# Big Data Computing

## Master's Degree in Computer Science

## 2021-2022

Gabriele Tolomei

Department of Computer Science

Sapienza Università di Roma

tolomei@di.uniroma1.it

SAPIENZA
UNIVERSITÀ DI ROMA

# Recap from Last Lecture

- Logistic Regression is a powerful tool for predicting binary variables through probability of each class
- It fits a regression line between input (features) and output (logarithm of the odds), assuming probability takes the form of a sigmoid function
- Parameter estimation is typically done via MLE (i.e., by minimizing Cross-Entropy error)
- We need a more sophisticated learning algorithm!

# LEARNING ALGORITHM

# Picking the Best Hypothesis

- So far, we have defined:
  - The model (logistic function)
  - The error measure (cross-entropy)

# Picking the Best Hypothesis

- So far, we have defined:
  - The model (logistic function)
  - The error measure (cross-entropy)

To actually select the best hypothesis, we have to pick the vector of parameters $\boldsymbol{\theta}^*$ so that the error measure is minimized

$$\boxed{E_{\text{in}}(\boldsymbol{\theta}) = \frac{1}{m} \sum_{i=1}^{m} \ln\left(e^{-y_i \boldsymbol{\theta}^T \mathbf{x_i}} + 1\right)}$$

# Mean Squared Error vs. Cross-Entropy

In the case of linear regression we have a similar expression for the error measure, i.e., Mean Squared Error (MSE)

$$E_{\text{in}}(\boldsymbol{\theta}) = \frac{1}{m} \sum_{i=1}^{m} \left(\boldsymbol{\theta}^T \mathbf{x_i} - y_i\right)^2$$

# Mean Squared Error vs. Cross-Entropy

In the case of linear regression we have a similar expression for the error measure, i.e., Mean Squared Error (MSE)

$$E_{\text{in}}(\boldsymbol{\theta}) = \frac{1}{m} \sum_{i=1}^{m} \left(\boldsymbol{\theta}^T \mathbf{x_i} - y_i\right)^2$$

Minimising MSE through Ordinary Least Squares (OLS) leads to a closed-form solution often referred to as the OLS estimator for $\boldsymbol{\theta}^*$

$$\boxed{\hat{\boldsymbol{\theta}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}}$$

# Mean Squared Error vs. Cross-Entropy

The problem is that using Cross-Entropy as error measure we cannot find a closed-form solution to the minimization problem

$$E_{\text{in}}(\boldsymbol{\theta}) = \frac{1}{m} \sum_{i=1}^{m} \ln\left(e^{-y_i \boldsymbol{\theta}^T \mathbf{x_i}} + 1\right)$$

# Mean Squared Error vs. Cross-Entropy

The problem is that using Cross-Entropy as error measure we cannot find a closed-form solution to the minimization problem

$$E_{\text{in}}(\boldsymbol{\theta}) = \frac{1}{m} \sum_{i=1}^{m} \ln\left(e^{-y_i \boldsymbol{\theta}^T \mathbf{x_i}} + 1\right)$$

Yet, Cross-Entropy is convex w.r.t. the parameters $\boldsymbol{\theta}$

# Mean Squared Error vs. Cross-Entropy

The problem is that using Cross-Entropy as error measure we cannot find a closed-form solution to the minimization problem

$$E_{\text{in}}(\boldsymbol{\theta}) = \frac{1}{m} \sum_{i=1}^{m} \ln\left(e^{-y_i \boldsymbol{\theta}^T \mathbf{x_i}} + 1\right)$$

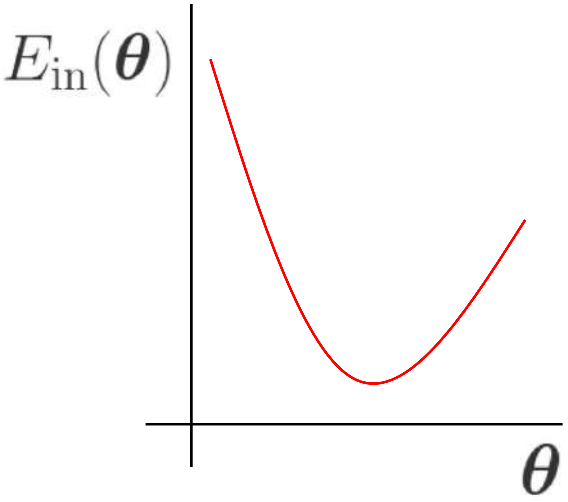
Yet, Cross-Entropy is convex w.r.t. the parameters $\boldsymbol{\theta}$

