabla f(\vec{x}), u \rangle}$$

IDC HERZLIYA

Most Rapid Increase at a point $\vec{x}$?



The pt $\vec{x}$

Most Rapid Increase at a point $\vec{x}$

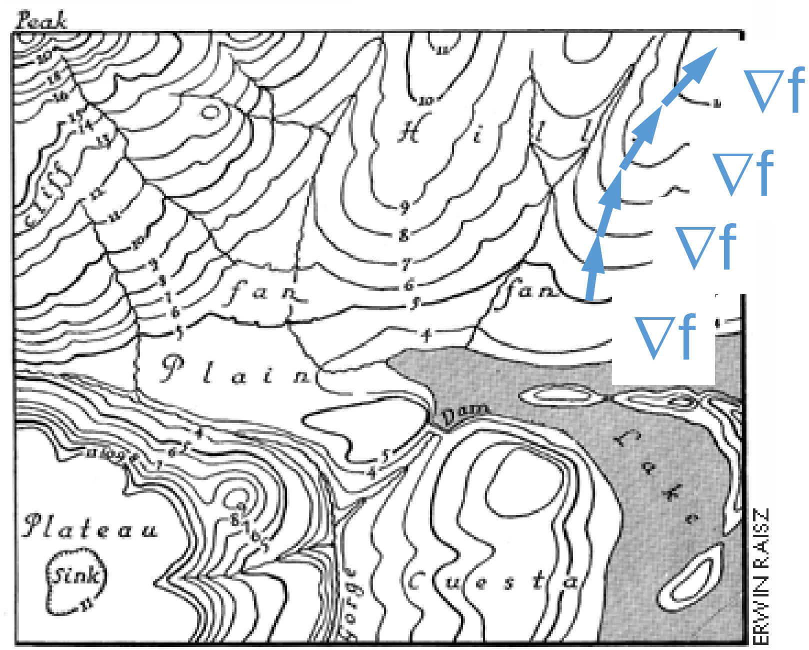
- The directional derivative in the direction of the vector $u = (u_1, u_2)$ (a scalar!) can also be written as:

$$D_u f(x_1, x_2) = \nabla f \cdot \vec{\mathbf{u}} = \left|\nabla f\right|\left|\vec{\mathbf{u}}\right|\cos\beta = \left|\nabla f\right|\cos\beta$$

- Where β is the angle between $\boldsymbol{u}$ and ∇f
- However, $\cos\beta \leq 1$.
- Therefore:
  1. The greatest increase in the function happens in the direction of the gradient (i.e. β =0)
  2. The greatest decrease is in the direction $-\nabla f$ (i.e β =180°)

Steepest ascent and Iso-Contours

- Steepest ascent follows the gradient direction
- Using same argument, when the direction u is perpendicular to the gradient, $\cos\beta = 0$ and there is no change in the function – this is exactly the direction of the iso-contours, walking with no change in altitude.



IDC HERZLIYA

The gradient of n-dimensional functions

- For an n-D function $y = f(x_1, x_2, \dots, x_n)$ the gradient is defined as :

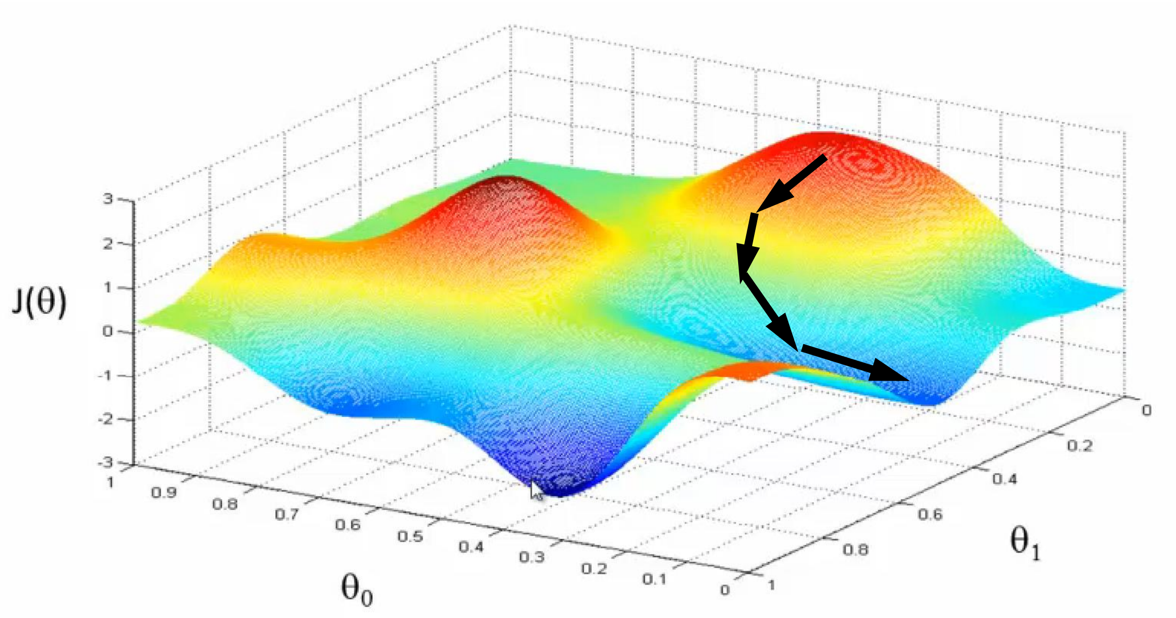
$$\nabla f = \left(\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \dots, \frac{\partial f}{\partial x_n} \right)$$

- The gradient vector at any given point is the direction of greatest increase of f at this point.
- The direction of greatest decrease of f is opposite to the gradient: $-\nabla f(x_1, x_2 \dots x_n)$

IDC HERZLIYA

How to find a minimum of a (reasonably smooth) n-dimensional function?

