
POLITECNICO
DI TORINO

AI ML

# Artificial Intelligence and Machine Learning

A.A. 2020/2021

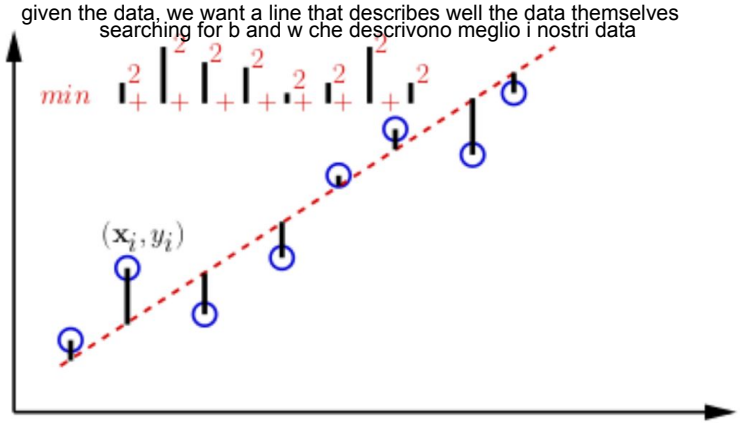
**Tatiana Tommasi**

# Summary

- Logistic Regression
- Gradient Descent
- Stochastic Gradient Descent

# Linear Regression

- Given pairs of input/output values, find $(b, \boldsymbol{w})$ to minimize the prediction errors (on average)
- I.e. using square loss on $\mathcal{H} = \{h_{(b,\boldsymbol{w})} : b \in \mathbb{R}, \boldsymbol{w} \in \mathbb{R}^d\}$: squared error on $(\boldsymbol{x}, y)$ is $(y - (b + \langle \boldsymbol{w}, \boldsymbol{x} \rangle))^2$

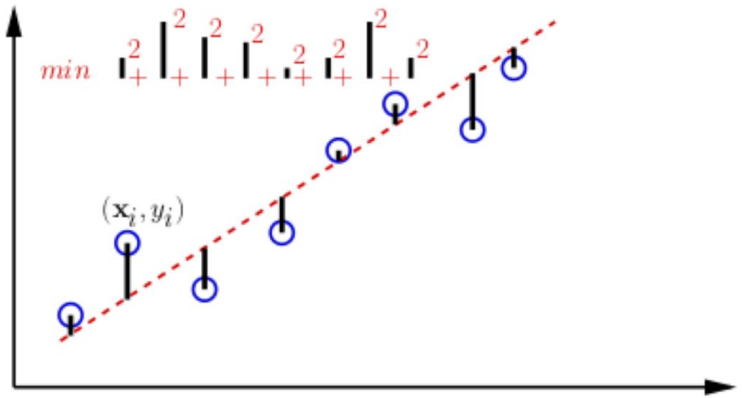





Slide Credit: Francesco Orabona

# Linear Regression

- Given pairs of input/output values, find $(b, \boldsymbol{w})$ to minimize the prediction errors (on average)
- I.e. using square loss on $\mathcal{H} = \{h_{(b,\boldsymbol{w})} : b \in \mathbb{R}, \boldsymbol{w} \in \mathbb{R}^d\}$: squared error on $(\boldsymbol{x}, y)$ is $(y - (b + \langle \boldsymbol{w}, \boldsymbol{x} \rangle))^2$




$f_\Theta$

al posto di h(b,w)

# Flashback: Loss Function

vogliamo minimizzare la loss function

- A loss function $L : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$ maps decisions to costs: $L(\hat{y}, y)$ defines the penalty paid for predicting $\hat{y}$ when the true value is $y$.
- Standard choice for regression: squared loss $L(\hat{y}, y) = (\hat{y} - y)^2$
  - Not the only possible choice
- Standard choice for classification: 0/1 loss

$$L_{0/1}(\hat{y}, y) = \begin{cases} 0 & \hat{y} = y \\ 1 & otherwise \end{cases}$$

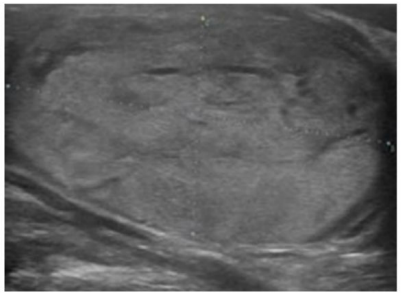
se la prediction è corretta



Slide Credit: Francesco Orabona

# Classification

What if we want to predict a category instead of a value?

$$f_\Theta(\text{  }) = \{0, 1\}$$

Possible solution: Do post-processing (e.g. thresholding) to convert linear regression to a binary output.

$\Rightarrow$ The solution is not necessarily an optimum anymore.

Instead: Modify the loss to minimize over categorical values directly.

# Logistic Regression

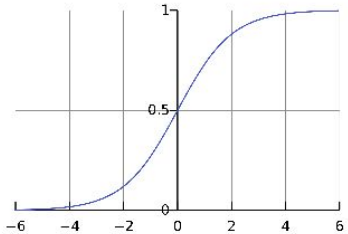
New loss:

mean square error

$$\ell_\Theta(\{x_i, y_i\}) = \sum_{i=1}^{n} (y_i - \sigma(\underbrace{ax_i + b}_{\text{linear}}))^2$$

Here, $\sigma$ is the nonlinear logistic sigmoid:

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$



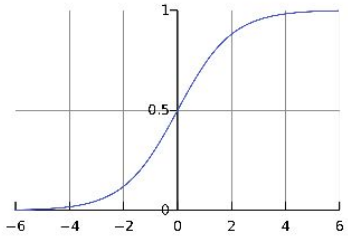
# Logistic Regression

New loss:

$$\ell_\Theta(\{x_i, y_i\}) = \sum_{i=1}^{n} (y_i - \sigma(\underbrace{ax_i + b}_{\text{linear}}))^2$$

Here, $\sigma$ is the nonlinear logistic sigmoid:

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$




$\sigma$ has a saturation effect as it maps $\mathbb{R} \mapsto (0, 1)$.

# Logistic Regression

y è 0 o 1

New loss:

$$\ell_\Theta(\{x_i, y_i\}) = \sum_{i=1}^{n} (y_i - \sigma(\underbrace{ax_i + b}_{\text{linear}}))^2$$ non-convex

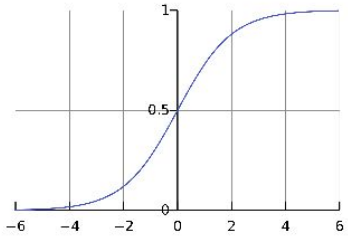
perchè a e b stanno dentro la funzione sigma

vogliamo cercare teta che minimizza la loss. in questo caso teta sono a e b (w e b)

Here, $\sigma$ is the nonlinear logistic sigmoid:

how to optimize non convex loss ?

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$




$\sigma$ has a saturation effect as it maps $\mathbb{R} \mapsto (0, 1)$.

# Logistic Regression

New loss:

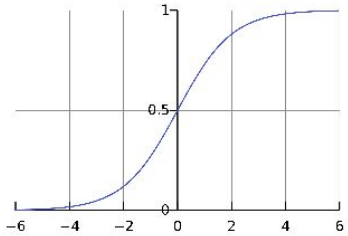
$$\ell_\Theta(\{x_i, y_i\}) = \sum_{i=1}^{n} c(x_i, y_i)\,, \quad \text{with}$$

$$c(x_i, y_i) = \begin{cases} -\ln(\sigma(ax_i + b)) & y_i = 1 \\ -\ln(1 - \sigma(ax_i + b)) & y_i = 0 \end{cases}$$

convex

Here, $\sigma$ is the nonlinear logistic sigmoid:

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$




$\sigma$ has a saturation effect as it maps $\mathbb{R} \mapsto (0, 1)$.

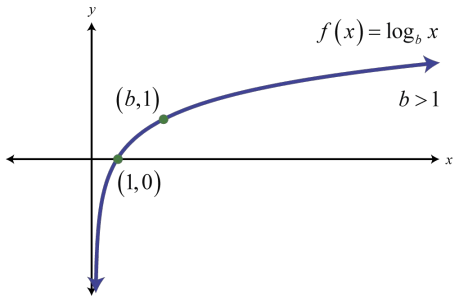
# Logistic Regression

New loss:

$$\ell_\Theta(\{x_i, y_i\}) = \sum_{i=1}^{n} c(x_i, y_i)\,, \quad \text{with}$$

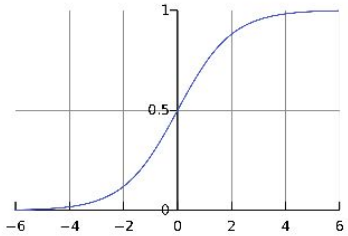
$$c(x_i, y_i) = \begin{cases} -\ln(\sigma(ax_i + b)) & y_i = 1 \\ -\ln(1 - \sigma(ax_i + b)) & y_i = 0 \end{cases}$$

convex




Here, $\sigma$ is the nonlinear logistic sigmoid:

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$




$\sigma$ has a saturation effect as it maps $\mathbb{R} \mapsto (0, 1)$.

# Logistic Regression

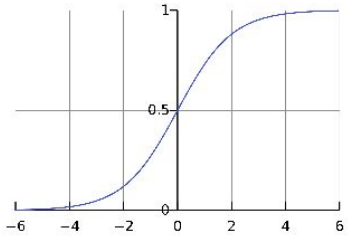
New loss:

$$\ell_\Theta(\{x_i, y_i\}) = \sum_{i=1}^{n} c(x_i, y_i)\,, \quad \text{with}$$

$$c(x_i, y_i) = -y_i \ln(\sigma(ax_i + b)) - (1 - y_i) \ln(1 - \sigma(ax_i + b)) \quad \text{convex}$$

Here, $\sigma$ is the nonlinear logistic sigmoid:

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$




$\sigma$ has a saturation effect as it maps $\mathbb{R} \mapsto (0, 1)$.

