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)

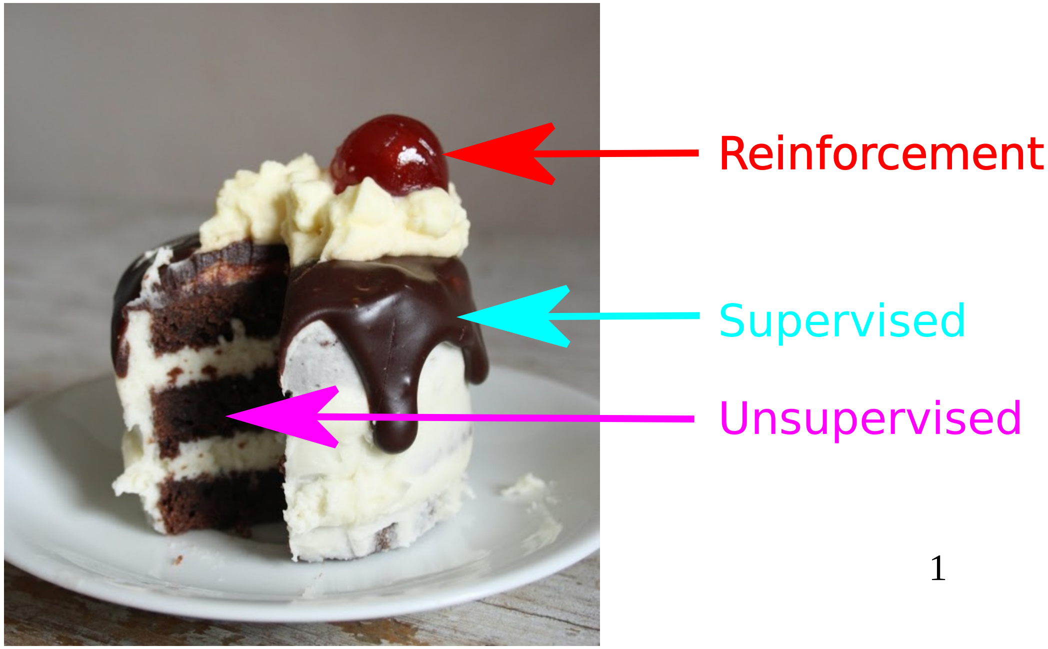
- **<u>3 types</u>** of learning :

- Supervised

- Unsupervised

- Reinforcement
(outside this course)

Yann LeCun's cake metaphor:



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

- Expected Output: y^{GT} or $t^{(n)}$ (***Ground Truth***)

  Which kind of Task → depends on $t^{(n)}$

- ***Model***: $y^{predicted} \equiv \hat{y}^{(n)} = \sigma(f_\Theta(\vec{x}^{(n)}))$