$\Downarrow$

**Iterative Solution**

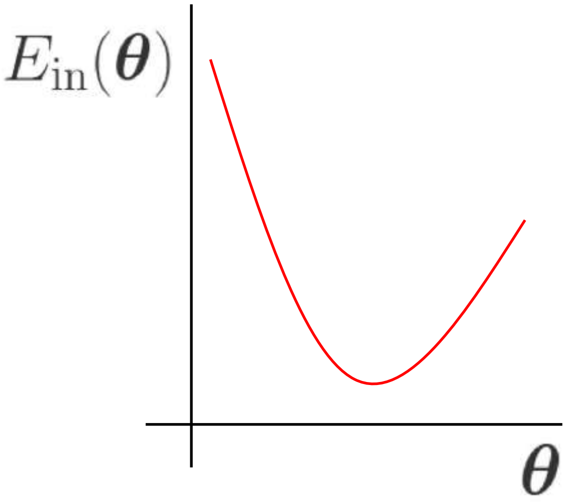
# (Batch) Gradient Descent

General iterative method for any nonlinear optimization



# (Batch) Gradient Descent

General iterative method for any nonlinear optimization

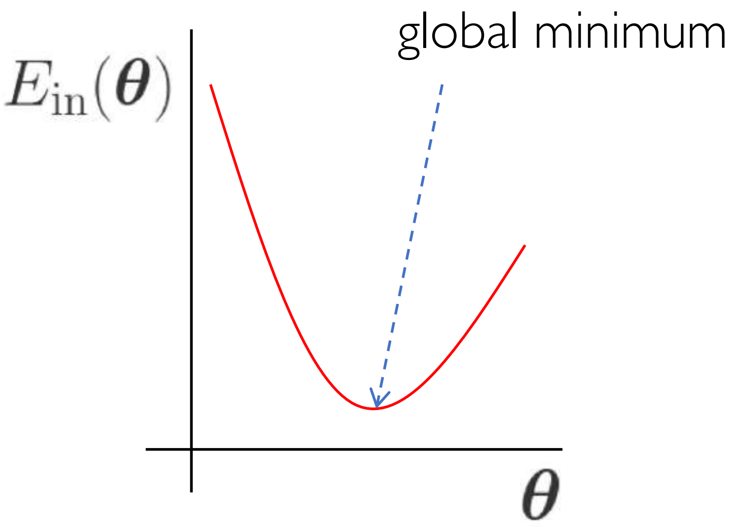



The method **guarantees the convergence to a local minimum**

(Under specific assumptions on the objective function and learning rate)

# (Batch) Gradient Descent

General iterative method for any nonlinear optimization




The method **guarantees the convergence to a local minimum**

(Under specific assumptions on the objective function and learning rate)

If the objective function is **convex** (like cross-entropy) then the local minimum is also the **global minimum**

# Gradient Descent: The Main Idea

1. At $t = 0$ initialize the (guessed) vector of parameters $\boldsymbol{\theta}$ to $\boldsymbol{\theta}(0)$

# Gradient Descent: The Main Idea

1. At $t = 0$ initialize the (guessed) vector of parameters $\boldsymbol{\theta}$ to $\boldsymbol{\theta}(0)$
2. Repeat until convergence:
   a. Update the current vector of parameters $\boldsymbol{\theta}(t)$ by taking a "step" along the "steepest" slope: $\boldsymbol{\theta}(t+1) = \boldsymbol{\theta}(t) + \eta v$
   b. Return to 2.

# Gradient Descent: The Main Idea

1. At $t$ = 0 initialize the (guessed) vector of parameters **θ** to **θ**(0)
2. Repeat until convergence:
   a. Update the current vector of parameters **θ**($t$) by taking a "step" along the "steepest" slope: **θ**($t$+1) = **θ**($t$) + ηv
   b. Return to 2.

Unit vector representing the direction of the steepest slope

$$\boxed{\boldsymbol{\theta}(t+1) = \boldsymbol{\theta}(t) + \eta \mathbf{v}}$$

step

# Gradient Descent: The Main Idea

1. At $t = 0$ initialize the (guessed) vector of parameters $\boldsymbol{\theta}$ to $\boldsymbol{\theta}(0)$
2. Repeat until convergence:
   a. Update the current vector of parameters $\boldsymbol{\theta}(t)$ by taking a "step" along the "steepest" slope: $\boldsymbol{\theta}(t+1) = \boldsymbol{\theta}(t) + \eta v$
   b. Return to 2.

Unit vector representing the direction of the steepest slope

$$\boxed{\boldsymbol{\theta}(t+1) = \boldsymbol{\theta}(t) + \eta \mathbf{v}}$$

step

How do we determine the direction v?

# Gradient Descent: The Direction v

- We already intuitively said that the direction **v** should be that of the "steepest" slope

# Gradient Descent: The Direction v

- We already intuitively said that the direction **v** should be that of the "steepest" slope
- Concretely, this means moving along the direction which mostly reduces the in-sample error function

$$\boxed{\Delta E_{\text{in}}(\boldsymbol{\theta}, t) = E_{\text{in}}(\boldsymbol{\theta}(t)) - E_{\text{in}}(\boldsymbol{\theta}(t-1))}$$

# Gradient Descent: The Direction v

- We already intuitively said that the direction **v** should be that of the "steepest" slope
- Concretely, this means moving along the direction which mostly reduces the in-sample error function

$$\boxed{\Delta E_{\text{in}}(\boldsymbol{\theta}, t) = E_{\text{in}}(\boldsymbol{\theta}(t)) - E_{\text{in}}(\boldsymbol{\theta}(t-1))}$$

We want $\Delta E_{in}$ to be as negative as possible, which means that we are actually reducing the error w.r.t. the previous iteration $t$-1

# Gradient Descent: The Direction v

$$\Delta E_{\text{in}}(\boldsymbol{\theta}, t) = E_{\text{in}}(\boldsymbol{\theta}(t-1) + \eta \mathbf{v}) - E_{\text{in}}(\boldsymbol{\theta}(t-1))$$

# Gradient Descent: The Direction v

$$\Delta E_{\text{in}}(\boldsymbol{\theta}, t) = E_{\text{in}}(\boldsymbol{\theta}(t-1) + \eta \mathbf{v}) - E_{\text{in}}(\boldsymbol{\theta}(t-1))$$

Let's first assume we are in the univariate case, i.e., $\boldsymbol{\theta} = \vartheta$ in R

