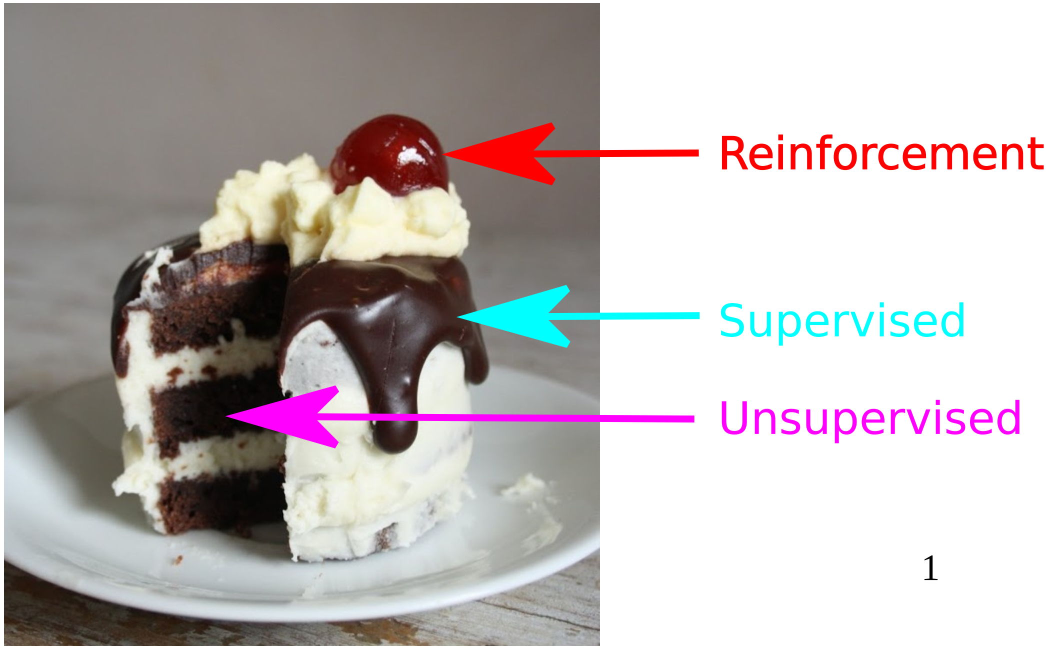
Definitions
What is ML ?

- ***a* definition:**

  For a given **Task** T, a **machine** (algorithm) A
  obtains better **performance** P
  after an **experiment** E. (It has ***learned*** from it)
  (Experiment ~ data)

Yann LeCun's cake metaphor:



- **<u>3 types</u>** of learning :

  - Supervised: w/ labels

  - Unsupervised: w/o labels
  (incl. self-supervised)

  - Reinforcement
  (outside this course)

Today – Outline

- **Supervised** Learning basics:
  - Linear **regression**
  - Polynomial regression
- Lots of **Vocabulary**, notations
- Optimization basics: **Gradient Descent**
- **Supervised** Learning
  - Classification with the Perceptron (maybe)

Today:
Supervised Learning

Input: $\vec{x}_n = (x_{n,d})_{d\in[1,\ldots,D]}, X = \{\vec{x}_n\}_{n\in[1,\ldots,N]}$

- Expected Output: y^{GT} or t_n (***Ground <u>T</u>ruth***)

  Which kind of Task → depends on type of t_n

- ***Model***: $y^{predicted} \equiv \hat{y}_n = \sigma(f_\Theta(\vec{x}_n))$
  fct. f_Θ is ***parameterized*** by ***parameters*** Θ