  fct. f_Θ is ***parameterized*** by ***parameters*** Θ

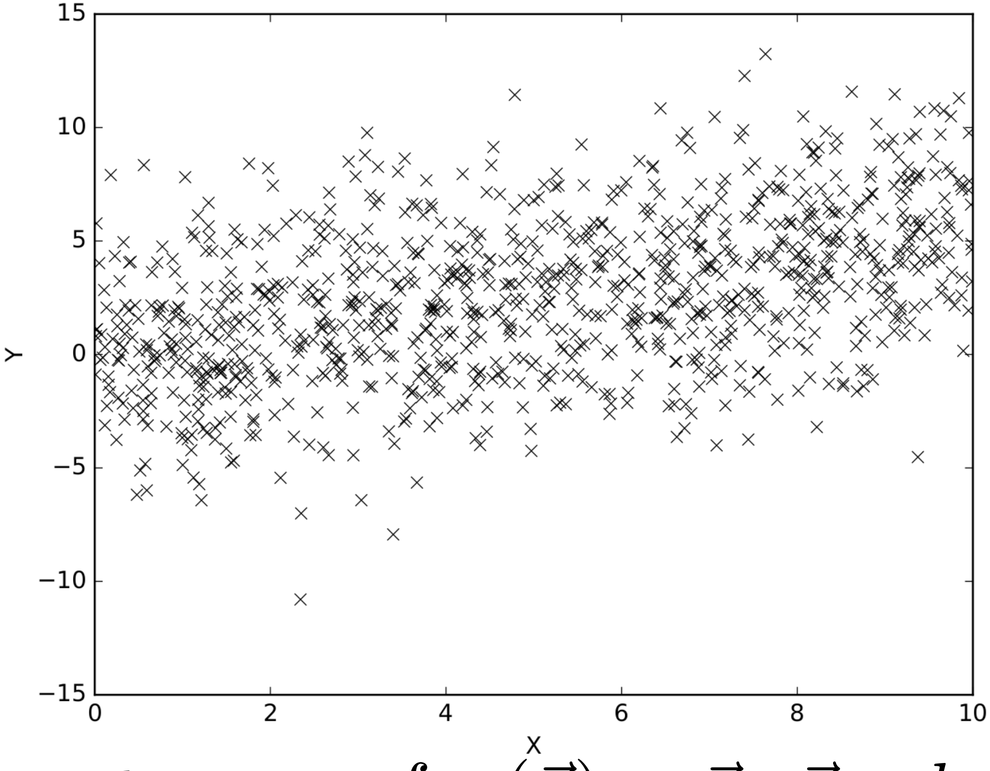
- ***Learning :*** finding *optimal* parameters to minimize discrepancy between $\hat{y}$ and Ground Truth t

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

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

Supervised Learning: **Regression**

Pairs of data points $\vec{x}^{(n)} = (x_1^{(n)}, x_2^{(n)})$



→ 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***: $$f_\Theta(\vec{x}) = \sum_{p=0}^{P} \vec{\theta}_p \cdot \vec{x}^p$$

  (degree P)

More Vocabulary
(+case of Regression)

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

- *Ground Truth*: $t^{(n)} \in \mathbb{R}, T = \{t^{(n)}\}_{n\in[1,\ldots,N]}$

  Continuous output → Task is **Regression**

- **Model: e.g.** a polynomial function of the input : $f_\Theta(\vec{x}) = \sum_{p=0}^{P} \vec{\theta}_p \cdot \vec{x}^p$

  - Parameters: $\Theta = \{\theta_0, \theta_{d,p}/d = 1, .., D, p = 1, \ldots, P\}$