$\boxed{f = E_{\text{in}}}$ $\boxed{x_0 = \boldsymbol{\theta}(t-1)}$ $\boxed{x = \boldsymbol{\theta}(t)}$

# Gradient Descent: The Direction v

$$\Delta E_{\text{in}}(\boldsymbol{\theta}, t) = E_{\text{in}}(\boldsymbol{\theta}(t-1) + \eta\mathbf{v}) - E_{\text{in}}(\boldsymbol{\theta}(t-1))$$

Let's first assume we are in the univariate case, i.e., $\boldsymbol{\theta} = \vartheta$ in R

$$\boxed{f = E_{\text{in}}} \qquad \boxed{x_0 = \boldsymbol{\theta}(t-1)} \qquad \boxed{x = \boldsymbol{\theta}(t)}$$

$$\boxed{\delta f = \Delta E_{\text{in}} = f(x) - f(x_0)} \qquad \boxed{\delta x = x - x_0 = \boldsymbol{\theta}(t) - \boldsymbol{\theta}(t-1) = \eta\mathbf{v}}$$

# Gradient Descent: The Direction v

$$\Delta E_{\text{in}}(\boldsymbol{\theta}, t) = E_{\text{in}}(\boldsymbol{\theta}(t-1) + \eta \mathbf{v}) - E_{\text{in}}(\boldsymbol{\theta}(t-1))$$

Let's first assume we are in the univariate case, i.e., $\boldsymbol{\theta} = \vartheta$ in R

$$\boxed{f = E_{\text{in}}} \qquad \boxed{x_0 = \boldsymbol{\theta}(t-1)} \qquad \boxed{x = \boldsymbol{\theta}(t)}$$

$$\boxed{\delta f = \Delta E_{\text{in}} = f(x) - f(x_0)} \qquad \boxed{\delta x = x - x_0 = \boldsymbol{\theta}(t) - \boldsymbol{\theta}(t-1) = \eta \mathbf{v}}$$

$$\boxed{f'(x_0) = \lim_{\delta x \to 0} \frac{f(x_0 + \delta x) - f(x_0)}{\delta x}} \qquad \boxed{f'(x_0) = \lim_{x \to x_0} \frac{f(x) - f(x_0)}{x - x_0} \approx \frac{\delta f}{\delta x}}$$

# Gradient Descent: The Direction v

$$\Delta E_{\text{in}}(\boldsymbol{\theta}, t) = E_{\text{in}}(\boldsymbol{\theta}(t-1) + \eta \mathbf{v}) - E_{\text{in}}(\boldsymbol{\theta}(t-1))$$

Let's first assume we are in the univariate case, i.e., $\boldsymbol{\theta} = \vartheta$ in R

$$f = E_{\text{in}}$$

$$x_0 = \boldsymbol{\theta}(t-1)$$

$$x = \boldsymbol{\theta}(t)$$

$$\delta f = \Delta E_{\text{in}} = f(x) - f(x_0)$$

$$\delta x = x - x_0 = \boldsymbol{\theta}(t) - \boldsymbol{\theta}(t-1) = \eta \mathbf{v}$$

$$f'(x_0) = \lim_{\delta x \to 0} \frac{f(x_0 + \delta x) - f(x_0)}{\delta x}$$

$$f'(x_0) = \lim_{x \to x_0} \frac{f(x) - f(x_0)}{x - x_0} \approx \frac{\delta f}{\delta x}$$

$$\delta f = f(x) - f(x_0) \approx f'(x_0)\delta x = f'(x_0)(x - x_0)$$

# Gradient Descent: The Direction v

$$\boxed{f(x) - f(x_0) \approx f'(x_0)(x - x_0)}$$

# Gradient Descent: The Direction v

$$\boxed{f(x) - f(x_0) \approx f'(x_0)(x - x_0)}$$

$$f(x) = \underbrace{f(x_0) + f'(x_0)(x - x_0)}_{\text{First-order Taylor approximation}} \underbrace{+O((x - x_0)^2)}_{\text{Second-order error term}}$$

# Gradient Descent: The Direction v

$$\boxed{f(x) - f(x_0) \approx f'(x_0)(x - x_0)}$$

$$f(x) = \underbrace{f(x_0) + f'(x_0)(x - x_0)}_{\text{First-order Taylor approximation}} \underbrace{+O((x - x_0)^2)}_{\text{Second-order error term}}$$

To summarize and generalize to the multivariate case of **θ**:

$$\boxed{\delta f = f(x) - f(x_0) = \Delta E_{\text{in}} = \eta \nabla E_{\text{in}}(\boldsymbol{\theta}(t-1))^T \mathbf{v} + O(\eta^2)}$$

The greek letter *nabla* indicates the gradient

# Gradient Descent: The Direction v

$$\Delta E_{\text{in}} = \eta \nabla E_{\text{in}}(\boldsymbol{\theta}(t-1))^T \mathbf{v} + O(\eta^2)$$

# Gradient Descent: The Direction v

$$\Delta E_{\text{in}} = \eta \nabla E_{\text{in}}(\boldsymbol{\theta}(t-1))^T \mathbf{v} + O(\eta^2)$$

The unit vector v only contributes to the direction and not to the magnitude of the iterative step

# Gradient Descent: The Direction v

$$\Delta E_{\text{in}} = \eta \nabla E_{\text{in}}(\boldsymbol{\theta}(t-1))^T \mathbf{v} + \cancel{O(\eta^2)}$$

The unit vector v only contributes to the direction and not to the magnitude of the iterative step

The second-order approximation term is negligible

(when the step size is small)