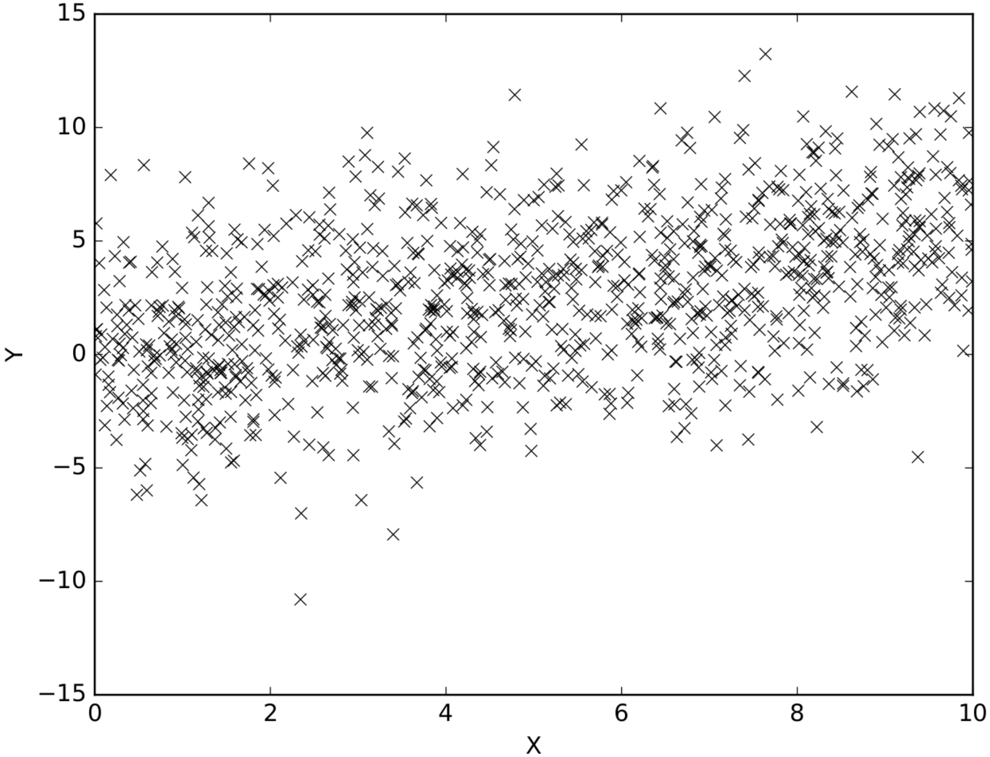
- ***Learning :*** finding *optimal* parameters to minimize discrepancy between $\hat{y}$ and Ground Truth t

$$\Theta^* = argmin_\Theta\Big(\sum_n^N \ell(\hat{y}_n, t_n)\Big)$$

- ***Cost*** *Function* ***(loss*** *function)* : to be chosen

Supervised Learning: **Regression**

Pairs of data points $\vec{x}_n = (x_{n,1}, x_{n,2})$



→ Relationship $f(x)=y$?

→ **Regression**

- ***linear***: $f_{a,b}(x) = ax + b$ or $f_{\vec{a},b}(\vec{x}) = \vec{a} \cdot \vec{x} + b$

- ***polynomial***: $f_{\Theta}(\vec{x}) = \vec{\theta} \cdot \Phi(\vec{x})$

(degree P) *(see polynomial feature maps)*

More Vocabulary
(+case of Regression)

Input: $\vec{x}_n = (x_{n,d})_{d\in[1,...,D]}, X = \{\vec{x}_n\}_{n\in[1,...,N]} = (x_{n,d})_{(N,D)}$

- *Ground Truth*: $t_n \in \mathbb{R}, T = \{t_n\}_{n\in[1,...,N]}$

  Continuous output → Task is **Regression**

- **Model: e.g.** a linear function of the input : $f_{\vec{a},b}(\vec{x}) = \vec{a}\cdot\vec{x} + b$

  - Parameters: $\Theta = \{b, a_d; d = 1, .., D\}$

  $$Card(\Theta) = 1 + D$$

  - **Prediction:** simply $\hat{y}_n = f_\Theta(\vec{x}_n)$

- Learning **Algorithm:**

  - ***Initialization:*** $\Theta = \Theta_0$

  - Minimize some Loss $\ell(\hat{y}_n, t_n)$ (to choose)

  - For this, use some minimization scheme (Grad. Desc.)