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

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

- Learning **Algorithm:**

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

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

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

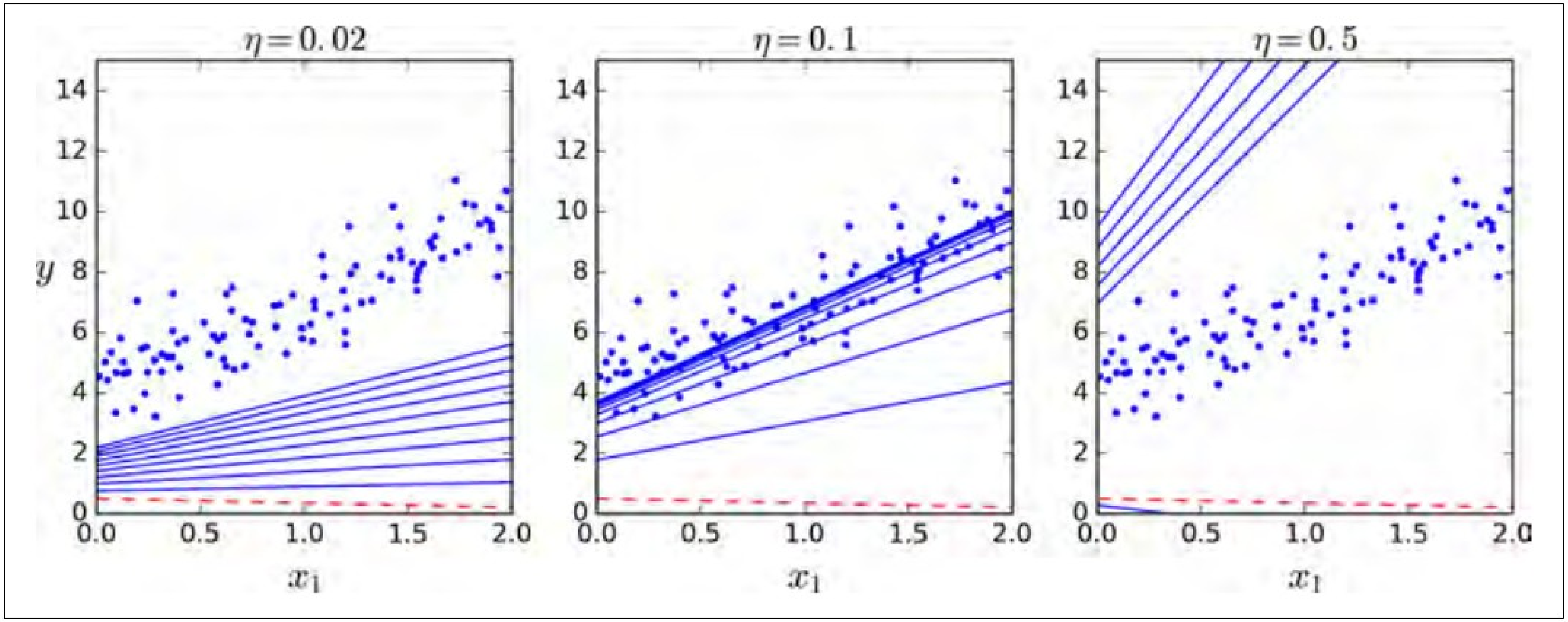
Supervised Learning: **Regression**

- We can choose: **Least Squares**

$$\text{Single data point Loss: } \mathcal{L}(f_\Theta(\vec{x}_n), t_n) = (\vec{f}(\vec{x}^{(n)}) - t^{(n)})^2$$

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

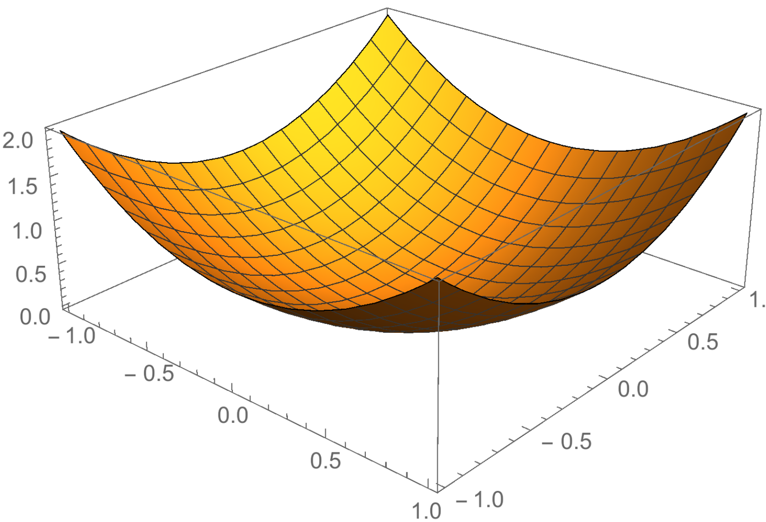
- **Gradient Descent**:

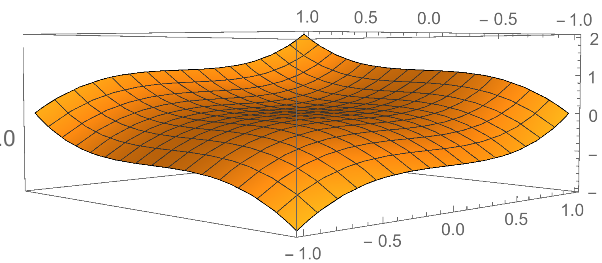


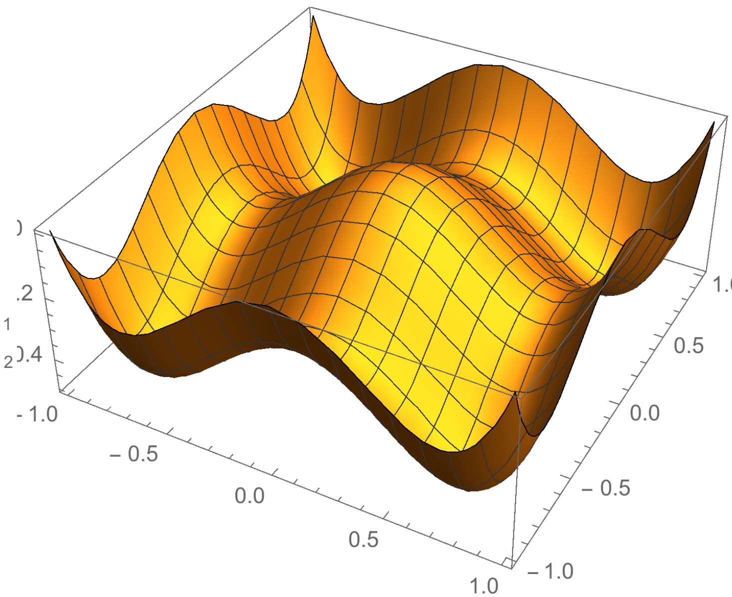
Gradient Descent
short reminder

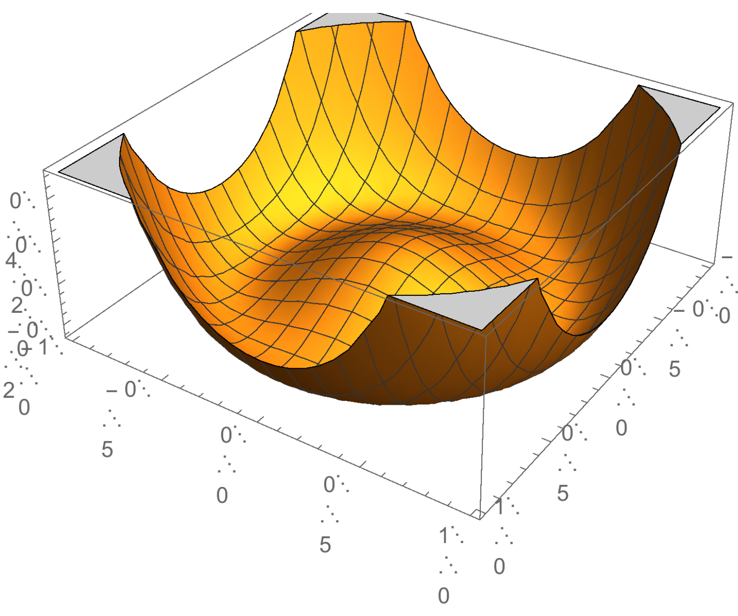
- I have a function $J(\theta)$ and want to find the value $\theta*$ for which $J(\theta)$ is minimum

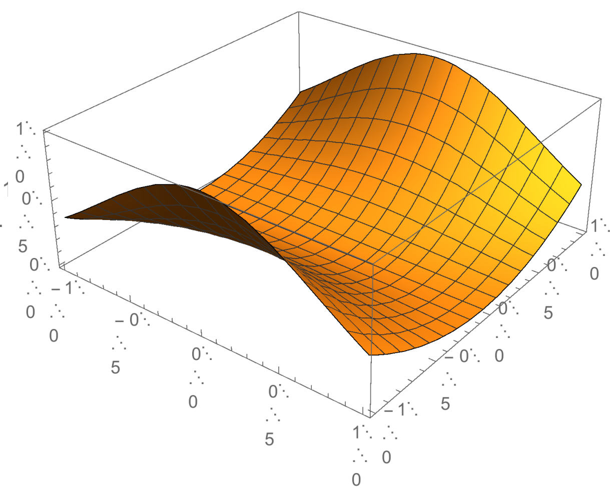
What is the gradient ?