# Gradient Descent: The Direction v

$$\nabla E_{\text{in}}(\boldsymbol{\theta}(t-1))^T = \mathbf{u}$$

$$\Delta E_{\text{in}} = \eta \mathbf{u} \cdot \mathbf{v}$$

# Gradient Descent: The Direction v

$$\nabla E_{\text{in}}(\boldsymbol{\theta}(t-1))^T = \mathbf{u}$$

$$\Delta E_{\text{in}} = \eta \mathbf{u} \cdot \mathbf{v}$$

$$\mathbf{u} \cdot \mathbf{v} = ||\mathbf{u}|| \underbrace{||\mathbf{v}||}_{=1} cos(\alpha) = ||\mathbf{u}|| cos(\alpha)$$

# Gradient Descent: The Direction v

$$\nabla E_{\text{in}}(\boldsymbol{\theta}(t-1))^T = \mathbf{u}$$

$$\Delta E_{\text{in}} = \eta \mathbf{u} \cdot \mathbf{v}$$

$$\mathbf{u} \cdot \mathbf{v} = ||\mathbf{u}|| \underbrace{||\mathbf{v}||}_{=1} cos(\alpha) = ||\mathbf{u}|| cos(\alpha) \qquad -1 \leq cos(\alpha) \leq 1$$

# Gradient Descent: The Direction v

$$\nabla E_{\text{in}}(\boldsymbol{\theta}(t-1))^T = \mathbf{u}$$

$$\Delta E_{\text{in}} = \eta \mathbf{u} \cdot \mathbf{v}$$

$$\mathbf{u} \cdot \mathbf{v} = ||\mathbf{u}|| \underbrace{||\mathbf{v}||}_{=1} cos(\alpha) = ||\mathbf{u}|| cos(\alpha) \qquad -1 \leq cos(\alpha) \leq 1$$

$$-||\mathbf{u}|| \leq \mathbf{u} \cdot \mathbf{v} \leq ||\mathbf{u}||$$

$$-\eta ||\mathbf{u}|| \leq \underbrace{\eta \mathbf{u} \cdot \mathbf{v}}_{\Delta E_{\text{in}}} \leq \eta ||\mathbf{u}||$$

# Gradient Descent: The Direction v

$$\nabla E_{\text{in}}(\boldsymbol{\theta}(t-1))^T = \mathbf{u}$$

$$\Delta E_{\text{in}} = \eta \mathbf{u} \cdot \mathbf{v}$$

$$\mathbf{u} \cdot \mathbf{v} = ||\mathbf{u}|| \underbrace{||\mathbf{v}||}_{=1} cos(\alpha) = ||\mathbf{u}|| cos(\alpha) \qquad -1 \leq cos(\alpha) \leq 1$$

$$-||\mathbf{u}|| \leq \mathbf{u} \cdot \mathbf{v} \leq ||\mathbf{u}||$$

$$-\eta||\mathbf{u}|| \leq \underbrace{\eta \mathbf{u} \cdot \mathbf{v}}_{\Delta E_{\text{in}}} \leq \boxed{\eta||\mathbf{u}||}$$

The most positive $\Delta E_{in}$ when $\cos(\alpha) = 1$ (i.e., $\alpha = 0°$)

Both error and step vectors have the same direction

# Gradient Descent: The Direction v

$$\nabla E_{\text{in}}(\boldsymbol{\theta}(t-1))^T = \mathbf{u}$$

$$\Delta E_{\text{in}} = \eta \mathbf{u} \cdot \mathbf{v}$$

$$\mathbf{u} \cdot \mathbf{v} = ||\mathbf{u}|| \underbrace{||\mathbf{v}||}_{=1} cos(\alpha) = ||\mathbf{u}|| cos(\alpha) \qquad -1 \leq cos(\alpha) \leq 1$$

$$-||\mathbf{u}|| \leq \mathbf{u} \cdot \mathbf{v} \leq ||\mathbf{u}||$$

$$\boxed{-\eta||\mathbf{u}||} \leq \underbrace{\eta \mathbf{u} \cdot \mathbf{v}}_{\Delta E_{\text{in}}} \leq \eta ||\mathbf{u}||$$

The most **negative** $\Delta E_{in}$ when $\cos(\alpha) = -1$ (i.e., $\alpha = 180°$)

The error and step vectors have opposite direction

# Gradient Descent: The Direction **v**

At each iteration $t$, we want the unit vector **v** which makes exactly **the most negative** $\Delta E_{in}$

$$\eta \mathbf{u} \cdot \mathbf{v} = -\eta ||\mathbf{u}||$$

# Gradient Descent: The Direction v

At each iteration $t$, we want the unit vector **v** which makes exactly the most negative $\Delta E_{in}$

$$\eta \mathbf{u} \cdot \mathbf{v} = -\eta ||\mathbf{u}||$$

$$\mathbf{u} \cdot \mathbf{v} = -||\mathbf{u}||$$
$$\mathbf{u}^T \cdot \mathbf{u} \cdot \mathbf{v} = -||\mathbf{u}||\mathbf{u}^T$$

$$\mathbf{v} = -\frac{||\mathbf{u}||\mathbf{u}^T}{||\mathbf{u}||^2} = -\frac{\mathbf{u}^T}{||\mathbf{u}||} = -\frac{\nabla E_{\text{in}}(\boldsymbol{\theta}(t-1))}{||\nabla E_{\text{in}}(\boldsymbol{\theta}(t-1))||}$$

