Logistic Regression Demystified (Hopefully)

Gabriele Tolomei

Yahoo Labs, London, UK

19th November 2015

Introduction

- 3 components need to be defined:
  - **Model:** describes the set of hypotheses (*hypothesis space*) that can be represented;
  - **Error Measure** (**Cost Function**): measures the price that must be paid if a misclassification error occurs
  - **Learning Algorithm**: is responsible of picking the *best* hypothesis (according to the error measure) by searching through the hypothesis space

The Model

Linear Signal

- Logistic Regression is an example of linear model
- Given a *d+1*-dimensional input **x**

$$\mathbf{x}^T = (x_0, x_1, \ldots, x_d),\ x_0 = 1$$

- We define the family of real-valued functions *F* having *d+1* parameters **θ**

$$\boldsymbol{\theta}^T = (\theta_0, \theta_1, \ldots, \theta_d)$$

- Each function $f_{\boldsymbol{\theta}}$ in *F* outputs a real scalar obtained as a linear combination of the input **x** with the parameters **θ**

$$\mathcal{F} = \{ f_{\boldsymbol{\theta}} : \mathbb{R}^{d+1} \longmapsto \mathbb{R} \mid f_{\boldsymbol{\theta}}(\mathbf{x}) = \boldsymbol{\theta}^T \mathbf{x} = \textstyle\sum_{i=0}^{d} \theta_i x_i \}$$

$f_{\boldsymbol{\theta}}(\mathbf{x})$ means "the application of *f* parametrized by **θ** to **x**" and it is referred to as *signal*