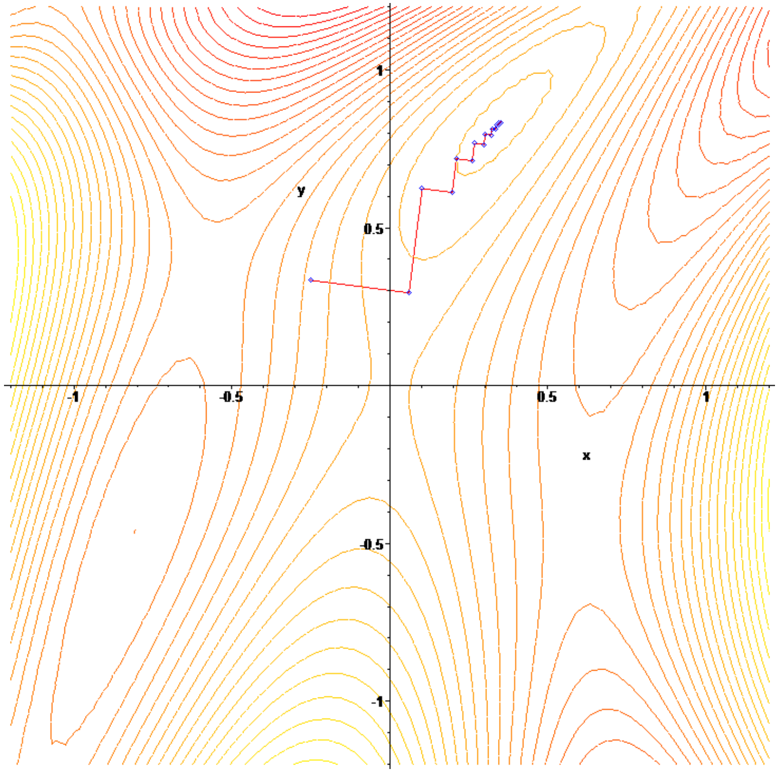
- Take steps in the direction of maximum decrease – i.e. opposite the gradient direction!
- This is called **Gradient Descent**



IDC HERZLIYA

Gradient Descent Steps

- For a small enough $\alpha > 0$ we have $f(x - \alpha\nabla f(x)) < f(x)$, so we take one step of size α in the opposite direction of the gradient.
- This represents greedy local descent
- Note:
  such steps can zig-zag



IDC HERZLIYA

Back to Minimizing the Cost Function in Linear Regression

- Recall that our best model, θ^* , is the one that minimizes the cost function. That is:

$$\boldsymbol{\theta}^* \ \textit{attains} \ \min_{\boldsymbol{\theta}}[\mathrm{J}(\boldsymbol{\theta})] = \min_{\boldsymbol{\theta}}\left[\frac{1}{2m}\sum_{i=1}^{m}\left(\boldsymbol{\theta}\cdot\mathbf{x}^{(\mathbf{i})} - \mathrm{y}^{(\mathrm{i})}\right)^2\right]$$

- $J(\vec{\theta})$ is a real valued function of $\vec{\boldsymbol{\theta}} \in \mathbb{R}^n$
- So - we can start with some initial guess $\overrightarrow{\boldsymbol{\theta}_{\mathbf{0}}^{*}}$ and then use gradient descent

Gradient Descent Algorithm

- Start with some value $\theta(0) = (\theta_0, \theta_1, \theta_2, \dots, \theta_n)$
- Repeat until you reach a minimum:

  For all $j = 0 \dots n$,

  Update $\theta_j\,(t+1) \leftarrow\ \theta_j(t) - \alpha \frac{\partial}{\partial \theta_j} J(\theta(t))$
- $\alpha > 0$ is a parameter of the algorithm called the <u>learning rate</u>
- Updates are simultaneous (in all $n+1$ directions)
- In the general case this process can still be trapped in local minima!

IDC HERZLIYA

Minimizing the Cost Function

- This gradient descent process is seeking:

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \left[\mathrm{J}(\boldsymbol{\theta}) \right] = \arg\min_{\boldsymbol{\theta}} \left[\frac{1}{2m} \sum_{i=1}^{m} \left(\boldsymbol{\theta} \cdot \mathbf{x}^{(\mathbf{i})} - \mathrm{y}^{(\mathrm{i})} \right)^2 \right]$$

- Missing details?