# Logistic Regression

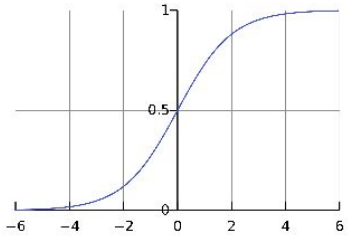
New loss:

$$\ell_\Theta(\{x_i, y_i\}) = -\sum_{i=1}^{n} y_i \ln(\sigma(ax_i + b)) + (1 - y_i) \ln(1 - \sigma(ax_i + b))$$

Here, $\sigma$ is the nonlinear logistic sigmoid:

vogliamo minimizzare il loss, cioè cerchiamo a e b che ottimizzano la nostra prediction

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$



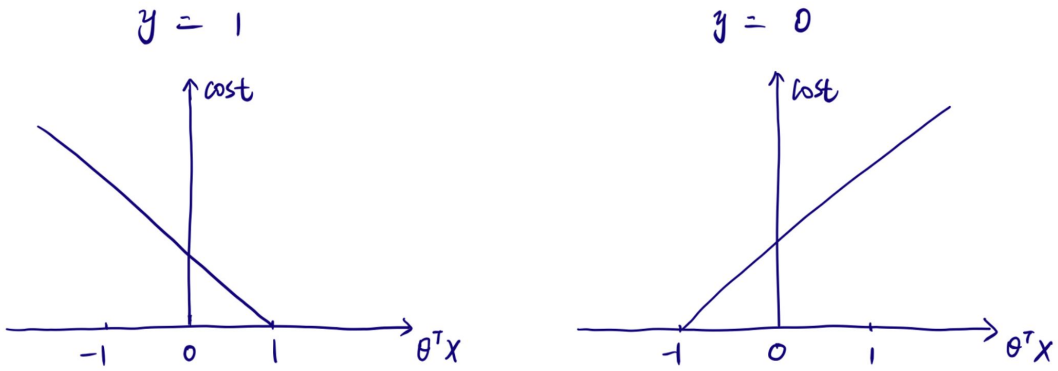

$\sigma$ has a saturation effect as it maps $\mathbb{R} \mapsto (0, 1)$.

# What was the loss in SVM?

stessa cosa di f

$$h_\theta(x) = \begin{cases} 1 & \text{if } \theta^T x >= 0 \\ 0 & \text{otherwise} \end{cases}$$

$$Cost(h_\theta(x), y) = \begin{cases} max(0, 1 - \theta^T x) & \text{if } y = 1 \\ max(0, 1 + \theta^T x) & \text{if } y = 0 \end{cases}$$

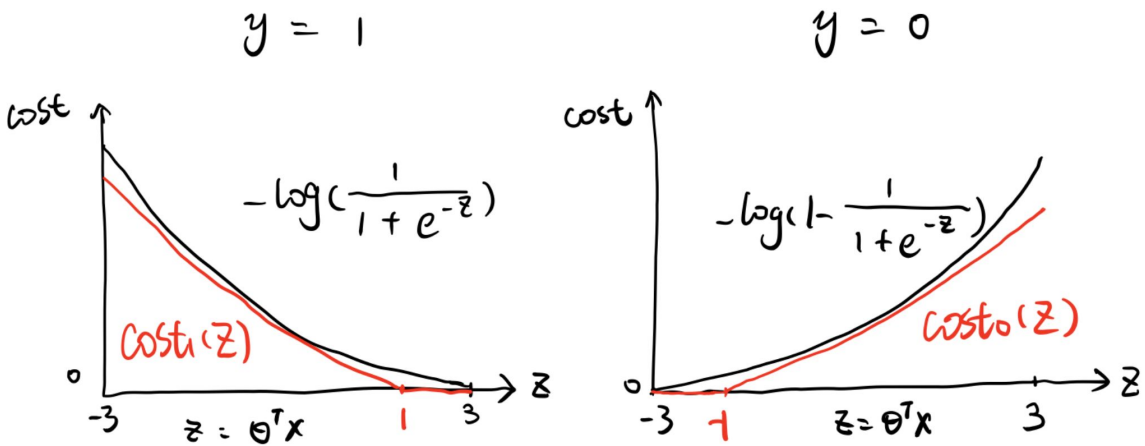



https://towardsdatascience.com/optimization-loss-function-under-the-hood-part-iii-5dff33fa015d

# What was the loss in SVM?

$$h_\theta(x) = \begin{cases} 1 & \text{if } \theta^T x >= 0 \\ 0 & \text{otherwise} \end{cases}$$

$$Cost(h_\theta(x), y) = \begin{cases} max(0, 1 - \theta^T x) & \text{if } y = 1 \\ max(0, 1 + \theta^T x) & \text{if } y = 0 \end{cases}$$




https://towardsdatascience.com/optimization-loss-function-under-the-hood-part-iii-5dff33fa015d

# What was the loss in SVM?

$$h_\theta(x) = \begin{cases} 1 & \text{if } \theta^T x >= 0 \\ 0 & \text{otherwise} \end{cases}$$

$$Cost(h_\theta(x), y) = \begin{cases} max(0, 1 - \theta^T x) & \text{if } y = 1 \\ max(0, 1 + \theta^T x) & \text{if } y = 0 \end{cases}$$

$$J(\theta) = \sum_{i=1}^{m} y^{(i)} Cost_1(\theta^T(x^{(i)}) + (1 - y^{(i)}) Cost_0(\theta^T(x^{(i)})$$

$$J(\theta) = \sum_{i=1}^{m} y^{(i)} max(0, 1 - \theta^T x) + (1 - y^{(i)}) max(0, 1 + \theta^T x)$$

convex

$$m = number\ of\ samples$$



https://towardsdatascience.com/optimization-loss-function-under-the-hood-part-iii-5dff33fa015d

# What was the loss in SVM?

$$h_\theta(x) = \begin{cases} 1 & \text{if } \theta^T x >= 0 \\ 0 & \text{otherwise} \end{cases}$$

$$Cost(h_\theta(x), y) = \begin{cases} max(0, 1 - \theta^T x) & \text{if } y = 1 \\ max(0, 1 + \theta^T x) & \text{if } y = 0 \end{cases}$$

$$J(\theta) = C[\sum_{i=1}^{m} y^{(i)} Cost_1(\theta^T(x^{(i)}) + (1 - y^{(i)}) Cost_0(\theta^T(x^{(i)})] + \frac{1}{2}\sum_{j=1}^{n} \theta_j^2$$

$$m = number\ of\ samples, \quad n = number\ of\ features$$



https://towardsdatascience.com/optimization-loss-function-under-the-hood-part-iii-5dff33fa015d

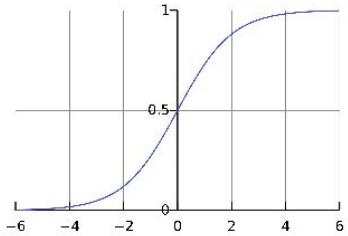
# Logistic Regression

New loss:

$$\ell_\Theta(\{x_i, y_i\}) = -\sum_{i=1}^{n} y_i \ln(\sigma(ax_i + b)) + (1 - y_i) \ln(1 - \sigma(ax_i + b))$$

Here, $\sigma$ is the nonlinear logistic sigmoid:

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$




$\sigma$ has a saturation effect as it maps $\mathbb{R} \mapsto (0, 1)$.

# Logistic Regression: Finding a Solution

Since the loss is convex, the first-order conditions apply:

$$\nabla_\Theta \ell_\Theta = 0$$

# Logistic Regression: Finding a Solution

Since the loss is convex, the first-order conditions apply:

$$\nabla_\Theta \sum_{i=1}^{n} y_i \ln(\sigma(ax_i + b)) + (1 - y_i) \ln(1 - \sigma(ax_i + b)) = 0$$

where $\Theta = \{a, b\}$.

# Logistic Regression: Finding a Solution

Since the loss is convex, the first-order conditions apply:

$$\nabla_\Theta \sum_{i=1}^{n} y_i \ln(\sigma(ax_i + b)) + (1 - y_i) \ln(1 - \sigma(ax_i + b)) = 0$$

where $\Theta = \{a, b\}$.

Consider the gradient of each term in the summation:

$$\nabla_\Theta \left( y_i \ln(\sigma(ax_i + b)) + (1 - y_i) \ln(1 - \sigma(ax_i + b)) \right)$$

# Logistic Regression: Finding a Solution

Since the loss is convex, the first-order conditions apply:

$$\nabla_\Theta \sum_{i=1}^{n} y_i \ln(\sigma(ax_i + b)) + (1 - y_i) \ln(1 - \sigma(ax_i + b)) = 0$$

where $\Theta = \{a, b\}$.

Consider the gradient of each term in the summation:

$$\nabla_\Theta \left( y_i \ln(\sigma(ax_i + b)) + (1 - y_i) \ln(1 - \sigma(ax_i + b)) \right)$$

$$\nabla_\Theta y_i \ln(\sigma(ax_i + b)) + \nabla_\Theta (1 - y_i) \ln(1 - \sigma(ax_i + b))$$

$$y_i \nabla_\Theta \ln(\sigma(ax_i + b)) + (1 - y_i) \nabla_\Theta \ln(1 - \sigma(ax_i + b))$$

# Logistic Regression: Finding a Solution

Since the loss is convex, the first-order conditions apply:

$$\nabla_\Theta \sum_{i=1}^{n} y_i \ln(\sigma(ax_i + b)) + (1 - y_i) \ln(1 - \sigma(ax_i + b)) = 0$$

where $\Theta = \{a, b\}$.