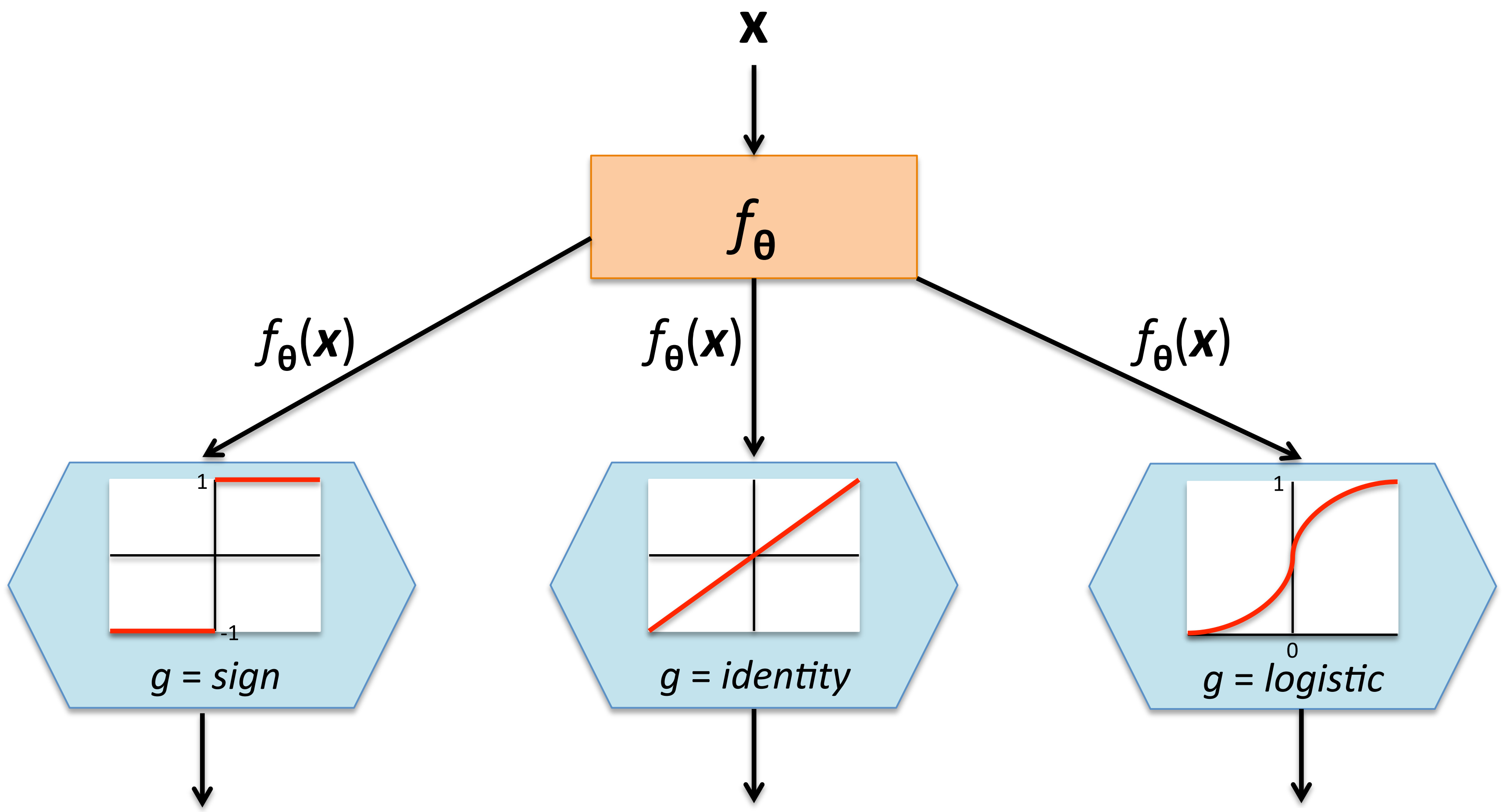
Hypothesis Space

- The signal alone is not enough to define the hypothesis space *H*
- Usually the signal is passed through a "filter", i.e. another real-valued function *g*
- $h_{\boldsymbol{\theta}}(\mathbf{x}) = g(f_{\boldsymbol{\theta}}(\mathbf{x}))$ defines the hypothesis space:

$$\mathcal{H} = \{h_{\boldsymbol{\theta}} : \mathbb{R}^{d+1} \longmapsto \mathbb{R} \mid h_{\boldsymbol{\theta}}(\mathbf{x}) = g(f_{\boldsymbol{\theta}}(\mathbf{x})) = g(\boldsymbol{\theta}^T\mathbf{x}) = g\big(\sum_{i=0}^{d} \theta_i x_i\big)\}$$

The set of possible hypotheses *H* changes depending on the parametric model ($f_{\boldsymbol{\theta}}$) and on the thresholding function (*g*)

Thresholding

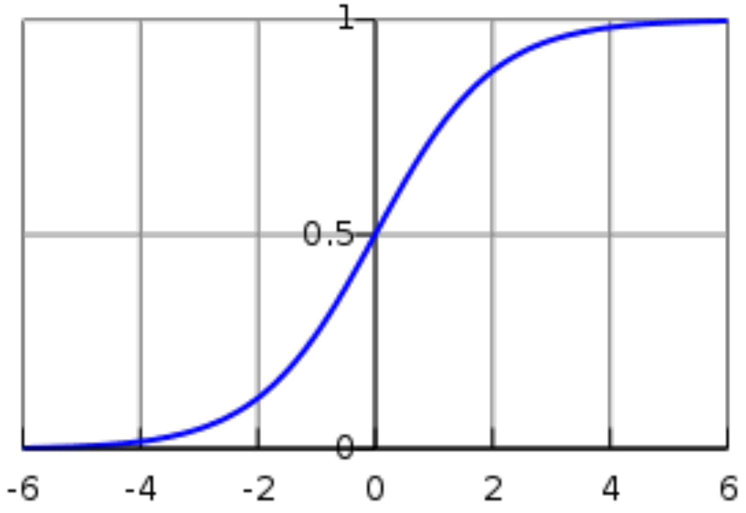

x
f_θ
$f_\theta(\boldsymbol{x})$
$f_\theta(\boldsymbol{x})$
$f_\theta(\boldsymbol{x})$
1
-1
g = sign
g = identity
1
0
g = logistic

$h_{\boldsymbol{\theta}}(\mathbf{x}) = \mathrm{sign}(f_{\boldsymbol{\theta}}(\mathbf{x})) = \mathrm{sign}(\boldsymbol{\theta}^T\mathbf{x})$

$h_{\boldsymbol{\theta}}(\mathbf{x}) = f_{\boldsymbol{\theta}}(\mathbf{x}) = \boldsymbol{\theta}^T\mathbf{x}$

$h_{\boldsymbol{\theta}}(\mathbf{x}) = \ell(f_{\boldsymbol{\theta}}(\mathbf{x})) = \ell(\boldsymbol{\theta}^T\mathbf{x}) = \frac{e^{(\boldsymbol{\theta}^T\mathbf{x})}}{1+e^{(\boldsymbol{\theta}^T\mathbf{x})}}$

The Logistic Function



$$\ell(z) = \frac{e^z}{1+e^z}$$

- Domain is R, Codomain is [0,1]
- Also known as sigmoid function do to its “S” shape or soft threshold (compared to hard threshold imposed by *sign*)
- When $z = \boldsymbol{\theta}^T\mathbf{x}$ we are applying a *non-linear* transformation to our *linear* signal
- Output can be *genuinely* interpreted as a probability value