Calculating the Partial Derivatives

$$\frac{\partial}{\partial\theta_j}J(\theta_0,\theta_1 \dots,\theta_n) = \frac{\partial}{\partial\theta_j}\frac{1}{2m}\sum_{i=1}^{m}(h_\theta\left(x^{(i)}\right) - y^{(i)})^2$$

$$= \frac{1}{2m}\sum_{i=1}^{m}\frac{\partial}{\partial\theta_j}(\theta_0 + \theta_1 x_1^{(i)} + \dots + \theta_j x_j^{(i)} + \dots \theta_n x_n^{(i)} - y^{(i)})^2$$

$$= \frac{1}{2m}\sum_{i=1}^{m}2(\theta_0 + \theta_1 x_1^{(i)} + \dots + \theta_j x_j^{(i)} + \dots \theta_n x_n^{(i)} - y^{(i)}) \cdot x_j^{(i)}$$

$$= \frac{1}{m}\sum_{i=1}^{m}\left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right) \cdot x_j^{(i)}$$

Gradient Descent for Linear Regression

- Initialize $\boldsymbol{\theta} = \left(\theta_0, \theta_1, \theta_2, \ldots, \theta_n\right)$
- Repeat until you reach a minimum (or stop cdn):
  - For all 0≤j ≤n,

$$\theta_j\,(t+1) = \theta_j(t) - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h_{\theta(t)}\left(x^{(i)}\right) - y^{(i)}\right) \cdot x_j^{(i)}$$

- In words: set the new θ_j to the current θ_j minus the learning rate (α) times the partial derivative of the error function with respect to θ_j , computed at the current θ.
- Also remember that $x_0^{(i)}$=1

IDC HERZLIYA

Closed form solution?

$$X \cdot \theta = y$$

$$\begin{array}{cc} & n+1 \\ m & \begin{pmatrix} x_0^{(1)} & x_1^{(1)} & \cdots & x_n^{(1)} \\ x_0^{(2)} & x_1^{(2)} & & x_n^{(2)} \\ & \vdots & \ddots & \vdots \\ x_0^{(m)} & x_1^{(m)} & \cdots & x_n^{(m)} \end{pmatrix} \end{array} \begin{array}{c} \theta \in \mathbb{R}^{n+1} \\ \begin{pmatrix} \theta_0 \\ \vdots \\ \theta_n \end{pmatrix} \end{array} = \begin{array}{c} Y \in \mathbb{R}^{m} \\ \begin{pmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(m)} \end{pmatrix} \end{array}$$

- If X is square and non singular we can write $\vec{\boldsymbol{\theta}} = X^{-1}y$
- In ML typically X is overdetermined ($m \gg n$).

Recitation

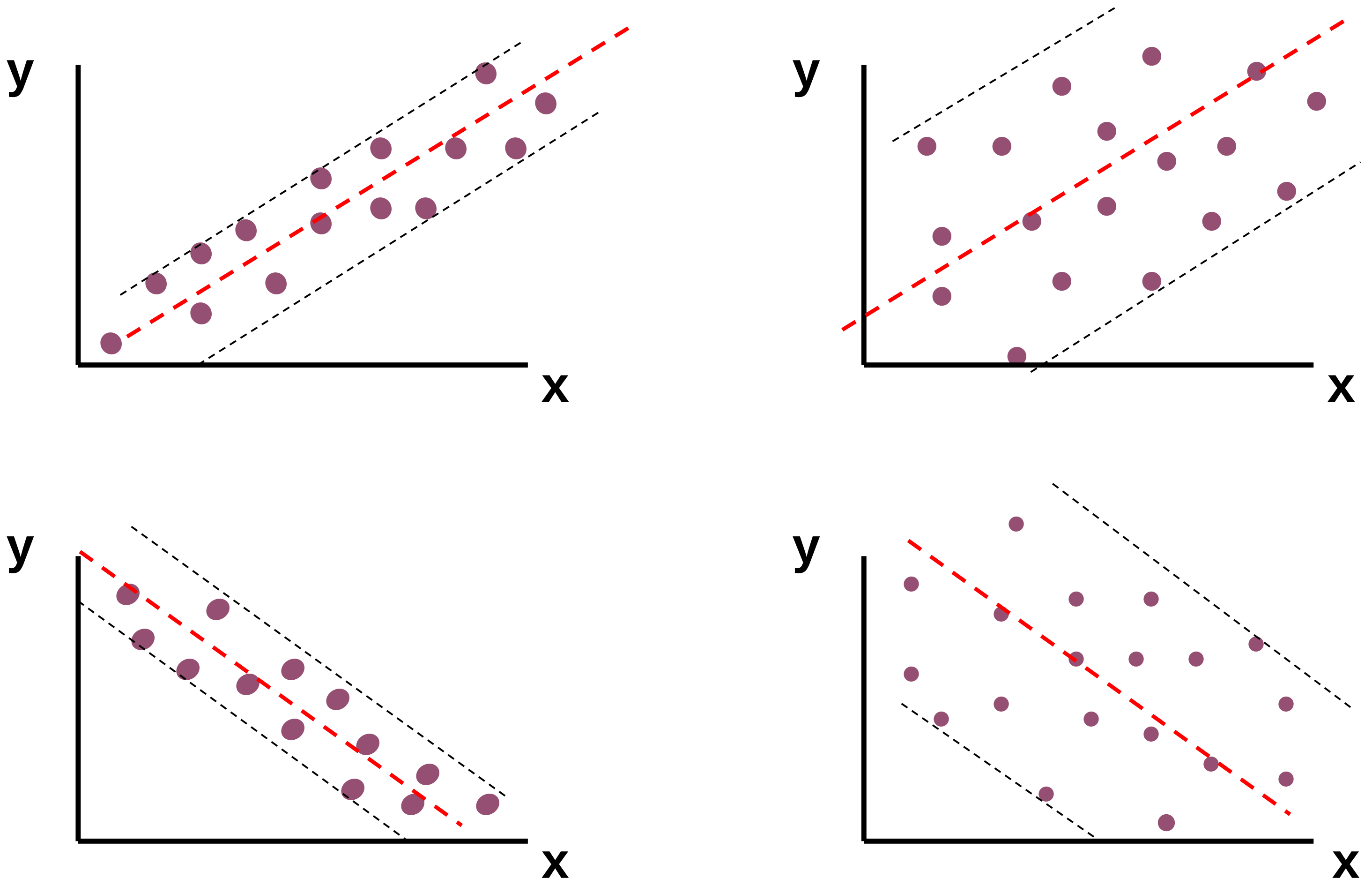
- Description of HWA1
- Feature scaling
- Learning rate
- Python