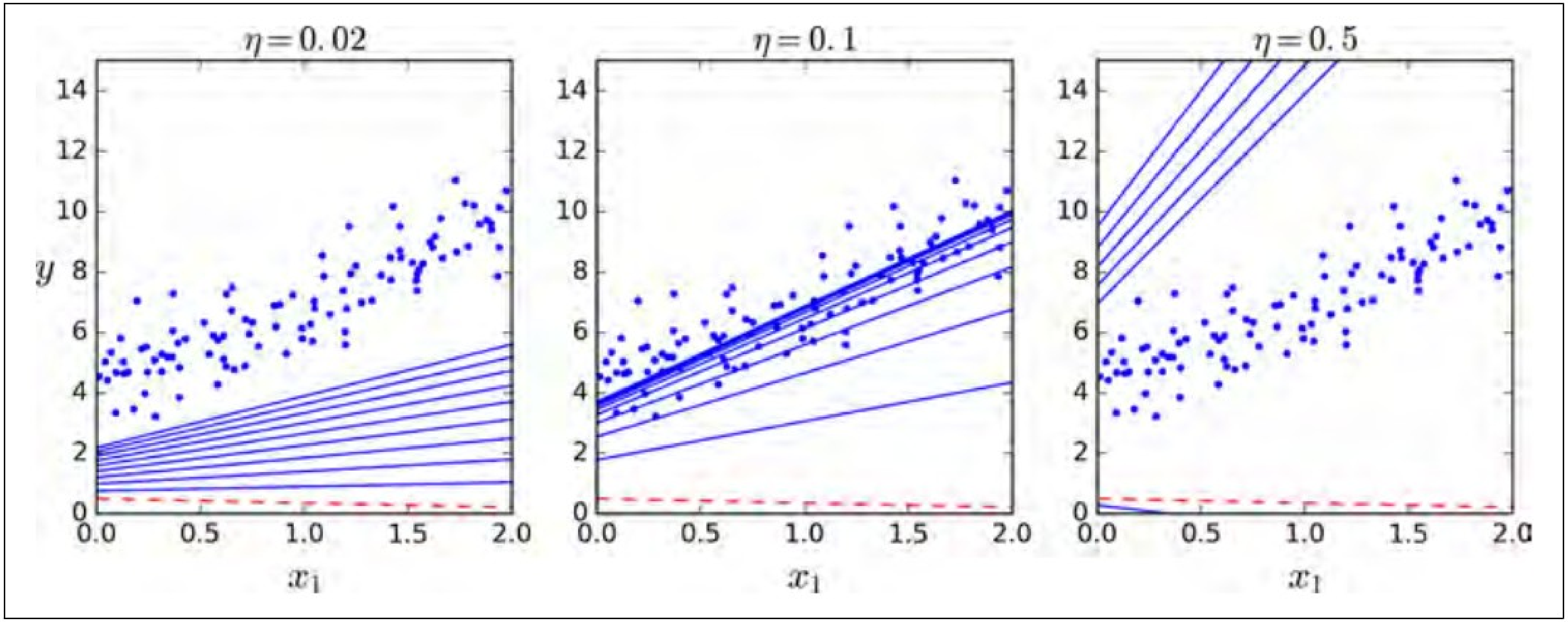
Supervised Learning: **Regression**

- We can choose: **Least Squares**

Single data point Loss: $\ell(f_\Theta(\vec{x}_n), t_n) = (f(\vec{x}_n) - t_n)^2$

$$\text{Gloabal Loss: } \mathcal{L}(\Theta, X, T) = \frac{1}{N}\sum_{n=1}^{N} \ell(f_\Theta(\vec{x}_n), t_n)$$

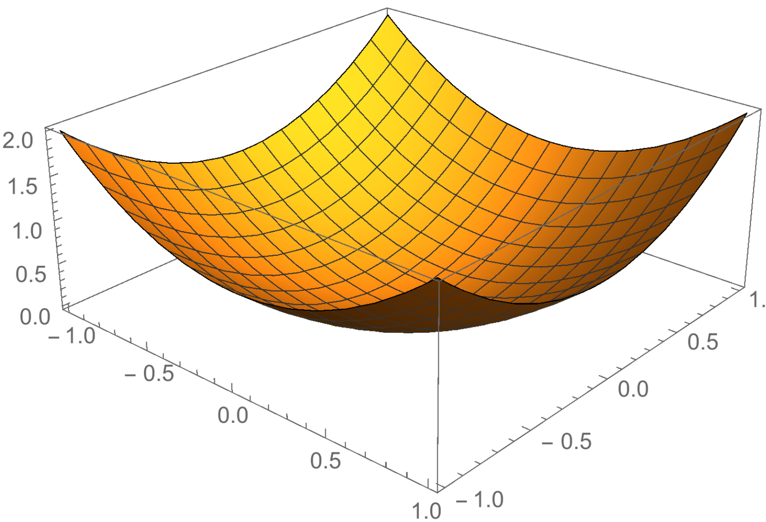
- **Gradient Descent**:

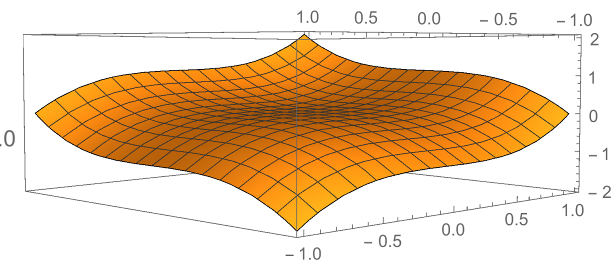


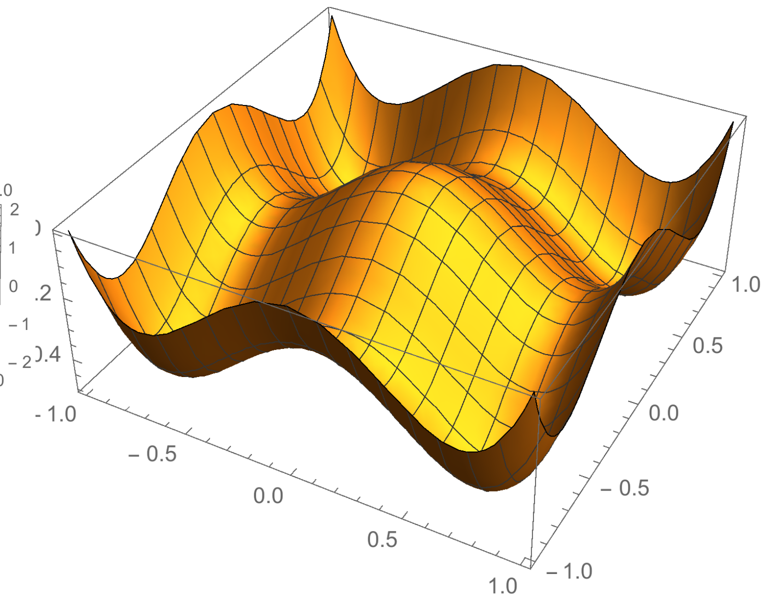
Gradient Descent
short reminder

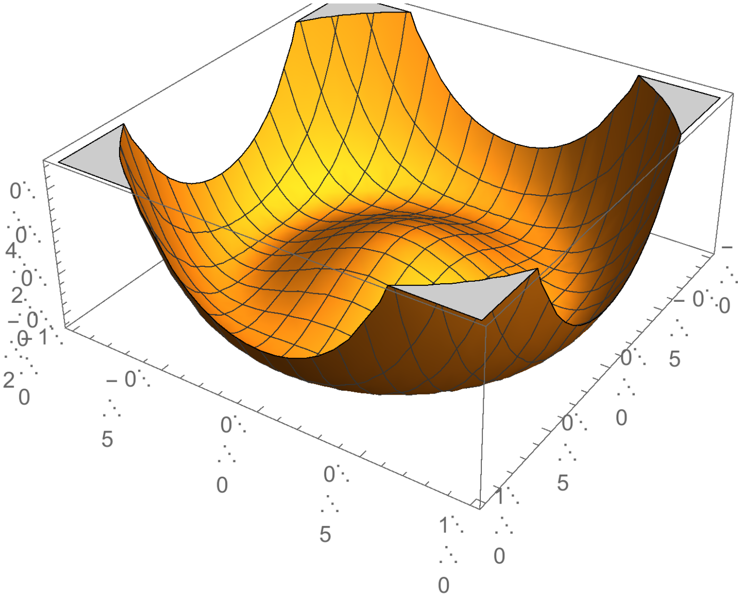
- I have a function $J(\theta)$ and want to find the value $\theta*$ for which $J(\theta)$ is minimum

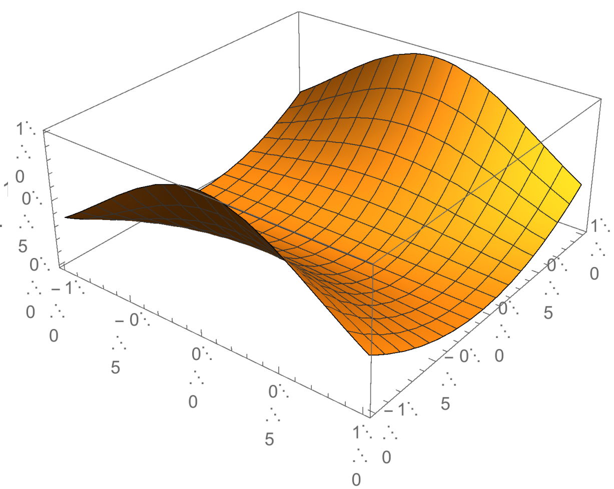
What is the gradient ?