Consider the gradient of each term in the summation:

$$\nabla_\Theta \left( y_i \ln(\sigma(ax_i + b)) + (1 - y_i) \ln(1 - \sigma(ax_i + b)) \right)$$

$$\nabla_\Theta y_i \ln(\sigma(ax_i + b)) + \nabla_\Theta (1 - y_i) \ln(1 - \sigma(ax_i + b))$$

$$y_i \nabla_\Theta \underbrace{\ln(\sigma(ax_i + b))}_{f(g(h(\Theta)))} + (1 - y_i) \nabla_\Theta \ln(1 - \sigma(ax_i + b))$$

# Logistic Regression: Finding a Solution

$$y_i \nabla_\Theta \underbrace{\ln(\sigma(ax_i + b))}_{f(g(h(\Theta)))} + (1 - y_i)\nabla_\Theta \ln(1 - \sigma(ax_i + b))$$

Apply the chain rule to each partial derivative:

$$\frac{\partial}{\partial a} f(g(h(a, b))) = \frac{\partial f}{\partial g} \cdot \frac{\partial g}{\partial h} \cdot \frac{\partial h}{\partial a}$$

# Logistic Regression: Finding a Solution

$$y_i \nabla_\Theta \underbrace{\ln(\sigma(ax_i + b))}_{f(g(h(\Theta)))} + (1 - y_i)\nabla_\Theta \ln(1 - \sigma(ax_i + b))$$

Apply the chain rule to each partial derivative:

$$\frac{\partial}{\partial a} f(g(h(a, b))) = \frac{\partial f}{\partial g} \cdot \frac{\partial g}{\partial h} \cdot \frac{\partial h}{\partial a} \quad = \frac{\partial f}{\partial g} \cdot \frac{\partial g}{\partial h} \cdot \frac{\partial}{\partial a} a x_i + b$$

# Logistic Regression: Finding a Solution

$$y_i \nabla_\Theta \underbrace{\ln(\sigma(ax_i + b))}_{f(g(h(\Theta)))} + (1 - y_i)\nabla_\Theta \ln(1 - \sigma(ax_i + b))$$

Apply the chain rule to each partial derivative:

$$\frac{\partial}{\partial a} f(g(h(a, b))) = \frac{\partial f}{\partial g} \cdot \frac{\partial g}{\partial h} \cdot \frac{\partial h}{\partial a} = \frac{\partial f}{\partial g} \cdot \frac{\partial g}{\partial h} \cdot \frac{\partial}{\partial a} ax_i + b$$

$$= \frac{\partial f}{\partial g} \cdot \frac{\partial \sigma(ax_i + b)}{\partial (ax_i + b)} \cdot x_i$$

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

# Logistic Regression: Finding a Solution

$$y_i \nabla_\Theta \underbrace{\ln(\sigma(ax_i + b))}_{f(g(h(\Theta)))} + (1 - y_i)\nabla_\Theta \ln(1 - \sigma(ax_i + b))$$

Apply the chain rule to each partial derivative:

$$\frac{\partial}{\partial a} f(g(h(a, b))) = \frac{\partial f}{\partial g} \cdot \frac{\partial g}{\partial h} \cdot \frac{\partial h}{\partial a} = \frac{\partial f}{\partial g} \cdot \frac{\partial g}{\partial h} \cdot \frac{\partial}{\partial a} ax_i + b$$

$$= \frac{\partial f}{\partial g} \cdot \frac{\partial \sigma(ax_i + b)}{\partial (ax_i + b)} \cdot x_i = \frac{\partial f}{\partial g} \cdot \frac{\partial}{\partial (ax_i + b)} \frac{1}{1 + e^{-(ax_i + b)}} \cdot x_i$$

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

# Logistic Regression: Finding a Solution

$$y_i \nabla_\Theta \underbrace{\ln(\sigma(ax_i + b))}_{f(g(h(\Theta)))} + (1 - y_i)\nabla_\Theta \ln(1 - \sigma(ax_i + b))$$

Apply the chain rule to each partial derivative:

$$\begin{aligned}
\frac{\partial}{\partial a} f(g(h(a, b))) &= \frac{\partial f}{\partial g} \cdot \frac{\partial g}{\partial h} \cdot \frac{\partial h}{\partial a} \quad = \frac{\partial f}{\partial g} \cdot \frac{\partial g}{\partial h} \cdot \frac{\partial}{\partial a} a x_i + b \\
&= \frac{\partial f}{\partial g} \cdot \frac{\partial \sigma(ax_i + b)}{\partial (ax_i + b)} \cdot x_i \quad = \frac{\partial f}{\partial g} \cdot \frac{\partial}{\partial (ax_i + b)} \frac{1}{1 + e^{-(ax_i + b)}} \cdot x_i \\
&= \frac{\partial f}{\partial g} \cdot \frac{e^{-(ax_i + b)}}{(1 + e^{-(ax_i + b)})^2} \cdot x_i
\end{aligned}$$

# Logistic Regression: Finding a Solution

$$y_i \nabla_\Theta \underbrace{\ln(\sigma(ax_i + b))}_{f(g(h(\Theta)))} + (1 - y_i)\nabla_\Theta \ln(1 - \sigma(ax_i + b))$$

Apply the chain rule to each partial derivative:

$$\begin{aligned}
\frac{\partial}{\partial a} f(g(h(a,b))) &= \frac{\partial f}{\partial g} \cdot \frac{\partial g}{\partial h} \cdot \frac{\partial h}{\partial a} = \frac{\partial f}{\partial g} \cdot \frac{\partial g}{\partial h} \cdot \frac{\partial}{\partial a} a x_i + b \\
&= \frac{\partial f}{\partial g} \cdot \frac{\partial \sigma(ax_i + b)}{\partial (ax_i + b)} \cdot x_i = \frac{\partial f}{\partial g} \cdot \frac{\partial}{\partial (ax_i + b)} \frac{1}{1 + e^{-(ax_i + b)}} \cdot x_i \\
&= \frac{\partial f}{\partial g} \cdot \frac{e^{-(ax_i + b)}}{(1 + e^{-(ax_i + b)})^2} \cdot x_i = \frac{\partial f}{\partial g} \cdot \frac{1}{1 + e^{-(ax_i + b)}} \frac{e^{-(ax_i + b)}}{1 + e^{-(ax_i + b)}} \cdot x_i
\end{aligned}$$

# Logistic Regression: Finding a Solution

$$y_i \nabla_\Theta \underbrace{\ln(\sigma(ax_i + b))}_{f(g(h(\Theta)))} + (1 - y_i)\nabla_\Theta \ln(1 - \sigma(ax_i + b))$$

Apply the chain rule to each partial derivative:

$$\frac{\partial}{\partial a} f(g(h(a, b))) = \frac{\partial f}{\partial g} \cdot \frac{\partial g}{\partial h} \cdot \frac{\partial h}{\partial a} = \frac{\partial f}{\partial g} \cdot \frac{\partial g}{\partial h} \cdot \frac{\partial}{\partial a} ax_i + b$$

$$= \frac{\partial f}{\partial g} \cdot \frac{\partial \sigma(ax_i + b)}{\partial (ax_i + b)} \cdot x_i = \frac{\partial f}{\partial g} \cdot \frac{\partial}{\partial (ax_i + b)} \frac{1}{1 + e^{-(ax_i + b)}} \cdot x_i$$

$$= \frac{\partial f}{\partial g} \cdot \frac{e^{-(ax_i + b)}}{(1 + e^{-(ax_i + b)})^2} \cdot x_i = \frac{\partial f}{\partial g} \cdot \frac{1}{1 + e^{-(ax_i + b)}} \frac{e^{-(ax_i + b)}}{1 + e^{-(ax_i + b)}} \cdot x_i$$

$$= \frac{\partial f}{\partial g} \cdot \frac{1}{1 + e^{-(ax_i + b)}} \frac{(1 + e^{-(ax_i + b)}) - 1}{1 + e^{-(ax_i + b)}} \cdot x_i = \frac{\partial f}{\partial g} \cdot \frac{1}{1 + e^{-(ax_i + b)}} (1 - \frac{1}{1 + e^{-(ax_i + b)}}) \cdot x_i$$

# Logistic Regression: Finding a Solution

$$y_i \nabla_\Theta \underbrace{\ln(\sigma(ax_i + b))}_{f(g(h(\Theta)))} + (1 - y_i)\nabla_\Theta \ln(1 - \sigma(ax_i + b))$$

Apply the chain rule to each partial derivative:

$$\frac{\partial}{\partial a} f(g(h(a, b))) = \frac{\partial f}{\partial g} \cdot \frac{\partial g}{\partial h} \cdot \frac{\partial h}{\partial a} = \frac{\partial f}{\partial g} \cdot \frac{\partial g}{\partial h} \cdot \frac{\partial}{\partial a} a x_i + b$$

$$= \frac{\partial f}{\partial g} \cdot \frac{\partial \sigma(ax_i + b)}{\partial (ax_i + b)} \cdot x_i = \frac{\partial f}{\partial g} \cdot \frac{\partial}{\partial (ax_i + b)} \frac{1}{1 + e^{-(ax_i + b)}} \cdot x_i$$

$$= \frac{\partial f}{\partial g} \cdot \frac{e^{-(ax_i + b)}}{(1 + e^{-(ax_i + b)})^2} \cdot x_i = \frac{\partial f}{\partial g} \cdot \frac{1}{1 + e^{-(ax_i + b)}} \frac{e^{-(ax_i + b)}}{1 + e^{-(ax_i + b)}} \cdot x_i$$

$$= \frac{\partial f}{\partial g} \cdot \frac{1}{1 + e^{-(ax_i + b)}} \frac{(1 + e^{-(ax_i + b)}) - 1}{1 + e^{-(ax_i + b)}} \cdot x_i = \frac{\partial f}{\partial g} \cdot \frac{1}{1 + e^{-(ax_i + b)}} (1 - \frac{1}{1 + e^{-(ax_i + b)}}) \cdot x_i$$

$$= \frac{\partial f}{\partial g} \cdot \sigma(ax_i + b)(1 - \sigma(ax_i + b)) \cdot x_i$$