Extra slides/notes

- Correlations
- Pseudo-inverse of a matrix (closed form solution)
- Going beyond Linear (also in the HW)

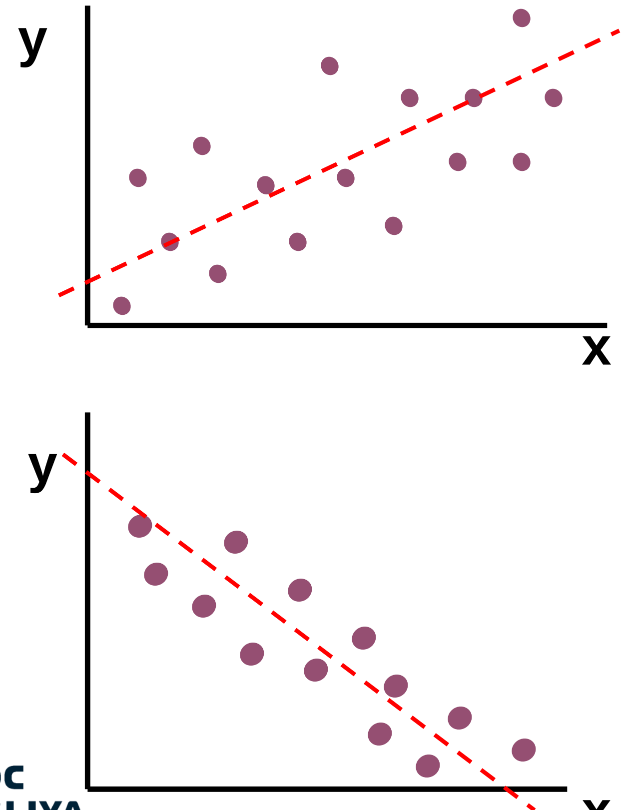
Strong vs. Weak Linear Relationship

Strong relationships | **Weak relationships**

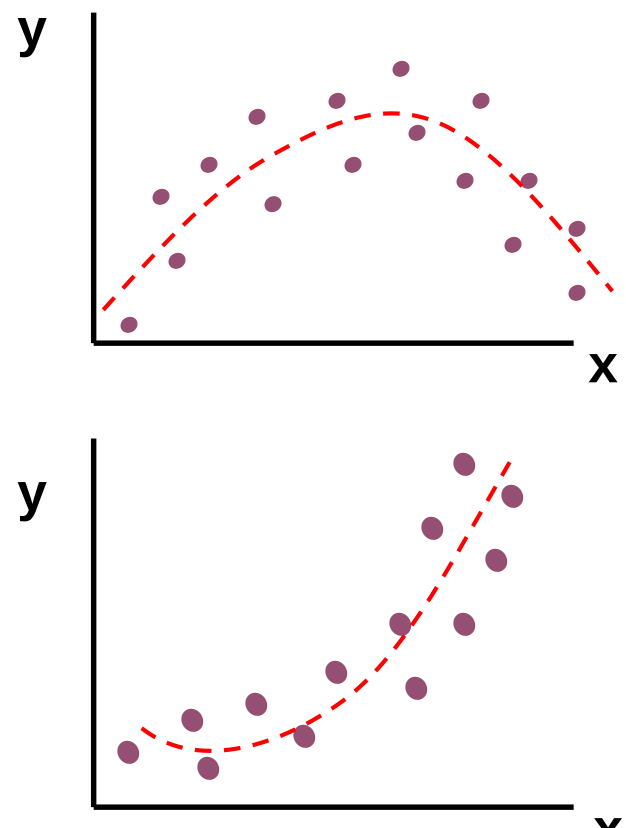


Not Everything is Linear!