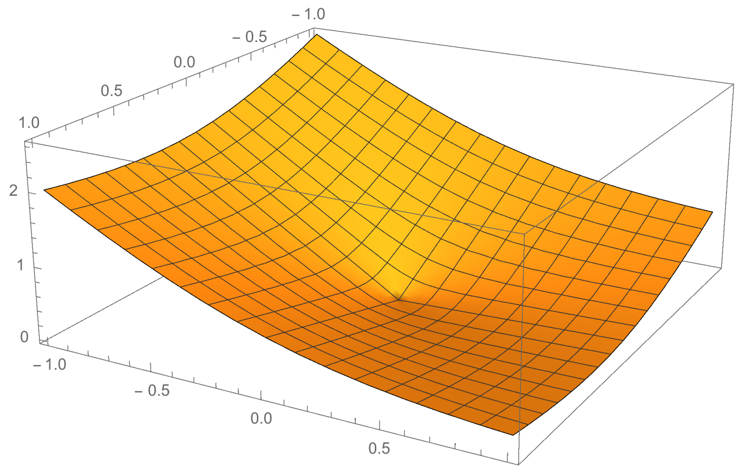




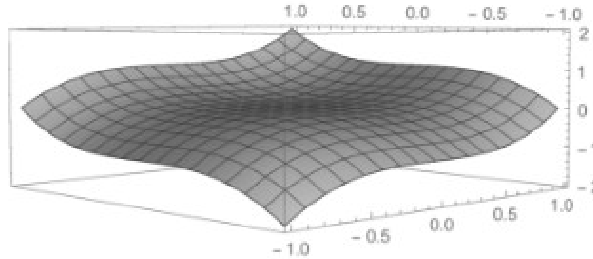


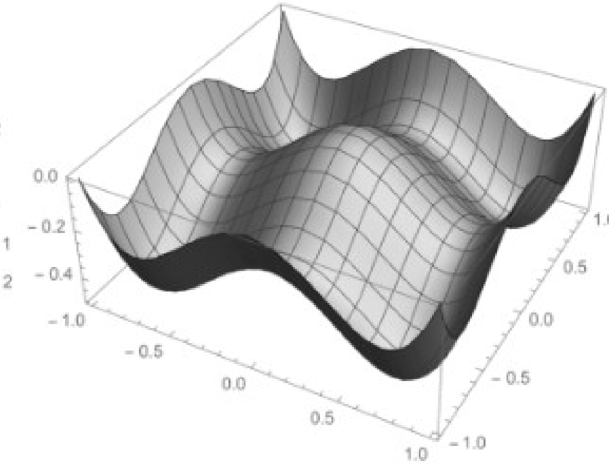




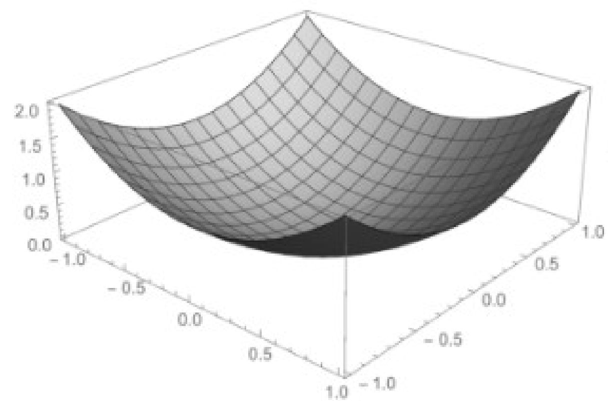


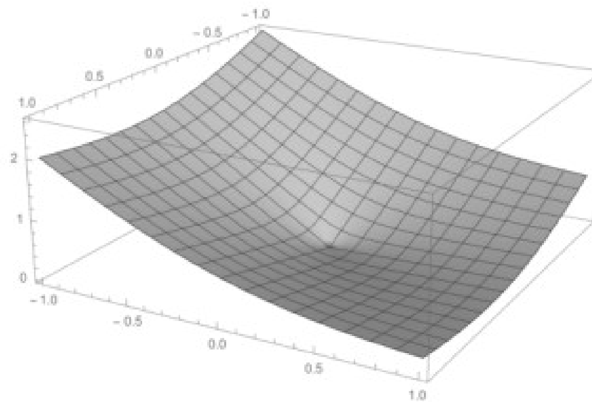
$x^3 + y^3$, et $-(x^2+y^2)+(x^4+y^4)$:



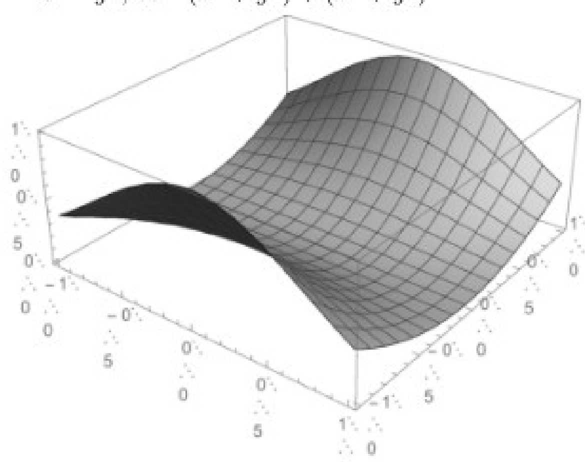


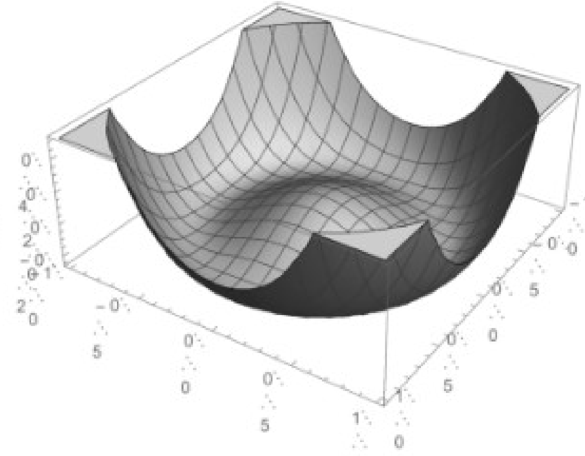
x^2+y^2 et $||\vec{x}|| - a\vec{w}\cdot\vec{x} = (x^2+y^2)^{1/2} - aw_1x - aw_2y$, avec $a=3, w_1=0.1, w_2=0.3$:





$e^{-x^2}y^2$, et $-(x^2+y^2)+(x^2+y^2)^2$





Gradient Descent

- It goes in the steepest direction (from the local point) → is also called "*steepest descent*"
- Limitations:
  - at best, converges to ***one of*** *the* ***local*** *minima*
  - *typically* converges to the **local attractor** (min. in the local basin of attraction)
  - Result **depends on starting** position !
  - it **may never converge** ! (diverge or continuously go down)

Least Squares

$$\mathcal{L} = \frac{1}{N}\sum_{n=1}^{N}\left(f_\Theta(\vec{x}_n) - y_n\right)^2$$

, with a linear model

(multivariate case) $$\mathcal{L} = \frac{1}{N}\sum_{n=1}^{N}\left(\vec{f}_\Theta(\vec{x}_n) - \vec{y}_n\right)^2$$

Trick: Augmented data

- Add 1's into X to take care of the offset, once and for all
  $\rightarrow$ get cleaner equations (and cleaner code) !

A word on unsupervised: the example of K-means

- Goal: find groups in data (X). No labels (y).
- Idea: assign “classes” (assignment to a *group*, aka *cluster*) to points anyway, and ask to have **homogeneous groups**:
  - close-by points should belong to the same cluster
  - clusters should be batches of points which are close enough
- In practice: cook a cost function J that realizes this, then minimize it.

  J =

- Numerical minimization is performed approximately, by starting at random, then iterating 2 steps:
  - each point is assigned to the closest cluster center
  - each cluster center is the barycenter of the data points assigned to it

References:

Linear regression (by G.D.)
→Bishop book, page 143-144, section 3.1.3 (sequential learning)

→ https://en.wikipedia.org/wiki/Least_squares#Linear_least_squares

- **Gradient Descent** (assumed known)
→ catch up lesson:
https://en.wikipedia.org/wiki/Gradient_descent

Key concepts

- **Supervised** Learning
- **Regression**
- **Task, Model,** parameters**, prediction**/decision, input **feature**