# Logistic Regression: Finding a Solution

$$y_i \nabla_\Theta \underbrace{\ln(\sigma(ax_i + b))}_{f(g(h(\Theta)))} + (1 - y_i)\nabla_\Theta \ln(1 - \sigma(ax_i + b))$$

Apply the chain rule to each partial derivative:

$$\frac{\partial}{\partial a} f(g(h(a, b))) = \frac{\partial f}{\partial g} \cdot \frac{\partial g}{\partial h} \cdot \frac{\partial h}{\partial a} = \frac{\partial f}{\partial g} \cdot \frac{\partial g}{\partial h} \cdot \frac{\partial}{\partial a} ax_i + b$$

$$= \frac{1}{\sigma(ax_i + b)} \cdot \sigma(ax_i + b)(1 - \sigma(ax_i + b)) \cdot x_i$$

$$= \frac{\partial f}{\partial g} \cdot \sigma(ax_i + b)(1 - \sigma(ax_i + b)) \cdot x_i$$

# Logistic Regression: Finding a Solution

$$y_i \nabla_\Theta \underbrace{\ln(\sigma(ax_i + b))}_{f(g(h(\Theta)))} + (1 - y_i)\nabla_\Theta \ln(1 - \sigma(ax_i + b))$$

Apply the chain rule to each partial derivative:

$$\frac{\partial}{\partial a} f(g(h(a, b))) = \frac{\partial f}{\partial g} \cdot \frac{\partial g}{\partial h} \cdot \frac{\partial h}{\partial a} = \frac{\partial f}{\partial g} \cdot \frac{\partial g}{\partial h} \cdot \frac{\partial}{\partial a} ax_i + b$$

$$= \frac{1}{\sigma(ax_i + b)} \cdot \sigma(ax_i + b)(1 - \sigma(ax_i + b)) \cdot x_i$$

$$= \frac{1}{\sigma(ax_i + b)} \cdot \sigma(ax_i + b)(1 - \sigma(ax_i + b)) \cdot x_i$$

$$= (1 - \sigma(ax_i + b))x_i$$

# Logistic Regression: Finding a Solution

Since the loss is convex, the first-order conditions apply:

$$\nabla_\Theta \sum_{i=1}^{n} y_i \ln(\sigma(ax_i + b)) + (1 - y_i) \ln(1 - \sigma(ax_i + b)) = 0$$

where $\Theta = \{a, b\}$.

Consider the gradient of each term in the summation:

$$y_i \nabla_\Theta \underbrace{\ln(\sigma(ax_i + b))}_{f(g(h(\Theta)))} + (1 - y_i) \nabla_\Theta \ln(1 - \sigma(ax_i + b))$$

Apply the chain rule to each partial derivative:

$$\frac{\partial}{\partial a} f(g(h(a, b))) = \frac{\partial f}{\partial g} \cdot \frac{\partial g}{\partial h} \cdot \frac{\partial h}{\partial a}$$

# Logistic Regression: Finding a Solution

Since the loss is convex, the first-order conditions apply:

$$\nabla_\Theta \sum_{i=1}^{n} y_i \ln(\sigma(ax_i + b)) + (1 - y_i) \ln(1 - \sigma(ax_i + b)) = 0$$

where $\Theta = \{a, b\}$.

Consider the gradient of each term in the summation:

$$y_i \nabla_\Theta \underbrace{\ln(\sigma(ax_i + b))}_{f(g(h(\Theta)))} + (1 - y_i) \nabla_\Theta \ln(1 - \sigma(ax_i + b))$$

Apply the chain rule to each partial derivative:

$$\frac{\partial}{\partial a} \ln(\sigma(ax_i + b)) = (1 - \sigma(ax_i + b))x_i$$

# Logistic Regression: Finding a Solution

Since the loss is convex, the first-order conditions apply:

$$\nabla_\Theta \sum_{i=1}^{n} y_i \ln(\sigma(ax_i + b)) + (1 - y_i) \ln(1 - \sigma(ax_i + b)) = 0$$

where $\Theta = \{a, b\}$.

Consider the gradient of each term in the summation:

$$y_i \nabla_\Theta \underbrace{\ln(\sigma(ax_i + b))}_{f(g(h(\Theta)))} + (1 - y_i) \nabla_\Theta \ln(1 - \sigma(ax_i + b))$$

Apply the chain rule to each partial derivative:

$$\frac{\partial}{\partial a} \ln(\sigma(ax_i + b)) = (1 - \sigma(ax_i + b))x_i$$

And similarly for the second term and for all parameters.



Slide Credit: E. Rodolà

# Logistic Regression: Finding a Solution

By looking at the partial derivative:

$$\frac{\partial}{\partial a} \ln(\sigma(ax_i + b)) = (1 - \sigma(ax_i + b))x_i$$

we see that the parameters enter the gradient in a nonlinear way.

# Logistic Regression: Finding a Solution

By looking at the partial derivative:

$$\frac{\partial}{\partial a} \ln(\sigma(ax_i + b)) = (1 - \sigma(ax_i + b))x_i$$

we see that the parameters enter the gradient in a nonlinear way.

Thus:

- $\nabla \ell_\Theta = 0$ is not a linear system to solve for the parameters.
- $\nabla \ell_\Theta = 0$ is a transcendental equation $\Rightarrow$ no analytical solution.

| model | loss | solution |
|---|---|---|
| linear regression | convex | least squares |
| logistic regression | convex | nonlinear optimization |

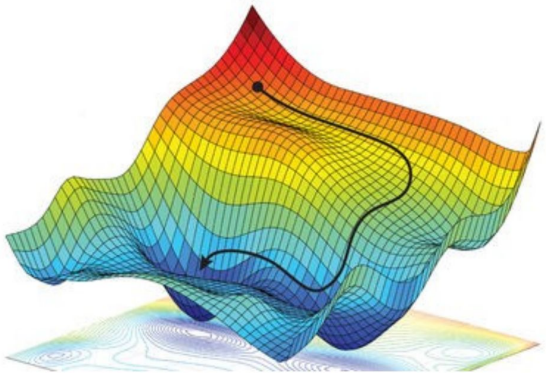
# Gradient Descent: Intuition

Gradient descent is a first-order iterative minimization algorithm.

# Gradient Descent: Intuition

Gradient descent is a first-order iterative minimization algorithm.

Example of a loss function $\ell_{\boldsymbol{\Theta}} : \mathbb{R}^2 \to \mathbb{R}$:

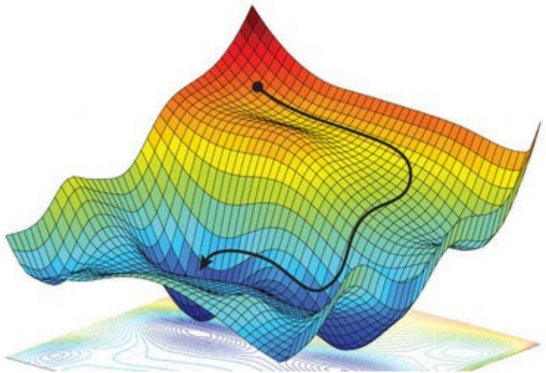


# Gradient Descent: Intuition

Gradient descent is a first-order iterative minimization algorithm.

Example of a loss function $\ell_{\mathbf{\Theta}} : \mathbb{R}^2 \to \mathbb{R}$:

the shape of the loss gives us the direction where we should move



**Overall idea:** Move where the function decreases the most.

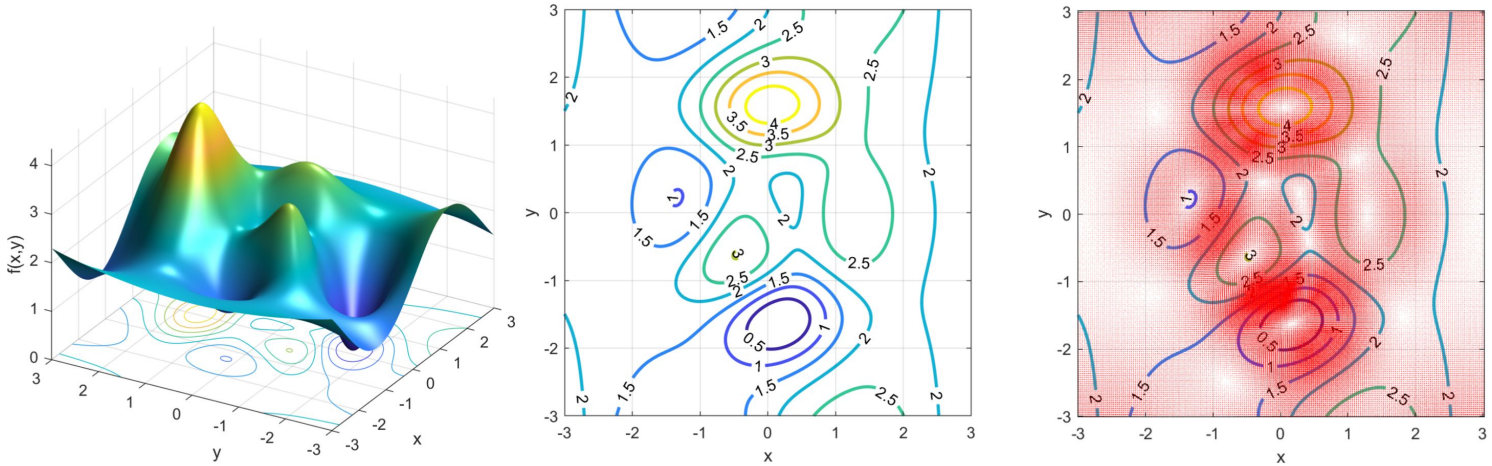
1. Start from some point $\mathbf{\Theta}^{(0)} \in \mathbb{R}^2$.
2. Iteratively compute:

$$\mathbf{\Theta}^{(t+1)} = \mathbf{\Theta}^{(t)} - \alpha \nabla \ell_{\mathbf{\Theta}^{(t)}}$$