# Gradient Descent: The Direction v

At each iteration $t$, we want the unit vector **v** which makes exactly the most negative $\Delta E_{in}$

$$\eta \mathbf{u} \cdot \mathbf{v} = -\eta ||\mathbf{u}||$$

$$\mathbf{u} \cdot \mathbf{v} = -||\mathbf{u}||$$

$$\mathbf{u}^T \cdot \mathbf{u} \cdot \mathbf{v} = -||\mathbf{u}||\mathbf{u}^T$$

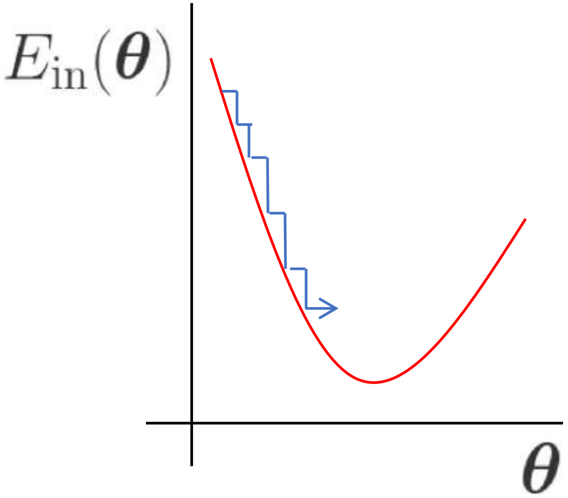
$$\mathbf{v} = -\frac{||\mathbf{u}||\mathbf{u}^T}{||\mathbf{u}||^2} = -\frac{\mathbf{u}^T}{||\mathbf{u}||} = \boxed{-\frac{\nabla E_{\text{in}}(\boldsymbol{\theta}(t-1))}{||\nabla E_{\text{in}}(\boldsymbol{\theta}(t-1))||}}$$

# Gradient Descent: The Step η

How the step magnitude η affects the convergence?

# Gradient Descent: The Step η

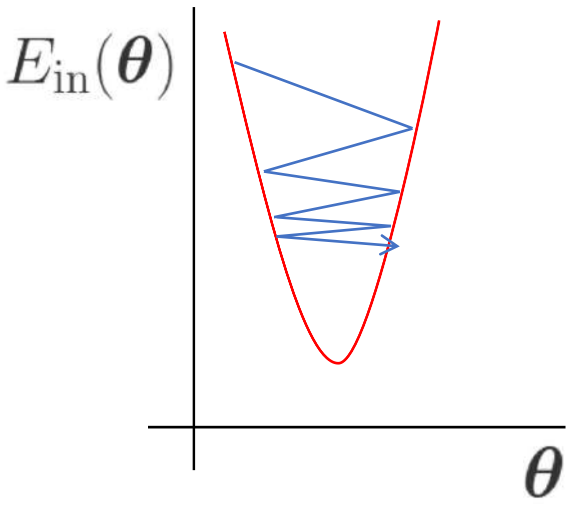
How the step magnitude η affects the convergence?




η too small

# Gradient Descent: The Step η

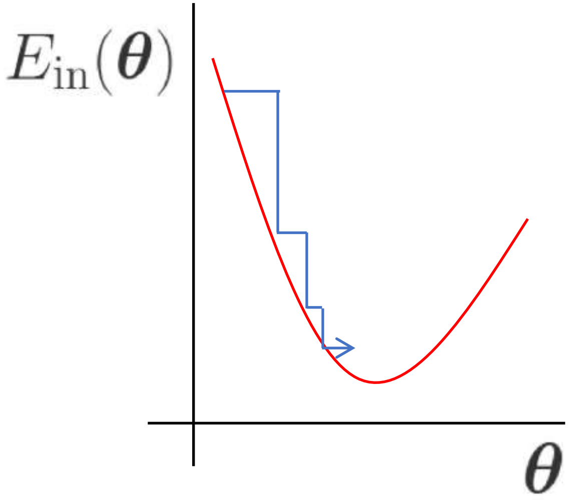
How the step magnitude η affects the convergence?




η too large

# Gradient Descent: The Step η

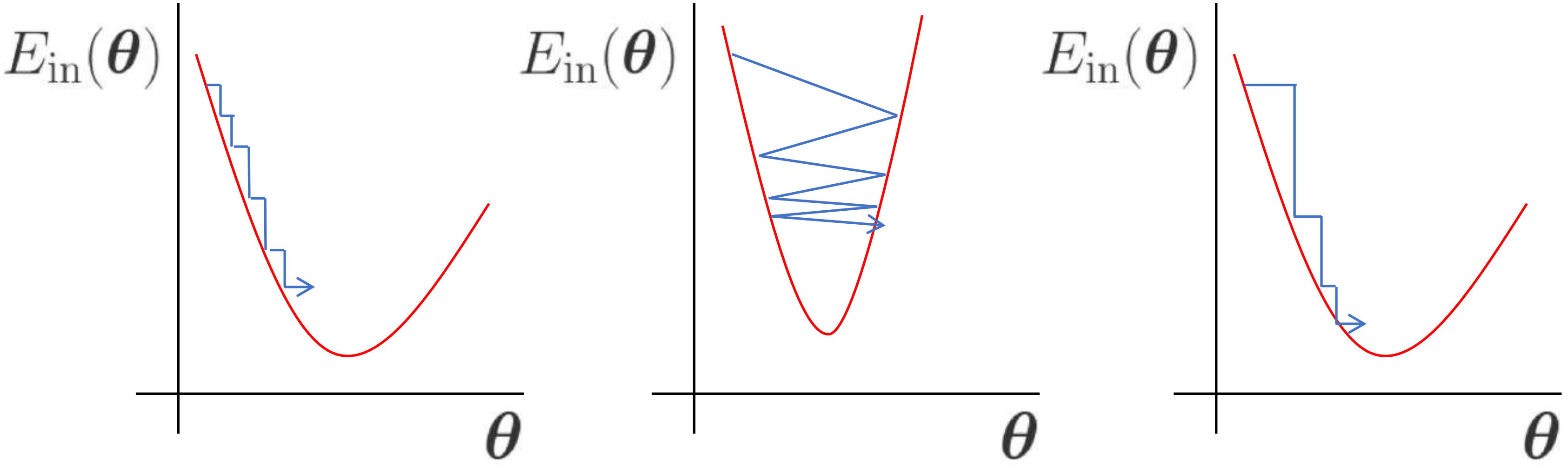
How the step magnitude η affects the convergence?