Probabilistic Interpretation

$$h_{\boldsymbol{\theta}}(\mathbf{x}) = \ell(f_{\boldsymbol{\theta}}(\mathbf{x})) = \ell(\boldsymbol{\theta}^T\mathbf{x}) = \frac{e^{(\boldsymbol{\theta}^T\mathbf{x})}}{1+e^{(\boldsymbol{\theta}^T\mathbf{x})}}$$

- Describing the set of hypotheses using the logistic function is not enough to state that the output can be interpreted as a probability
  - All we know is that the logistic function always produce a real value between 0 and 1
  - Other functions may be defined having the same property
    - e.g., 1/π *arctan(x) + 1/2*
- The key points here are:
  - the output of the logistic function can be interpreted as a probability even during learning
  - the logistic function is mathematically convenient!

Probabilistic Interpretation: Odds Ratio

- Let *p* (resp., *q = 1-p*) be the probability of success (resp., failure) of an event
- *odds(success) = p/q = p/(1-p)*
- *odds(failure) = q/p = 1/p/q = 1/odds(success)*
- *logit(p) = ln(odds(success)) = ln(p/q) = ln(p/1-p)*
- Logistic Regression is in fact an ordinary linear regression where the logit is the response variable!

$$\mathrm{logit}(p) = \ln\left(\tfrac{p}{1-p}\right) = \theta_0 + \theta_1 x_1 + \ldots + \theta_d x_d = \boldsymbol{\theta}^T \mathbf{x}$$

- The coefficients of logistic regression are expressed in terms of the natural logarithm of odds

Probabilistic Interpretation: Odds Ratio

$$\text{logit}(p) = \ln\left(\tfrac{p}{1-p}\right) = \theta_0 + \theta_1 x_1 + \ldots + \theta_d x_d = \boldsymbol{\theta}^T \mathbf{x}$$

$$e^{\text{logit}(p)} = e^{\ln\left(\frac{p}{1-p}\right)} = \tfrac{p}{1-p} = e^{(\boldsymbol{\theta}^T \mathbf{x})}$$

$$p = e^{(\boldsymbol{\theta}^T \mathbf{x})}(1-p) = e^{(\boldsymbol{\theta}^T \mathbf{x})} - e^{(\boldsymbol{\theta}^T \mathbf{x})}p$$

$$p + e^{(\boldsymbol{\theta}^T \mathbf{x})}p = e^{(\boldsymbol{\theta}^T \mathbf{x})}$$

$$p(1 + e^{(\boldsymbol{\theta}^T \mathbf{x})}) = e^{(\boldsymbol{\theta}^T \mathbf{x})}$$

$$p = \tfrac{e^{(\boldsymbol{\theta}^T \mathbf{x})}}{1+e^{(\boldsymbol{\theta}^T \mathbf{x})}} = \tfrac{1}{e^{-(\boldsymbol{\theta}^T \mathbf{x})}+1}$$

Probabilistic-generated Data

As for any other supervised learning problem we can only deal with a finite set *D* of *m* labelled examples which we can try to learn from

$$\mathcal{D} = \{(\mathbf{x_1}, y_1)\}, \ldots, (\mathbf{x_m}, y_m)\}$$

where each y_i is a binary variable taking on two values {-1,+1}

That means we **do not** have access to the individual probability associated with each training sample!

Still we can assume that data we observe from *D*, i.e. **positive** (+1) and **negative** (-1) samples are actually generated by an *underlying and unknown probability function* (**noisy target**) which we want to estimate

Estimating the Noisy Target

More formally, given the generic training example (**x**,*y*) we claim there exists a conditional probability *P*(*y*|**x**), which is defined as:

$$P(y \mid \mathbf{x}) = \phi(\mathbf{x}) \text{ if } y = +1; 1 - \phi(\mathbf{x}) \text{ if } y = -1$$

where each ϕ is the noisy target function

- <u>Deterministic function</u>: given x as input it always outputs either *y = +1* or *y = -1* (mutually exclusive)
- <u>Noisy target function</u>: given *x* as input it always outputs both *y = +1* and *y = -1*, each with a "degree of certainty" associated

Goal: If we assume $\phi: R^{d+1} \rightarrow [0,1]$ is the underlying and unknown noisy target which generates our examples, our aim is to find an estimate ϕ^* which <u>best approximates</u> ϕ

Hypothesized Noisy Target

We claim that the best estimate ϕ^* of ϕ is $h^*_{\boldsymbol{\theta}}(\mathbf{x})$ which in turn is picked from the set of hypotheses defined by logistic function

$$\phi^*(\mathbf{x}) = h^*_{\boldsymbol{\theta}}(\mathbf{x}) = \ell(\boldsymbol{\theta}^T\mathbf{x}) \approx \phi(\mathbf{x})$$

But how do we select $h^*_{\boldsymbol{\theta}}(\mathbf{x})$?