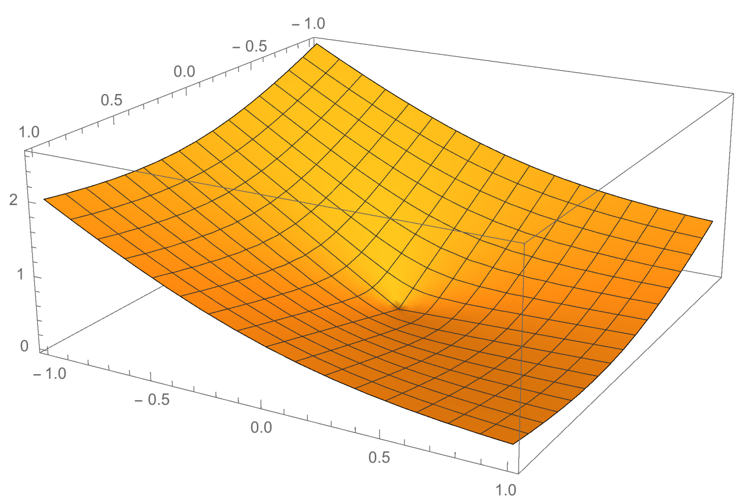




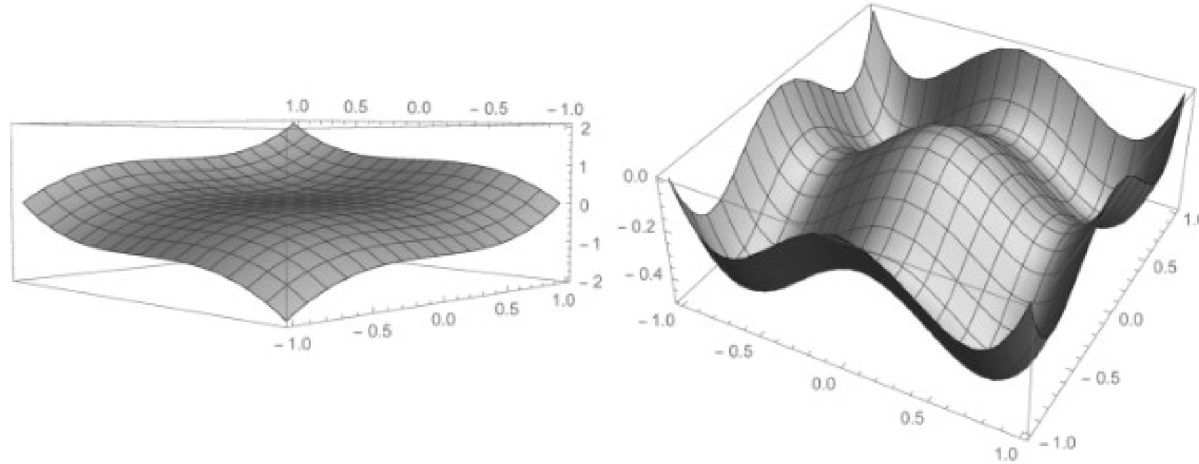




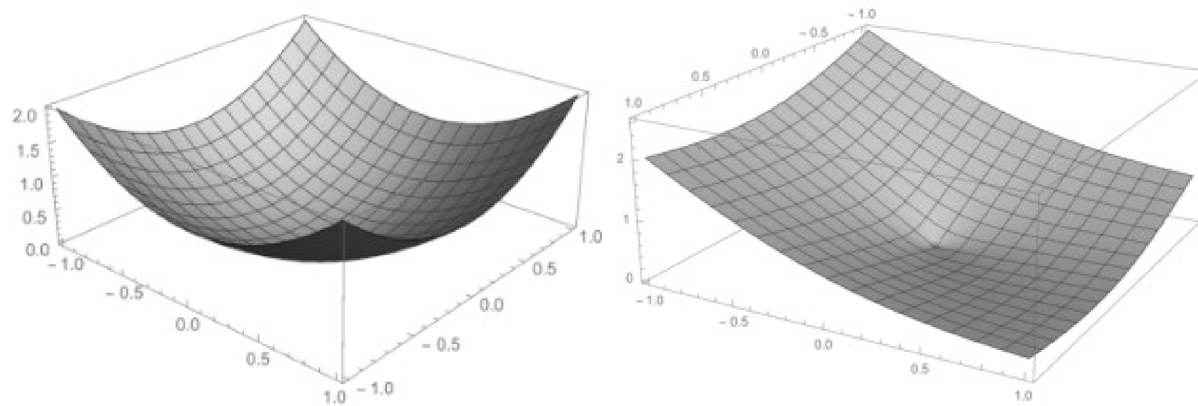




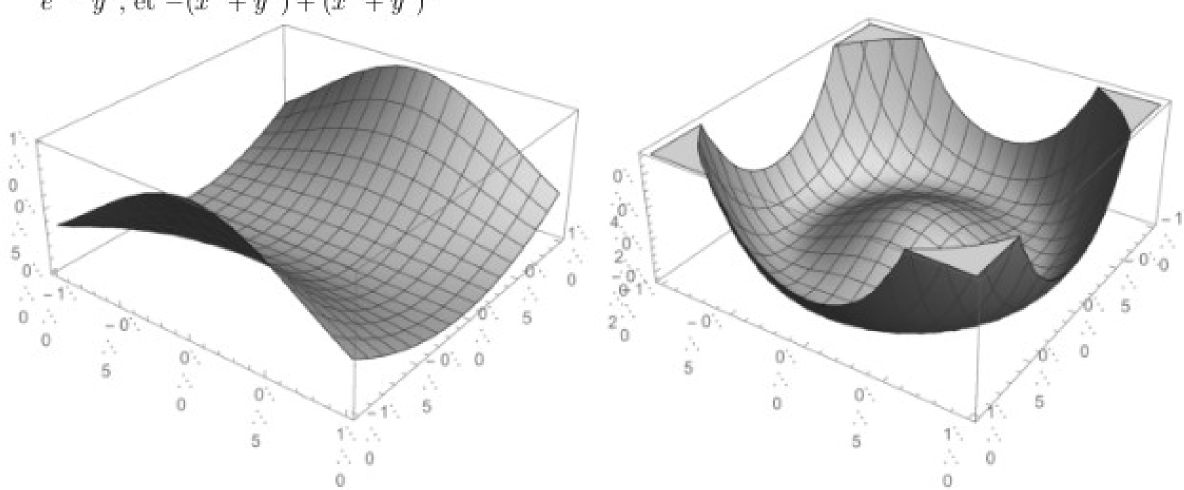
$x^3 + y^3$, et $-(x^2 + y^2) + (x^4 + y^4)$:



$x^2 + y^2$ et $||\vec{x}|| - a\vec{w} \cdot \vec{x} = (x^2 + y^2)^{1/2} - aw_1 x - aw_2 y$, 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

- Limitations:
  - at best, converges to *one of the* ***local*** *minima*
  - typically converges to the minimum of the local basin of attraction we are in
  - there may be many local minima. The best one may not be close to our (random) starting position...
  - it may never converge (diverge or continuously go down)
  - it goes in the steepest direction (from the local point) $\rightarrow$ is also called “*steepest descent*”

Least Squares

$$LSE = \frac{1}{N}\sum_{n=1}^{N}(\vec{f}_\Theta(x^{(n)}) - \vec{y}^{(n)})^2$$, with $$f_\Theta(\vec{x}) = \sum_{p=0}^{P}\vec{\theta}_p \cdot \vec{x}^p$$

Case $P = 1$

Trick: Augmented data

- Add 1’s into X to take care of the offset, once and for
  → get cleaner equations (and cleaner code) !