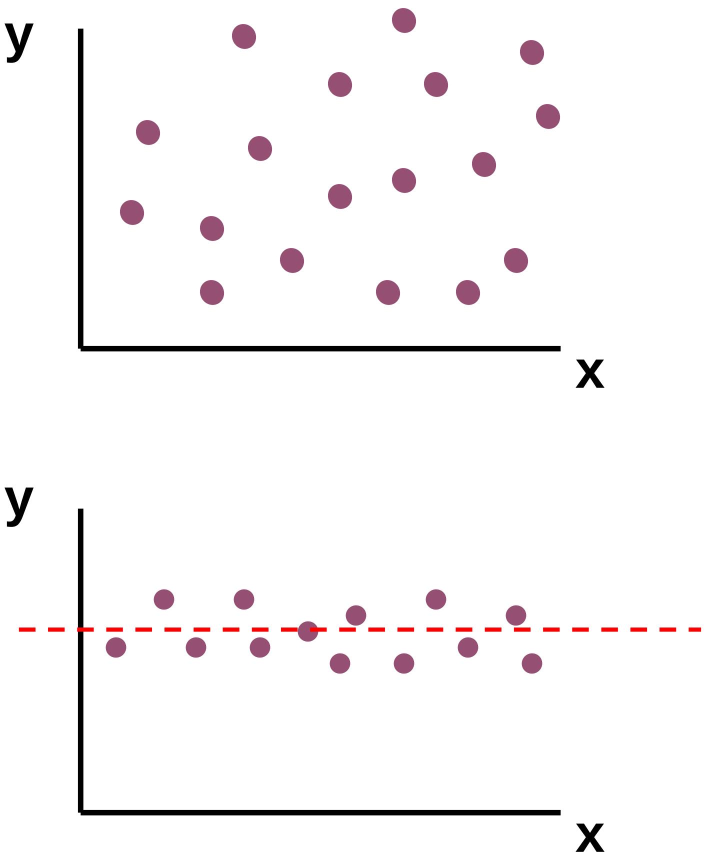
Linear relationships



Curvilinear relationships



No Evident Relationship



Correlation Analysis

- Correlation analysis is used to measure strength of the association (**linear relationship**) between two variables
  - Only concerned with strength of the relationship
  - No causal effect is implied
- The sample (Pearson) correlation coefficient **r** is a measure of the strength of the linear relationship between two variables, based on sample observations

The Pearson Correlation Coefficient

$$r = \frac{\sum_{i=1}^{m}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{m}(x_i - \bar{x})^2 \sum_{i=1}^{m}(y_i - \bar{y})^2}} = \frac{\sigma_{xy}}{\sigma_x \sigma_y}$$

Where:

r = Sample Pearson correlation coefficient

m = Number of samples

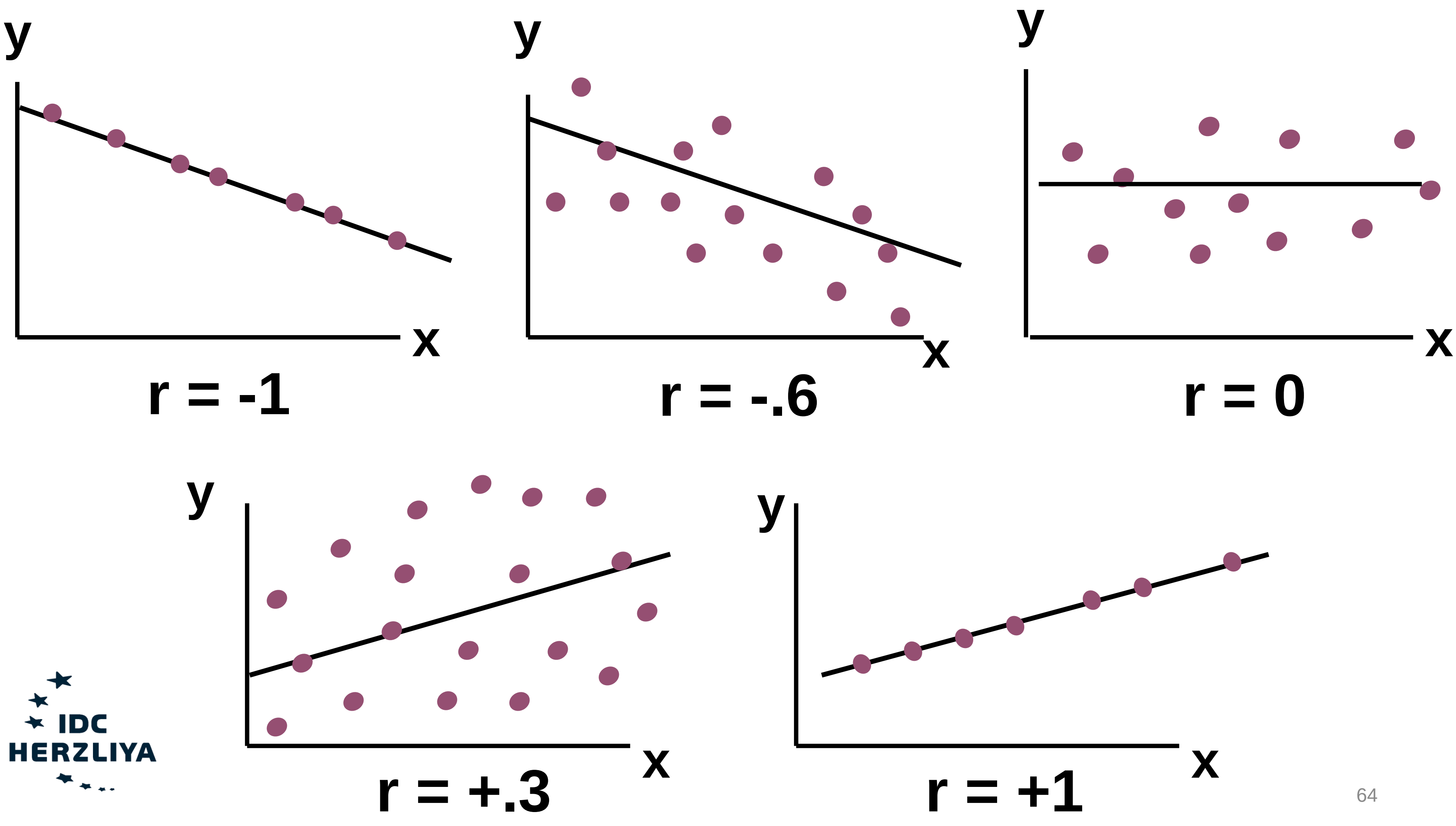
x = Value of an explaining variable

y = Value of the dependent variable

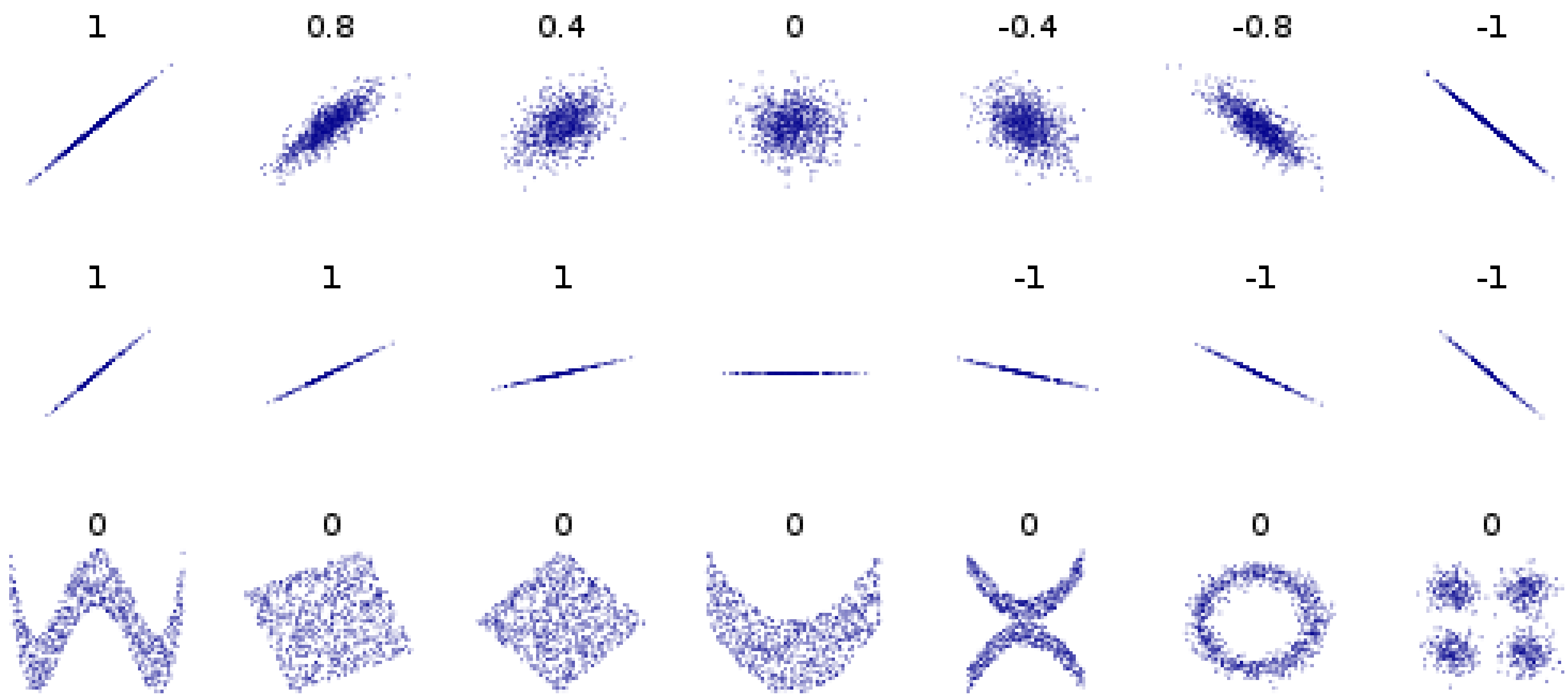
Features of r

- Unit free
- Ranges between -1 and 1
- The closer to 0, the weaker the <u>linear</u> relationship
- The closer to -1, the stronger the negative <u>linear</u> relationship
- The closer to 1, the stronger the positive <u>linear</u> relationship