3. Stop when a minimum is reached.

# Gradient Descent

$$\mathbf{x}^{(t+1)} = \mathbf{x}^{(t)} - \alpha \nabla f(\mathbf{x}^{(t)})$$

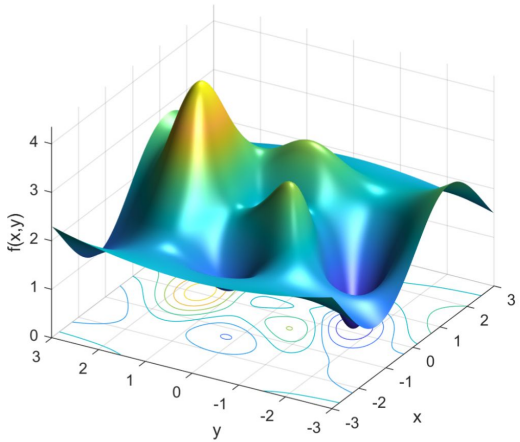


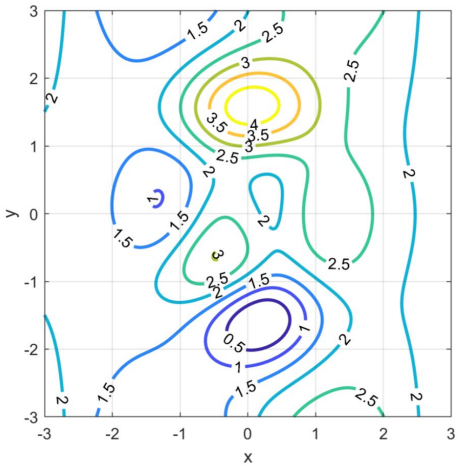


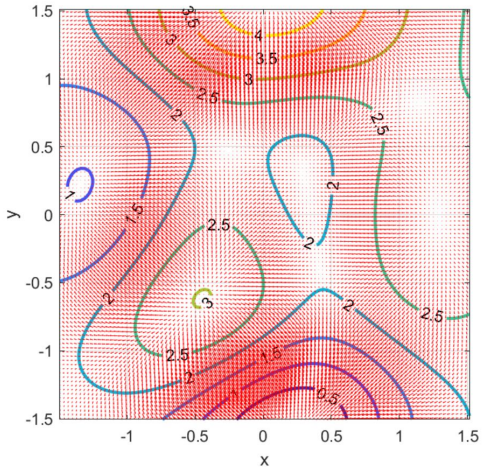

Slide Credit: E. Rodolà

# Gradient Descent

$$\mathbf{x}^{(t+1)} = \mathbf{x}^{(t)} - \alpha \nabla f(\mathbf{x}^{(t)})$$



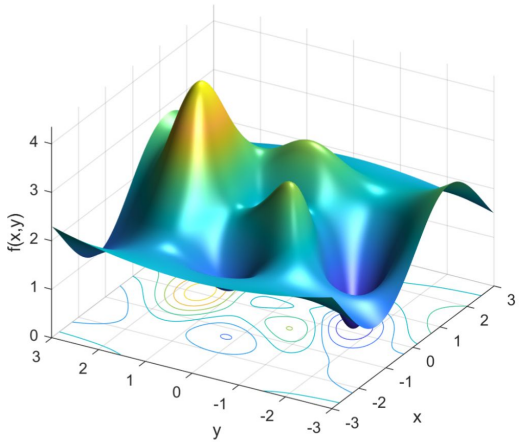



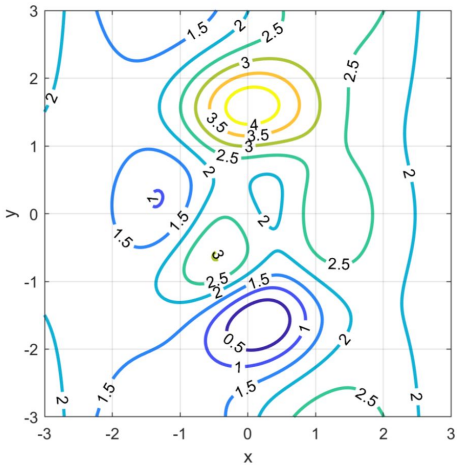


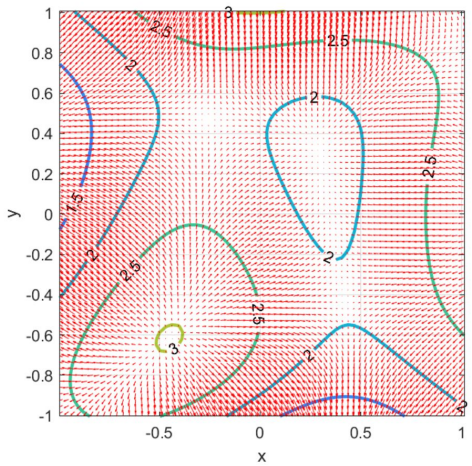

Slide Credit: E. Rodolà

# Gradient Descent

$$\mathbf{x}^{(t+1)} = \mathbf{x}^{(t)} - \alpha \nabla f(\mathbf{x}^{(t)})$$



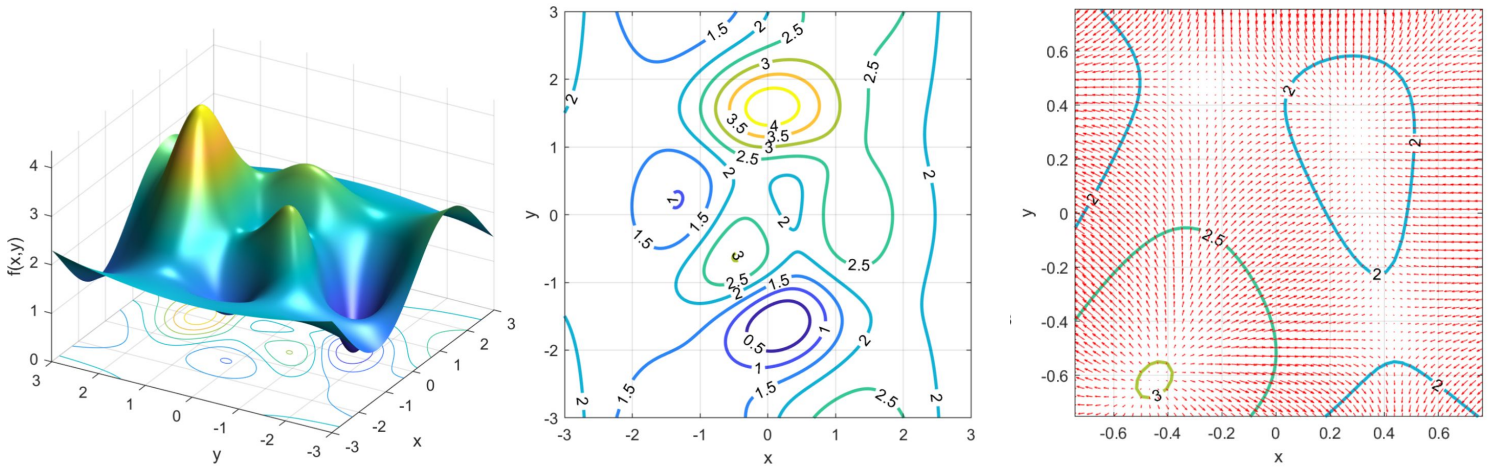






Slide Credit: E. Rodolà

# Gradient Descent

$$\mathbf{x}^{(t+1)} = \mathbf{x}^{(t)} - \alpha \nabla f(\mathbf{x}^{(t)})$$



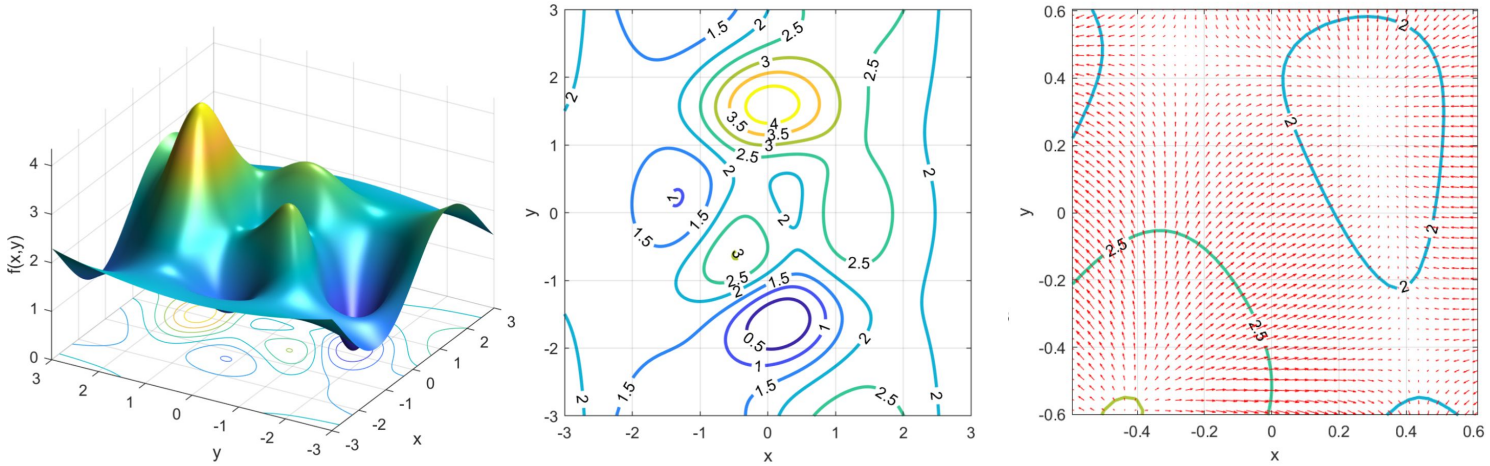



Slide Credit: E. Rodolà

# Gradient Descent

## Gradient descent

$$\mathbf{x}^{(t+1)} = \mathbf{x}^{(t)} - \alpha \nabla f(\mathbf{x}^{(t)})$$

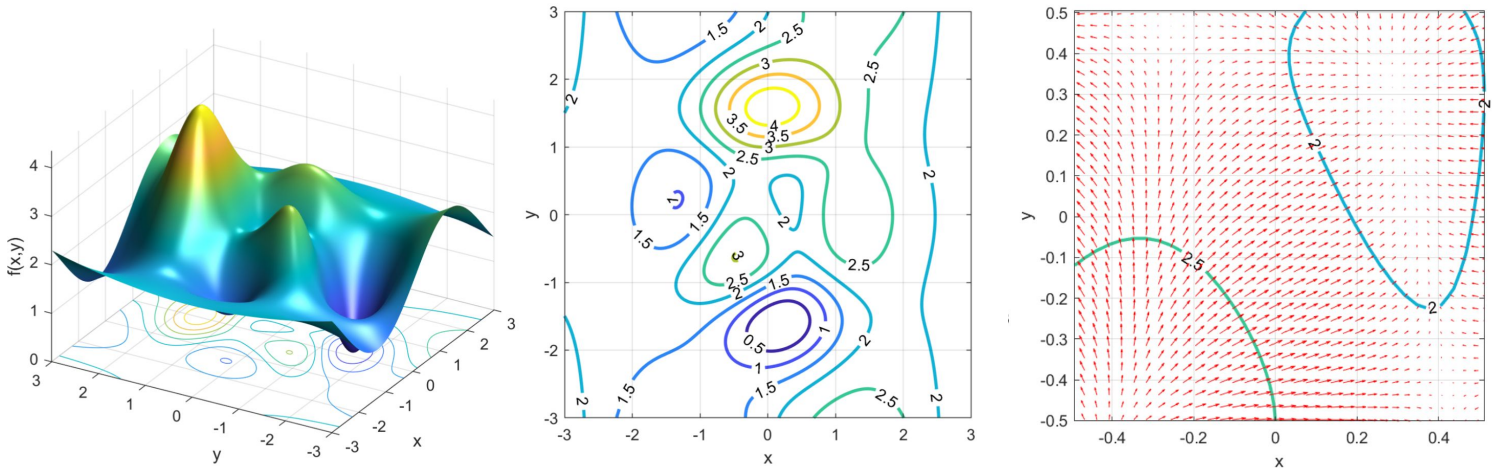





Slide Credit: E. Rodolà

# Gradient Descent

$$\mathbf{x}^{(t+1)} = \mathbf{x}^{(t)} - \alpha \nabla f(\mathbf{x}^{(t)})$$



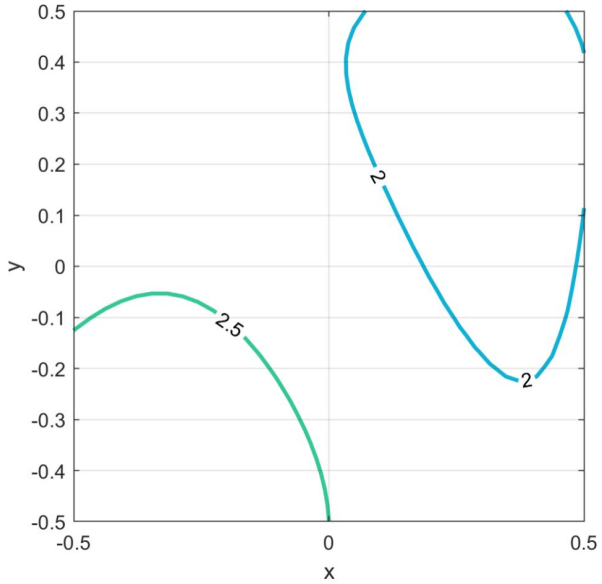



Slide Credit: E. Rodolà

# Gradient Descent

$$\mathbf{x}^{(t+1)} = \mathbf{x}^{(t)} - \alpha \nabla f(\mathbf{x}^{(t)})$$

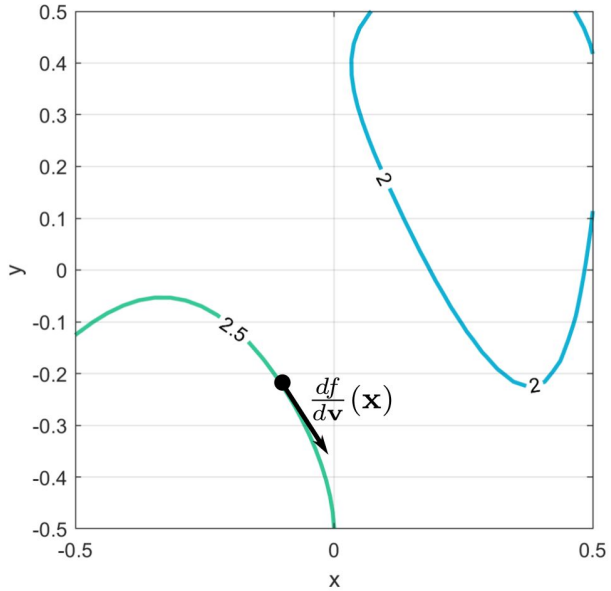
The gradient is orthogonal to level curves / level surfaces.



# Gradient Descent

$$\mathbf{x}^{(t+1)} = \mathbf{x}^{(t)} - \alpha \nabla f(\mathbf{x}^{(t)})$$

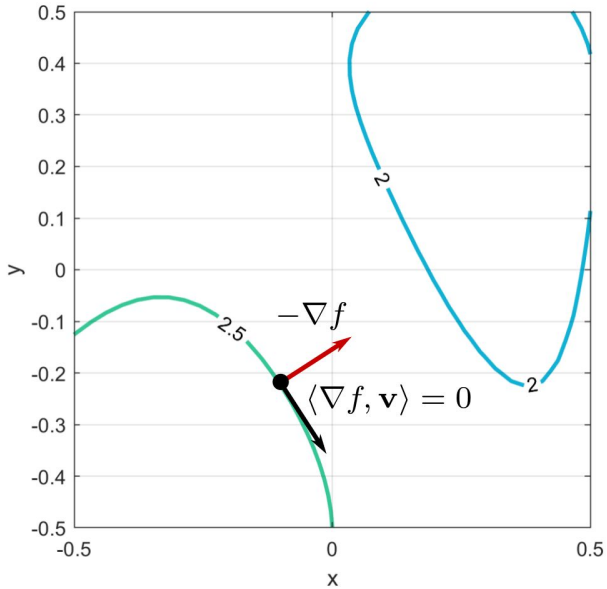