η variable

# Gradient Descent: The Step η

How the step magnitude η affects the convergence?




η too small | η too large | η variable

**Rule of thumb**

Dynamically change η proportionally to the gradient!

# Gradient Descent: The Step η

Remember that at each iteration the update strategy is:

$$\boldsymbol{\theta}(t+1) = \boldsymbol{\theta}(t) + \eta \mathbf{v}$$

$$\mathbf{v} = -\frac{\nabla E_{\text{in}}(\boldsymbol{\theta}(t))}{\|\nabla E_{\text{in}}(\boldsymbol{\theta}(t))\|}$$

# Gradient Descent: The Step η

Remember that at each iteration the update strategy is:

$$\boldsymbol{\theta}(t+1) = \boldsymbol{\theta}(t) + \eta \mathbf{v}$$

$$\mathbf{v} = -\frac{\nabla E_{\text{in}}(\boldsymbol{\theta}(t))}{\|\nabla E_{\text{in}}(\boldsymbol{\theta}(t))\|}$$

At each iteration *t*, the step η is fixed

$$\boldsymbol{\theta}(t+1) = \boldsymbol{\theta}(t) - \eta \frac{\nabla E_{\text{in}}(\boldsymbol{\theta}(t))}{\|\nabla E_{\text{in}}(\boldsymbol{\theta}(t))\|}$$

# Gradient Descent: The Step η

Instead of having a fixed η at each iteration, use a variable $\eta_t$ as function of η

$$\boldsymbol{\theta}(t+1) = \boldsymbol{\theta}(t) + \eta_t \mathbf{v} \qquad \eta_t = \eta k$$

# Gradient Descent: The Step η

Instead of having a fixed η at each iteration, use a variable $\eta_t$ as function of η

$$\boldsymbol{\theta}(t+1) = \boldsymbol{\theta}(t) + \eta_t \mathbf{v} \qquad \eta_t = \eta k$$

Let's take: $$\boldsymbol{\theta}(t+1) = \boldsymbol{\theta}(t) - \eta k \frac{\nabla E_{\text{in}}(\boldsymbol{\theta}(t))}{\|\nabla E_{\text{in}}(\boldsymbol{\theta}(t))\|}$$

# Gradient Descent: The Step η

Instead of having a fixed η at each iteration, use a variable $\eta_t$ as function of η

$$\boldsymbol{\theta}(t+1) = \boldsymbol{\theta}(t) + \eta_t \mathbf{v} \qquad \eta_t = \eta k$$

Let's take: $$\boldsymbol{\theta}(t+1) = \boldsymbol{\theta}(t) - \eta k \frac{\nabla E_{\text{in}}(\boldsymbol{\theta}(t))}{\|\nabla E_{\text{in}}(\boldsymbol{\theta}(t))\|}$$

$$\boldsymbol{\theta}(t+1) = \boldsymbol{\theta}(t) - \eta \boxed{\|\nabla E_{\text{in}}(\boldsymbol{\theta}(t))\|} \frac{\nabla E_{\text{in}}(\boldsymbol{\theta}(t))}{\|\nabla E_{\text{in}}(\boldsymbol{\theta}(t))\|}$$

# Gradient Descent: The Step η

Instead of having a fixed η at each iteration, use a variable $\eta_t$ as function of η

$$\boldsymbol{\theta}(t+1) = \boldsymbol{\theta}(t) + \eta_t \mathbf{v} \qquad \eta_t = \eta k$$

Let's take: $$\boldsymbol{\theta}(t+1) = \boldsymbol{\theta}(t) - \eta k \frac{\nabla E_{\text{in}}(\boldsymbol{\theta}(t))}{\|\nabla E_{\text{in}}(\boldsymbol{\theta}(t))\|}$$

$$\boldsymbol{\theta}(t+1) = \boldsymbol{\theta}(t) - \eta \cancel{\|\nabla E_{\text{in}}(\boldsymbol{\theta}(t))\|} \frac{\nabla E_{\text{in}}(\boldsymbol{\theta}(t))}{\cancel{\|\nabla E_{\text{in}}(\boldsymbol{\theta}(t))\|}}$$

$$\boxed{\boldsymbol{\theta}(t+1) = \boldsymbol{\theta}(t) - \eta \nabla E_{\text{in}}(\boldsymbol{\theta}(t))}$$

# Computing the Gradient of Cross-Entropy

$$\nabla E_{\text{in}}(\boldsymbol{\theta}) = \nabla \left[ \frac{1}{m} \sum_{i=1}^{m} \ln(e^{-y_i \boldsymbol{\theta}^T \mathbf{x}_i} + 1) \right]$$