Examples of r Values



More Complex Relationships May Not Be Captured Using Pearson r



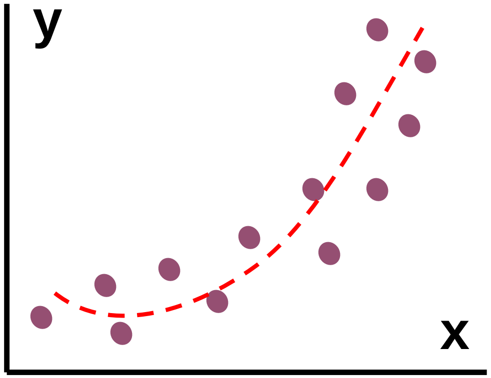
Polynomial Regression

- We can expand our feature space by using functions of the original features.
- For example, if we want to use a cubic function feature space we can define:

$$x_0 = 1, x_1 = x, x_2 = x^2, x_3 = x^3$$

then use regular regression and in essence we are learning the function

$$\theta_0 + \theta_1 x + \theta_1 x^2 + \theta_3 x^3$$



Other Functions

- We can use other functions of features such as $x^{1/q}$
- We can define functions of sets of variables such as $x_1x_5x_7$ or $x_1^2x_5x_7^3$
- All of these can be represented as new features and increase the dimension of the entire calculation

IDC HERZLIYA

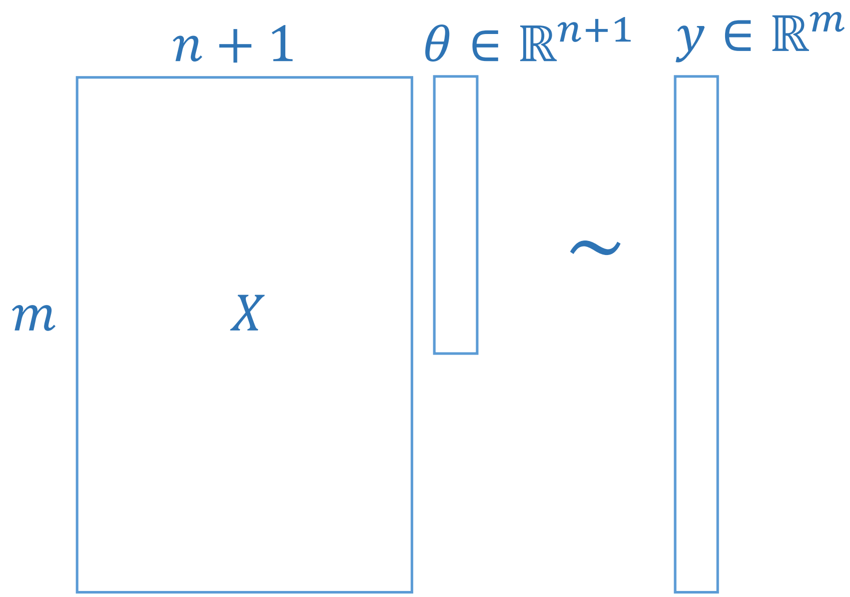
Closed form solution?

$$X \cdot \theta = y$$

$$\begin{array}{cc} & \begin{array}{c} n+1 \end{array} \quad \theta \in \mathbb{R}^{n+1} \quad Y \in \mathbb{R}^m \\ m & \begin{pmatrix} x_0^{(1)} & x_1^{(1)} & \cdots & x_n^{(1)} \\ x_0^{(2)} & x_1^{(2)} & & x_n^{(2)} \\ & \vdots & \ddots & \vdots \\ x_0^{(m)} & x_1^{(m)} & \cdots & x_n^{(m)} \end{pmatrix} \begin{pmatrix} \theta_0 \\ \vdots \\ \theta_n \end{pmatrix} = \begin{pmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(m)} \end{pmatrix} \end{array}$$

- If X is square and non singular we can write $\vec{\boldsymbol{\theta}} = X^{-1}y$
- Most of the time X is overdetermined ($m \gg n$).