The gradient is orthogonal to level curves / level surfaces.



# Gradient Descent

$$\mathbf{x}^{(t+1)} = \mathbf{x}^{(t)} - \alpha \nabla f(\mathbf{x}^{(t)})$$

The gradient is orthogonal to level curves / level surfaces.




The directional derivative is zero along isocurves.

# Gradient Descent: Differentiability

$$\mathbf{x}^{(t+1)} = \mathbf{x}^{(t)} - \alpha \nabla f(\mathbf{x}^{(t)})$$

Gradient descent requires $f$ to be differentiable at all points.

Recall:

$f$ has partial (or even directional) derivatives $\not\Rightarrow$ $f$ is differentiable

$f$ has continuous gradient $\Rightarrow$ $f$ is differentiable

# Gradient Descent: Differentiability

$$\mathbf{x}^{(t+1)} = \mathbf{x}^{(t)} - \alpha \nabla f(\mathbf{x}^{(t)})$$

Gradient descent requires $f$ to be differentiable at all points.

**Example in $\mathbb{R}^2$:** $f(x,y) = \begin{cases} 0 & \text{at } (0,0) \\ \frac{x^2 y}{x^2+y^2} & \text{elsewhere} \end{cases}$

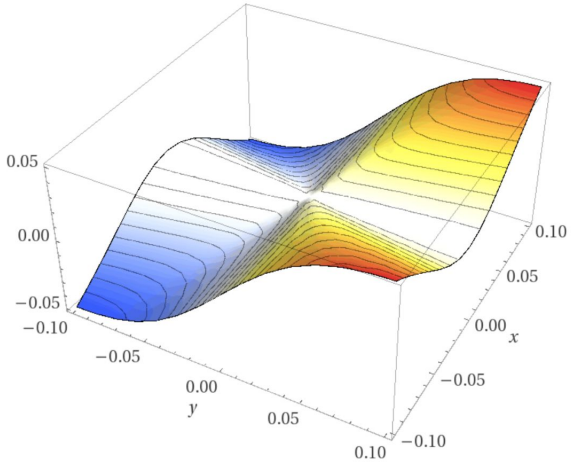
$\frac{\partial}{\partial y} f(x,0) = 1$ for $x \neq 0$

$\frac{\partial}{\partial y} f(0,y) = 0$ for $y \neq 0$

$\Rightarrow \frac{\partial}{\partial y} f(x,y)$ discontinuous at $(0,0)$

the partial derivatives of $f$ are defined everywhere, but are discontinuous at the origin

$\Rightarrow f$ is not differentiable



# Gradient Descent: Stationary Points

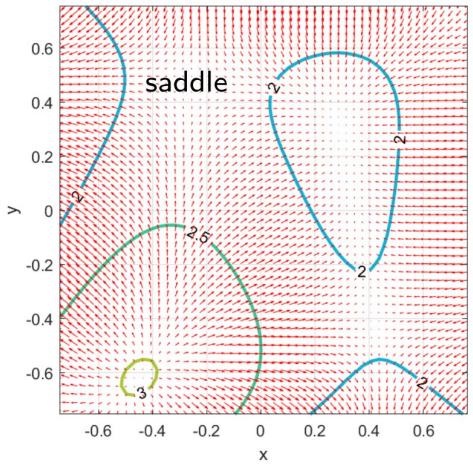
A stationary point is such that:

$$\mathbf{x}^{(t+1)} = \mathbf{x}^{(t)} - \alpha \cancelto{0}{\nabla f(\mathbf{x}^{(t)})}$$

Gradient descent "gets stuck" at stationary points.