# Computing the Gradient of Cross-Entropy

$$\nabla E_{\text{in}}(\boldsymbol{\theta}) = \nabla \left[ \frac{1}{m} \sum_{i=1}^{m} \ln(e^{-y_i \boldsymbol{\theta}^T \mathbf{x}_i} + 1) \right]$$

$$= \left[ \frac{1}{m} \sum_{i=1}^{m} \nabla \ln(e^{-y_i \boldsymbol{\theta}^T \mathbf{x}_i} + 1) \right]$$

# Computing the Gradient of Cross-Entropy

$$\nabla E_{\text{in}}(\boldsymbol{\theta}) = \nabla\left[\frac{1}{m}\sum_{i=1}^{m}\ln(e^{-y_i\boldsymbol{\theta}^T\mathbf{x}_i}+1)\right]$$

$$= \left[\frac{1}{m}\sum_{i=1}^{m}\nabla\ln(e^{-y_i\boldsymbol{\theta}^T\mathbf{x}_i}+1)\right] = \left[\frac{1}{m}\sum_{i=1}^{m}\frac{1}{e^{-y_i\boldsymbol{\theta}^T\mathbf{x}_i}+1}\nabla(e^{-y_i\boldsymbol{\theta}^T\mathbf{x}_i}+1)\right]$$

chain rule of derivative

# Computing the Gradient of Cross-Entropy

$$\nabla E_{\text{in}}(\boldsymbol{\theta}) = \nabla\left[\frac{1}{m}\sum_{i=1}^{m}\ln(e^{-y_i\boldsymbol{\theta}^T\mathbf{x}_i}+1)\right]$$

$$= \left[\frac{1}{m}\sum_{i=1}^{m}\nabla\ln(e^{-y_i\boldsymbol{\theta}^T\mathbf{x}_i}+1)\right] = \left[\frac{1}{m}\sum_{i=1}^{m}\frac{1}{e^{-y_i\boldsymbol{\theta}^T\mathbf{x}_i}+1}\nabla(e^{-y_i\boldsymbol{\theta}^T\mathbf{x}_i}+1)\right]$$

chain rule of derivative

$$= \frac{1}{m}\sum_{i=1}^{m}\frac{-y_i\mathbf{x}_i e^{-y_i\boldsymbol{\theta}^T\mathbf{x}_i}}{e^{-y_i\boldsymbol{\theta}^T\mathbf{x}_i}+1}$$

# Computing the Gradient of Cross-Entropy

$$\nabla E_{\text{in}}(\boldsymbol{\theta}) = \nabla\left[\frac{1}{m}\sum_{i=1}^{m}\ln(e^{-y_i\boldsymbol{\theta}^T\mathbf{x}_i}+1)\right]$$

$$= \left[\frac{1}{m}\sum_{i=1}^{m}\nabla\ln(e^{-y_i\boldsymbol{\theta}^T\mathbf{x}_i}+1)\right] = \left[\frac{1}{m}\sum_{i=1}^{m}\frac{1}{e^{-y_i\boldsymbol{\theta}^T\mathbf{x}_i}+1}\nabla(e^{-y_i\boldsymbol{\theta}^T\mathbf{x}_i}+1)\right]$$

chain rule of derivative

$$= \frac{1}{m}\sum_{i=1}^{m}\frac{-y_i\mathbf{x}_i e^{-y_i\boldsymbol{\theta}^T\mathbf{x}_i}}{e^{-y_i\boldsymbol{\theta}^T\mathbf{x}_i}+1} = \boxed{-\frac{1}{m}\sum_{i=1}^{m}\frac{y_i\mathbf{x}_i}{1+e^{y_i\boldsymbol{\theta}^T\mathbf{x}_i}}}$$

# Gradient Descent: The Algorithm

1. At $t = 0$ initialize the (guessed) vector of parameters $\boldsymbol{\theta}$ to $\boldsymbol{\theta}(0)$

# Gradient Descent: The Algorithm

1. At $t = 0$ initialize the (guessed) vector of parameters $\boldsymbol{\theta}$ to $\boldsymbol{\theta}(0)$
2. For $t = 0, 1, 2, \ldots$ until stop:
   a. Compute the gradient of the cross-entropy error $\boxed{E_{\text{in}}(\boldsymbol{\theta}) = \frac{1}{m} \sum_{i=1}^{m} \ln\left(e^{-y_i \boldsymbol{\theta}^T \mathbf{x_i}} + 1\right)}$

$$\boxed{\nabla E_{\text{in}}(\boldsymbol{\theta}(t)) = -\frac{1}{m} \sum_{i=1}^{m} \frac{y_i \mathbf{x}_i}{1 + e^{y_i \boldsymbol{\theta}(t)^T \mathbf{x}_i}}}$$

# Gradient Descent: The Algorithm

1. At $t = 0$ initialize the (guessed) vector of parameters $\boldsymbol{\theta}$ to $\boldsymbol{\theta}(0)$
2. For $t = 0, 1, 2, \ldots$ until stop:
   a. Compute the gradient of the cross-entropy error $\boxed{E_{\text{in}}(\boldsymbol{\theta}) = \frac{1}{m} \sum_{i=1}^{m} \ln\left(e^{-y_i \boldsymbol{\theta}^T \mathbf{x_i}} + 1\right)}$

$$\boxed{\nabla E_{\text{in}}(\boldsymbol{\theta}(t)) = -\frac{1}{m} \sum_{i=1}^{m} \frac{y_i \mathbf{x}_i}{1 + e^{y_i \boldsymbol{\theta}(t)^T \mathbf{x}_i}}}$$

   b. Update the vector of parameters: $\boldsymbol{\theta}(t+1) = \boldsymbol{\theta}(t) - \eta \nabla E_{\text{in}}(\boldsymbol{\theta}(t))$
   c. Return to 2.