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

Pause-Questions

First nuance: various **optimization strategies**

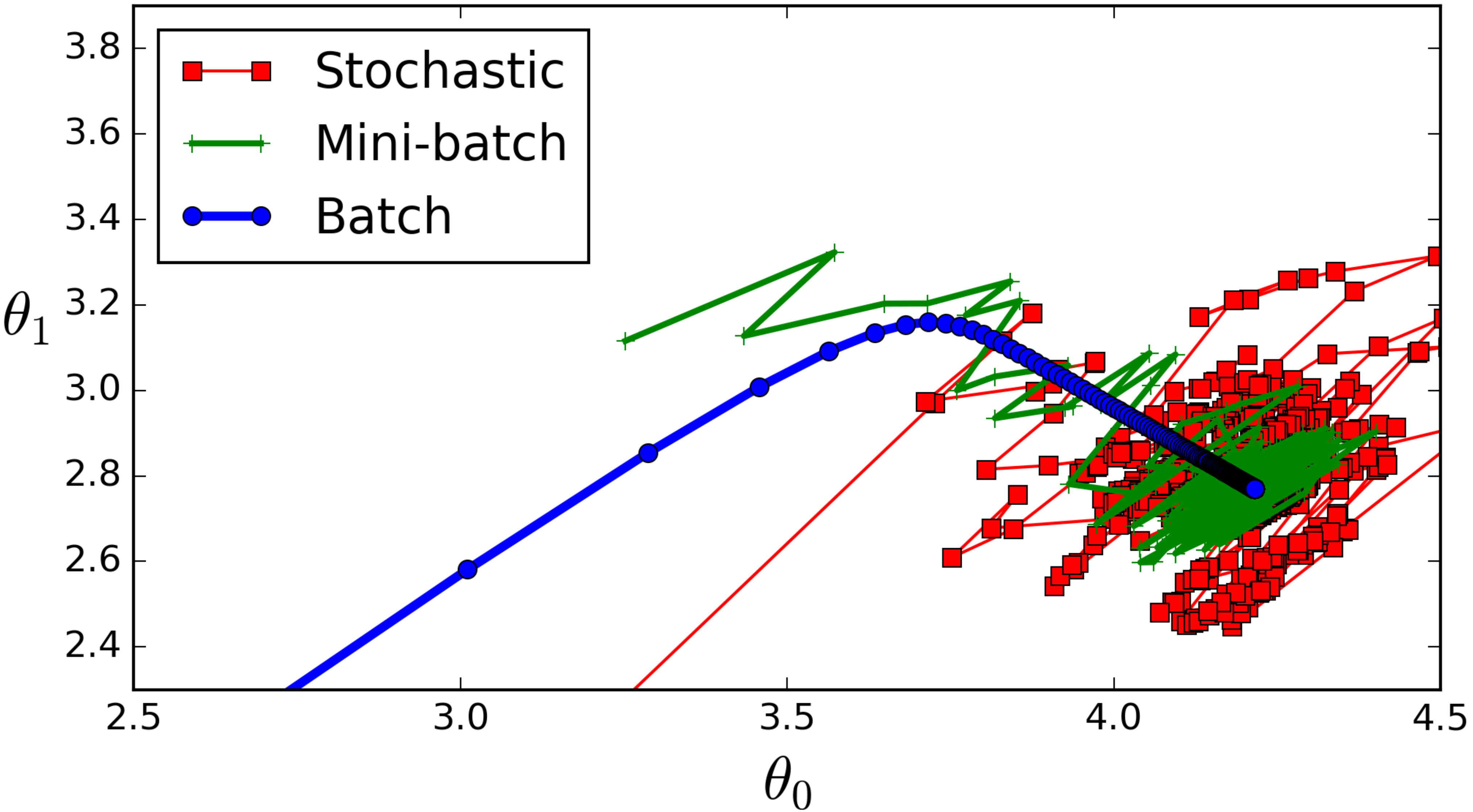
- ***Examples*** seen one by one:
  "Online" learning

- Examples seen all at once $(m{=}N)$
  globale update **(optimization viewpoint)**

- Intermediate solution: ***batch size*** m, $(m{>}1)$
  mini-batch Gradient Descent

- ***Stochastic*** *Gradient Descent* (**SGD**): $(m{=}1)$
  ~looks like *Online* (but more random)

SGD vs *mini-batch* vs *full batch*



Key concepts

- **Supervised** Learning
- **Regression**
- **Task, Model,** parameters**, prediction**/decision, input **feature**
- [SGD, mini-batch, full batch, Online]