However:

- Stationary point $\not\Rightarrow$ local minimum






Slide Credit: E. Rodolà

# Gradient Descent: Stationary Points

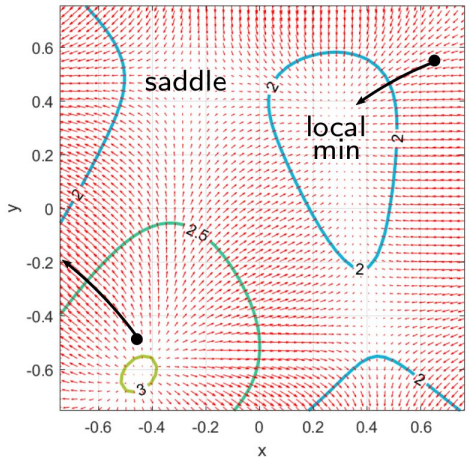
A stationary point is such that:

$$\mathbf{x}^{(t+1)} = \mathbf{x}^{(t)} - \alpha \cancelto{0}{\nabla f(\mathbf{x}^{(t)})}$$

Gradient descent "gets stuck" at stationary points.

However:

- Stationary point $\not\Rightarrow$ local minimum
- Which stationary point depends on the initialization.



# Gradient Descent: Learning Rate

$$\mathbf{x}^{(t+1)} = \mathbf{x}^{(t)} - \alpha \nabla f(\mathbf{x}^{(t)})$$

The parameter $\alpha > 0$ is also called learning rate in ML.

**Remark:** The length of a step is not simply $\alpha$, but $\alpha \|\nabla f\|$.

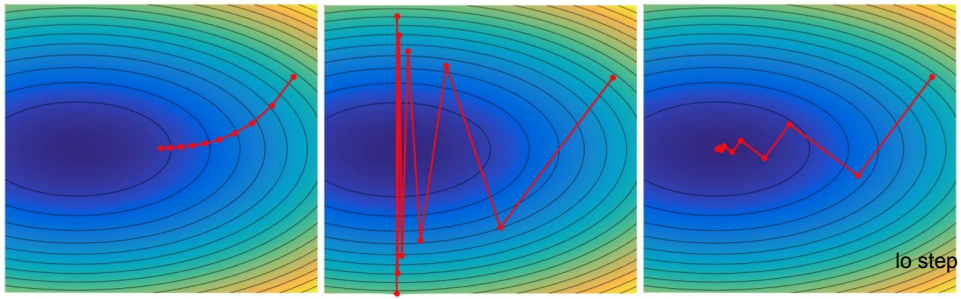
# Gradient Descent: Learning Rate

$$\mathbf{x}^{(t+1)} = \mathbf{x}^{(t)} - \alpha \nabla f(\mathbf{x}^{(t)})$$

The parameter $\alpha > 0$ is also called learning rate in ML.

**Remark:** The length of a step is not simply $\alpha$, but $\alpha \|\nabla f\|$.

- Too small: slow convergence speed
- Too big: risk of overshooting
- Optimal values can be found via line search algorithms



small $\alpha$ large $\alpha$ optimal $\alpha$

il miglior alpha si trova quando tutti gli step sono ortogonali fra loro

lo step si ferma tangente all'isocurva

$$\arg\min_{\alpha} f(\mathbf{x}^{(t)} - \alpha \nabla f(\mathbf{x}^{(t)}))$$

# Decay and Momentum

The learning rate can be adaptive or follow a schedule.

- Decrease $\alpha$ according to a decay parameter $\rho$:

  Examples:

$$\alpha^{(t+1)} = (1 - \frac{t}{\rho})\alpha^{(0)} + \frac{t}{\rho}\alpha^{(\rho)}\,, \quad \alpha^{(t+1)} = \frac{\alpha^{(t)}}{1+\rho t}\,, \quad \alpha^{(t+1)} = \alpha^{(0)} e^{-\rho t}$$

# Decay and Momentum

The learning rate can be adaptive or follow a schedule.

- Decrease $\alpha$ according to a decay parameter $\rho$:
  Examples:

$$\alpha^{(t+1)} = (1-\frac{t}{\rho})\alpha^{(0)} + \frac{t}{\rho}\alpha^{(\rho)}\,, \quad \alpha^{(t+1)} = \frac{\alpha^{(t)}}{1+\rho t}\,, \quad \alpha^{(t+1)} = \alpha^{(0)} e^{-\rho t}$$

- Accumulate past gradients and keep moving in their direction:

$$\mathbf{v}^{(t+1)} = \lambda \mathbf{v}^{(t)} - \alpha \nabla f(\mathbf{x}^{(t)}) \quad \text{momentum}$$

come andare giù da una collina

$$\mathbf{x}^{(t+1)} = \mathbf{x}^{(t)} + \mathbf{v}^{(t+1)}$$

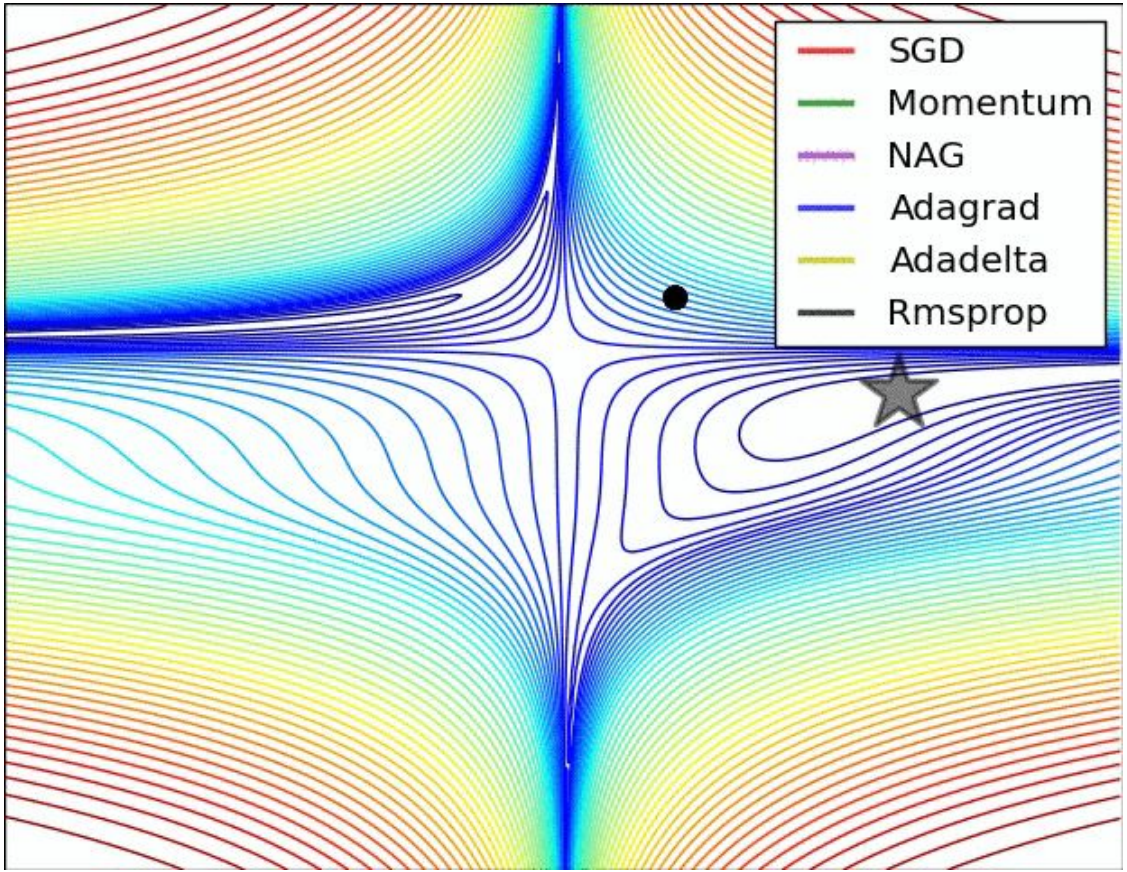
Step length $\propto$ how aligned is the sequence of gradients.

$$\frac{1}{1-\lambda}\alpha\|\nabla f\|$$

Acceleration effect for big $\lambda$ + escape from local minima.

# Momentum

il momentum, siccome gli step cambiano e si accumulano, la palla verde va verso il real minimum




https://cs231n.github.io/neural-networks-3/

# First Order Acceleration Methods

Let us try to unroll gradient descent:

$$\mathbf{x}^{(t+1)} = \mathbf{x}^{(t)} - \alpha \nabla f(\mathbf{x}^{(t)})$$

$$\begin{aligned}
\mathbf{x}^{(1)} &= \mathbf{x}^{(0)} - \alpha \nabla f(\mathbf{x}^{(0)}) \\
\mathbf{x}^{(2)} &= \mathbf{x}^{(1)} - \alpha \nabla f(\mathbf{x}^{(1)}) \\
&= \mathbf{x}^{(0)} - \alpha \nabla f(\mathbf{x}^{(0)}) - \alpha \nabla f(\mathbf{x}^{(1)}) \\
&\vdots \\
\mathbf{x}^{(t+1)} &= \mathbf{x}^{(0)} - \alpha \sum_{i=1}^{t} \nabla f(\mathbf{x}^{(i)})
\end{aligned}$$



Slide Credit: E. Rodolà

# First Order Acceleration Methods

Let us try to unroll gradient descent:

$$\mathbf{x}^{(t+1)} = \mathbf{x}^{(0)} - \alpha \sum_{i=1}^{t} \nabla f(\mathbf{x}^{(i)})$$

With momentum:

$$\mathbf{x}^{(t+1)} = \mathbf{x}^{(0)} + \alpha \sum_{i=1}^{t} \frac{1-\lambda^{t+1-i}}{1-\lambda} \nabla f(\mathbf{x}^{(i)})$$

The more general form:

$$\mathbf{x}^{(t+1)} = \mathbf{x}^{(0)} + \alpha \sum_{i=1}^{t} \gamma_i^t \nabla f(\mathbf{x}^{(i)}) \quad \text{for some } \gamma_i$$

# First Order Acceleration Methods

Let us try to unroll gradient descent:

$$\mathbf{x}^{(t+1)} = \mathbf{x}^{(0)} - \alpha \sum_{i=1}^{t} \nabla f(\mathbf{x}^{(i)})$$

With momentum:

$$\mathbf{x}^{(t+1)} = \mathbf{x}^{(0)} + \alpha \sum_{i=1}^{t} \frac{1-\lambda^{t+1-i}}{1-\lambda} \nabla f(\mathbf{x}^{(i)})$$

The more general form:

$$\mathbf{x}^{(t+1)} = \mathbf{x}^{(0)} + \alpha \sum_{i=1}^{t} \Gamma_i^t \nabla f(\mathbf{x}^{(i)}) \quad \text{for some diag. matrix } \Gamma_i$$

generalizes optimization algorithms like ADAM, AdaGrad, etc.