Schematic set-up



Minimum when $\nabla = 0$

- Remember the (MSE) cost function:

$$E(\boldsymbol{\theta}) = \left\| X \cdot \vec{\boldsymbol{\theta}} - y \right\|_2^2 = \sum_{i=1}^{m} \left(x^{(i)} \theta^T - y^{(i)} \right)^2$$

- This will be minimal when the gradient ∇E is 0.
- Forming the derivatives yields (see below):

$$\nabla E(\boldsymbol{\theta}) = \sum_{\boldsymbol{i=1}}^{\boldsymbol{m}} \mathbf{2}\left(\boldsymbol{x}^{(\boldsymbol{i})} \boldsymbol{\theta}^{\boldsymbol{T}} - \boldsymbol{y}^{(\boldsymbol{i})}\right) \boldsymbol{x}^{(\boldsymbol{i})} = \mathbf{2} \boldsymbol{X}^{\boldsymbol{T}} (\boldsymbol{X\theta} - \boldsymbol{y})$$

Proof

$$\nabla E(\theta) = 0\,, \quad \nabla E(\theta) = \left(\frac{\partial E}{\partial \theta_j}(\theta)\right)_{j=1}^{n}$$

Expanding the partial derivatives, we get the following expression (for setting to 0)

$$\left(\nabla E(\theta)\right)_j = \frac{\partial E}{\partial \theta_j}(\theta) = \frac{\partial}{\partial \theta_j}\left(\sum_{i=1}^{m} \left(X(i,:)\cdot\theta - y(i)\right)^2\right)$$

Recall that $X(i,:)$ is the notation for the i^{th} row. Now we expand the dot product

$$\frac{\partial E}{\partial \theta_j}(\theta) = \frac{\partial}{\partial \theta_j}\left(\sum_{i=1}^{m}\left(\left(\sum_{k=1}^{n} X(i,k)\cdot\theta_k\right) - y(i)\right)^2\right)$$

$$\frac{\partial E}{\partial \theta_j}(\theta) = \left(\sum_{i=1}^{m}\frac{\partial}{\partial \theta_j}\left(\left(\sum_{k=1}^{n} X(i,k)\cdot\theta_k\right) - y(i)\right)^2\right)$$

$$\frac{\partial E}{\partial \theta_j}(\theta) = \left(\sum_{i=1}^{m} 2\left(\left(\sum_{k=1}^{n} X(i,k)\cdot\theta_k\right) - y(i)\right)\cdot X(i,j)\right)$$

Proof, cont

For all $j \neq k$ the derivative is 0, so we only consider θ_j

$$\nabla E(\theta) = \left(2\sum_{i=1}^{m}\big(X(i,:)\cdot\theta - y(i)\big)\cdot X(i,j)\right)$$

$$\nabla E(\theta) = (2X(:,j)(X\cdot\theta - y))$$

Notice that $\big(X(:,j)\big)^T = X^T(j,:)$ and therefore

$$\nabla E(\theta) = (2X^T(j,:)(X\cdot\theta - y))$$

$$\nabla E(\theta) = (2X^T(X\cdot\theta - y))$$

Recall that we want to solve for the θ that makes this 0:

$$0 = 2X^T(X\cdot\theta - y)$$

$$0 = (X^TX\cdot\theta - X^Ty)$$

$$X^TX\cdot\theta = X^Ty$$

$$\theta = (X^TX)^{-1}X^Ty$$

IDC HERZLIYA

Pseudo-inverse & the closed form solution - Summary

- Setting the gradient of E to zero yields the necessary condition for minimum (see notes and slides above): $X^T X \cdot \theta = X^T y$

- Now, $\boldsymbol{X^T X}$ is square and often nonsingular and so we can solve for θ uniquely as:

$$\boxed{\theta = pinv(X)y \qquad \text{where} \qquad pinv(X) = (X^T X)^{-1} X^T}$$

- The n×m matrix $\boldsymbol{pinv(X) = (X^T X)^{-1} X^T}$ is called the **pseudo inverse** of **X** (which is mxn)
- If $\boldsymbol{X}$ is square it is just its inverse.

Gradient Descent and Pseudo Inverse

- $\boldsymbol{X^T X}$ can still be singular (or not full rank) and not have an inverse. This can be resolved with some more algebra.
- This pinv technique doesn't work for all error functions J. Gradient descent is more general.
- Gradinet descent allows parallelization.

IDC HERZLIYA

Summary

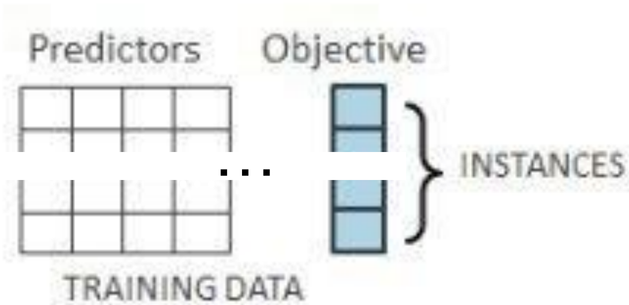
- Regression learns a function to predict values based on a vector of features
- Linear Regression uses a Gradient Descent Algorithm or a Pseudo-Inverse solution
- Gradient descent is a general minimization procedure
- There are other (non-linear) relations between variables
- The Pearson correlation coefficient is a measure of the strength of the linear relationship between two variables
- Pseudo inverse solution

IDC HERZLIYA

Summary - cont

The execution algorithm:

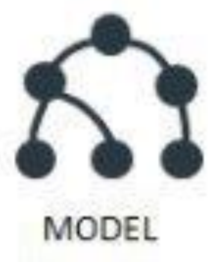
$y = f(x) = x\theta^T$



X (instances by features)

y (instances)

The learning algorithm (regression)



θ