# Gradient Descent: The Algorithm

1. At $t = 0$ initialize the (guessed) vector of parameters $\boldsymbol{\theta}$ to $\boldsymbol{\theta}(0)$
2. For $t = 0, 1, 2, \ldots$ until stop:
   a. Compute the gradient of the cross-entropy error $\boxed{E_{\text{in}}(\boldsymbol{\theta}) = \frac{1}{m} \sum_{i=1}^{m} \ln\left(e^{-y_i \boldsymbol{\theta}^T \mathbf{x_i}} + 1\right)}$

$$\boxed{\nabla E_{\text{in}}(\boldsymbol{\theta}(t)) = -\frac{1}{m} \sum_{i=1}^{m} \frac{y_i \mathbf{x}_i}{1 + e^{y_i \boldsymbol{\theta}(t)^T \mathbf{x}_i}}}$$

   b. Update the vector of parameters: $\boldsymbol{\theta}(t+1) = \boldsymbol{\theta}(t) - \eta \nabla E_{\text{in}}(\boldsymbol{\theta}(t))$
   c. Return to 2.
3. Return the final vector of parameters $\boldsymbol{\theta}(\infty)$

# Gradient Descent: Initialization

- How do we choose the initial value of the parameters $\boldsymbol{\theta}(0)$?

# Gradient Descent: Initialization

- How do we choose the initial value of the parameters $\boldsymbol{\theta}(0)$?
- Typically, random initialization!

# Gradient Descent: Initialization

- How do we choose the initial value of the parameters $\boldsymbol{\theta}(0)$?
- Typically, random initialization!
- If the function is convex we are guaranteed to reach the global minimum no matter what is the initial value of $\boldsymbol{\theta}(0)$

# Gradient Descent: Initialization

- How do we choose the initial value of the parameters $\boldsymbol{\theta}(0)$?
- Typically, random initialization!
- If the function is convex we are guaranteed to reach the global minimum no matter what is the initial value of $\boldsymbol{\theta}(0)$
- In general, we may get to the local minimum nearest to $\boldsymbol{\theta}(0)$

# Gradient Descent: Non-Convex Objectives

- GD can still be used to try to optimize non-convex objectives

# Gradient Descent: Non-Convex Objectives

- GD can still be used to try to optimize non-convex objectives
- Problem: non-convex functions may have several local minima

# Gradient Descent: Non-Convex Objectives

- GD can still be used to try to optimize non-convex objectives
- Problem: non-convex functions may have several local minima
- A bad initialization might cause GD to end up into a "bad" local minimum and miss "better" ones (or even the global if it exists)

# Gradient Descent: Non-Convex Objectives

- GD can still be used to try to optimize non-convex objectives
- Problem: non-convex functions may have several local minima
- A bad initialization might cause GD to end up into a "bad" local minimum and miss "better" ones (or even the global if it exists)
- Solution (heuristic): repeating GD 100÷1,000 times each time with a different $\boldsymbol{\theta}(0)$ may reduce the chance the above issue occurs

# Gradient Descent: Stopping Criterion

- If the function is convex GD reaches the global minimum when

$\nabla E_{\text{in}}(\boldsymbol{\theta}(t)) = 0$

# Gradient Descent: Stopping Criterion

- If the function is convex GD reaches the global minimum when $\nabla E_{in}(\boldsymbol{\theta}(t)) = 0$
- In general, we don't know if eventually the gradient gets to 0 therefore we can use several criteria of termination:
  - stop whenever the difference between two iterations is "small enough" → may converge "prematurely"
  - stop when the error equals to ε → may not converge if the target error is not achievable
  - stop after T iterations
  - combinations of the above in practice works…

# Gradient Descent: Advanced Topics

- Gradient Descent using second-order approximation
  - Better local approximation than first-order but each step requires computing the second derivative (Hessian matrix)

# Gradient Descent: Advanced Topics

- Gradient Descent using second-order approximation
  - Better local approximation than first-order but each step requires computing the second derivative (Hessian matrix)
- Stochastic vs. Mini-Batch Gradient Descent (SGD vs. MBGD)
  - At each iteration, compute the gradient only from one instance (SGD) or a sample of $k$ instances (MBGD) rather than the full dataset

# Gradient Descent: Advanced Topics

- Gradient Descent using second-order approximation
  - Better local approximation than first-order but each step requires computing the second derivative (Hessian matrix)
- Stochastic vs. Mini-Batch Gradient Descent (SGD vs. MBGD)
  - At each iteration, compute the gradient only from one instance (SGD) or a sample of *k* instances (MBGD) rather than the full dataset
- Regularization
  - Include the L1- or L2-norm of the vector of parameters $\boldsymbol{\theta}$ in the cross-entropy error to avoid overfitting

# Take-Home Message of Today

- Gradient Descent (GD) is the standard method for solving optimization objectives (i.e., finding minimum/maximum of a function)
- It requires the function to be differentiable
- If the function is convex, it guarantees to converge to the global minimum
- If the function is quasi-convex, it must avoid getting stuck at a saddle point
- Many variants are currently used: Momentum, RMSProp, Adam, etc. (https://ruder.io/optimizing-gradient-descent/)