# Gradient Descent for Deep Learning

Gradient descent can be applied to nonconvex problems, without optimality guarantees.

In order to gain generalization, the following consideration is crucial:

We are rarely interested in the global optimum.

# Gradient Descent for Deep Learning

Gradient descent can be applied to nonconvex problems, without optimality guarantees.

In order to gain generalization, the following consideration is crucial:

We are rarely interested in the global optimum.

Even for convex problems like:

- Linear regression ($\mathbf{X}$ can be huge and must be inverted/factorized)
- Logistic regression (no closed form solution)

We get more efficient and numerically stable solutions.

# Gradient Descent for Deep Learning

In the general DL setting:

Each parameter gets updated so as to decrease the loss:

$$\boldsymbol{\theta}_i \leftarrow \boldsymbol{\theta}_i - \alpha \frac{\partial \ell}{\partial \boldsymbol{\theta}_i}$$

The gradient tells us how to modify the parameters.

- $\boldsymbol{\theta}$ stores the neural network parameters, possibly millions
- The loss can be non-convex and non-differentiable
  Cannot even apply gradient descent!
- Must address computational complexity and numerical stability

Very often in DL we confront problems that have well defined solutions in mathematical terms, but require extra care and tweaking in practice.

# Stochastic Gradient Descent

Recall that the loss is usually defined over $n$ training examples:

$$\ell_\Theta(\{x_i, y_i\}) = \frac{1}{n}\sum_{i=1}^{n}(y_i - f_\Theta(x_i))^2$$

$$\ell_\Theta(\{x_i, y_i\}) = \frac{1}{n}\sum_{i=1}^{n}\hat{\ell}_\Theta(\{x_i, y_i\})$$

which requires computing the gradient for each term in the summation:

$$\nabla\ell_\Theta(\{x_i, y_i\}) = \frac{1}{n}\sum_{i=1}^{n}\nabla\hat{\ell}_\Theta(\{x_i, y_i\})$$

Two bottlenecks make gradient descent impractical:

- Number of examples
- Number of parameters



Slide Credit: E. Rodolà

# Stochastic Gradient Descent

$$\ell_\Theta(\{x_i, y_i\}) = \frac{1}{n}\sum_{i=1}^{n} \hat{\ell}_\Theta(\{x_i, y_i\})$$

$$\nabla\ell_\Theta(\mathcal{T}) = \frac{1}{n}\sum_{i=1}^{n} \nabla\hat{\ell}_\Theta(\mathcal{T})$$

Compute $\nabla\ell_\Theta$ for a small representative subset of $m \ll n$ examples:

il piccolo gruppo di data deve essere rappresentativo dell'intero dataset

$$\frac{1}{m}\sum_{i=1}^{m} \nabla\hat{\ell}_\Theta(\mathcal{B}) \approx \frac{1}{n}\sum_{i=1}^{n} \nabla\hat{\ell}_\Theta(\mathcal{T})$$

The mini-batch $\mathcal{B} \subset \mathcal{T}$ is drawn uniformly.

The true gradient $\nabla\ell_\Theta$ is approximated, but with a significant speed-up.

**Example:** MNIST dataset

$$n = 60,000\,,\ m = 10 \quad \Rightarrow \quad 6,000 \times \text{ speedup}$$

# Stochastic Gradient Descent

$$\ell_\Theta(\{x_i, y_i\}) = \frac{1}{n}\sum_{i=1}^{n} \hat{\ell}_\Theta(\{x_i, y_i\})$$

$$\nabla\ell_\Theta(\mathcal{T}) = \frac{1}{n}\sum_{i=1}^{n} \nabla\hat{\ell}_\Theta(\mathcal{T})$$

Compute $\nabla\ell_\Theta$ for a small representative subset of $m \ll n$ examples:

$$\frac{1}{m}\sum_{i=1}^{m} \nabla\hat{\ell}_\Theta(\mathcal{B}) \approx \frac{1}{n}\sum_{i=1}^{n} \nabla\hat{\ell}_\Theta(\mathcal{T})$$

The mini-batch $\mathcal{B} \subset \mathcal{T}$ is drawn uniformly.

The true gradient $\nabla\ell_\Theta$ is approximated, but with a significant speed-up.

**Example:** MNIST dataset

$$n = 60,000\,,\ m = 10 \quad \Rightarrow \quad 6,000\times \text{ speedup}$$

# Stochastic Gradient Descent

The algorithm is as follows:

1. Initialize $\boldsymbol{\theta}$.
2. Pick a mini-batch $\mathcal{B}$.
3. Update with the downhill step (use momentum if desired):

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \alpha \nabla \ell_{\boldsymbol{\theta}}(\mathcal{B})$$

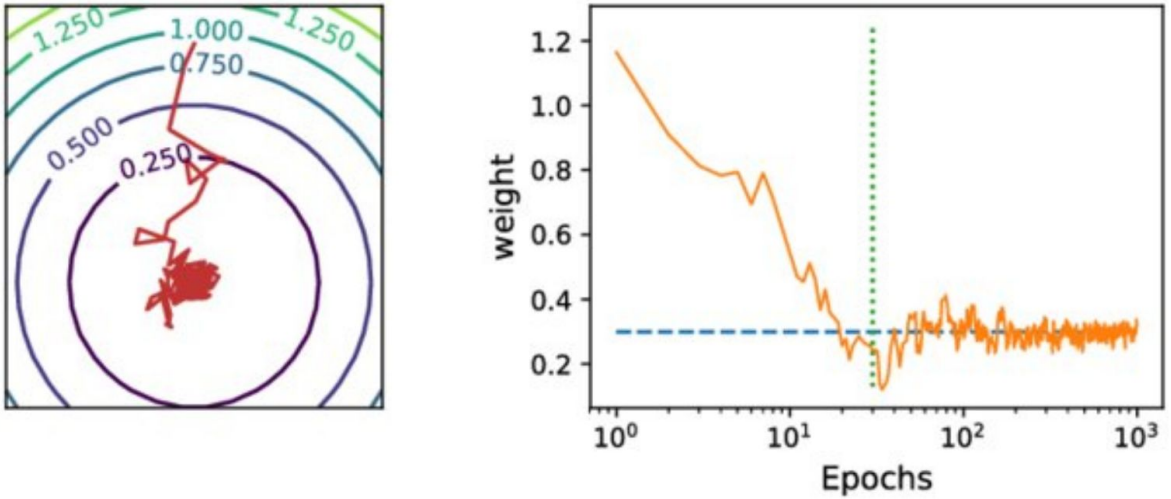
4. Go back to step (2).

Testo

When steps (2)-(4) cover the entire training set $\mathcal{T}$ we have an epoch.
Like gradient descent, the algorithm proceeds for many epochs.

**Remark:** The update cost is constant regardless of the size of $\mathcal{T}$.

# Stochastic Gradient Descent




SGD does not stop at the minimum.

Oscillations are due to the noise induced by the random sampling.

# Asymptotic upper bounds

For a convex problem, consider the inequality:

f* è la scelta giusta di parametri che ci permette di minimizzare il loss

$$|\underbrace{\ell(f_\Theta)}_{\text{GD/SGD}} - \underbrace{\ell(f^*)}_{\text{true}}| < \underbrace{\rho}_{\text{accuracy}}$$

$n$ training examples
$d$ parameters
$\kappa, \nu$ are constants related to the conditioning of the problem

| | cost per iteration | iterations to reach $\rho$ |
|---|---|---|
| GD | $O(nd)$ | $O(\kappa \log \frac{1}{\rho})$ |
| SGD | $O(d)$ | $\frac{\nu\kappa^2}{\rho} + o(\frac{1}{\rho})$ |

questa è la vera differenza

queste curve non sono molto diverse

SGD does not depend on the number of examples, implying better generalization

Bottou and Bousquet, "The Tradeoffs of Large Scale Learning", NIPS 2008

# Stochastic Gradient Descent

Some practical considerations:

- Needs the training data to be shuffled to avoid data bias.
- Each mini-batch can be processed in parallel.
  Batch size is limited by hardware.
- Large datasets: more steps to reach convergence.
  Remarkably rapid initial progress and slow asymptotic convergence.
- Small mini-batches can offer a regularizing effect.
  Very small batches $\rightarrow$ high variance in the estimation of the gradient
  $\rightarrow$ use a small learning rate to maintain stability.

$$\frac{1}{m}\sum_{i=1}^{m}\nabla\hat{\ell}_{\Theta}(\mathcal{B}) \approx \frac{1}{n}\sum_{i=1}^{n}\nabla\hat{\ell}_{\Theta}(\mathcal{T})$$

SGD can find a low value of the loss quickly enough to be useful, even if it's not a minimum.