2 elements are needed:

- Training set *D*
- Error Measure (Cost Function) to minimize

The Error Measure

The Best Hypothesis

If the hypothesis space *H* is made of a family of parametric models, $h^*_{\boldsymbol{\theta}}(\mathbf{x})$ can be picked as:

$$h^*_{\boldsymbol{\theta}} = \mathtt{argmax}_{h_{\boldsymbol{\theta}} \in \mathcal{H}} \; P(h_{\boldsymbol{\theta}} \mid \mathcal{D})$$

That is, we want to maximise the probability of the chosen hypothesis given the data *D* we observed

Flipping the Coin: (Data) Likelihood

We measure the error we are making by assuming that $h^*_{\boldsymbol{\theta}}(\mathbf{x})$ approximates the true noisy target ϕ

How likely is that the observed data *D* have been generated by our selected hypothesis $h^*_{\boldsymbol{\theta}}(\mathbf{x})$?

Find the hypothesis which maximises the probability of the observed data *D* given a particular hypothesis

$$h^*_{\boldsymbol{\theta}} = \mathtt{argmax}_{h_{\boldsymbol{\theta}} \in \mathcal{H}} \; P(\,\mathcal{D}\;|h_{\boldsymbol{\theta}}\,)$$

The Likelihood Function

Given a generic training example (**x**,*y*) and assuming it has been generated by a hypothesis $h_{\boldsymbol{\theta}}(\mathbf{x})$ the likelihood function is:

$$P(y \mid \mathbf{x}) = h_{\boldsymbol{\theta}}(\mathbf{x}) \text{ if } y = +1; 1 - h_{\boldsymbol{\theta}}(\mathbf{x}) \text{ if } y = -1$$

where ϕ has been replaced with our hypothesis

If we assume the hypothesis is the logistic function

$$h_{\boldsymbol{\theta}}(\mathbf{x}) = \ell(\boldsymbol{\theta}^T \mathbf{x})$$

And by noticing that logistic function is symmetric, i.e. $\ell(-z) = 1 - \ell(z)$, the likelihood for a single example is:

$$P(y \mid \mathbf{x}) = \ell(y \boldsymbol{\theta}^T \mathbf{x})$$

The Likelihood Function

Having access to a full set of *m* i.i.d. training examples *D*

$$\mathcal{D} = \{(\mathbf{x_1}, y_1)\}, \ldots, (\mathbf{x_m}, y_m)\}$$

The overall likelihood function is computed as:

$$\prod_{i=1}^{m} P(y_i \mid \mathbf{x_i}) = \prod_{i=1}^{m} \ell(y_i \boldsymbol{\theta}^T \mathbf{x_i})$$

Why Does Likelihood Make Sense?

How does the likelihood $\ell(y_i\boldsymbol{\theta}^T\mathbf{x}_i)$ changes w.r.t. the sign of y_i and $\boldsymbol{\theta}^T\mathbf{x}_i$?

	$\theta^T x_i > 0$	$\theta^T x_i < 0$
$y_i > 0$	≈ 1	≈ 0
$y_i < 0$	≈ 0	≈ 1

If the label is **concordant** with the signal (either positively or negatively) then $\ell(y_i\boldsymbol{\theta}^T\mathbf{x}_i)$ approaches to 1

Our prediction agrees with the true label

Conversely, if the label is **discordant** with the signal then $\ell(y_i\boldsymbol{\theta}^T\mathbf{x}_i)$ approaches to 0

Our prediction disagrees with the true label

Maximum Likelihood Estimate

Find the vector of parameters **θ** such that the likelihood function is maximum

$$\texttt{argmax}_{\boldsymbol{\theta}}\left(\prod_{i=1}^{m} P(y_i \mid \mathbf{x_i}) \right) = \texttt{argmax}_{\boldsymbol{\theta}}\left(\prod_{i=1}^{m} \ell(y_i \boldsymbol{\theta}^T \mathbf{x_i}) \right)$$

From MLE to In-Sample Error

Generally speaking, given a hypothesis $h_{\boldsymbol{\theta}}$ and a training set *D* of *m* labelled samples we are interested in measuring the “in-sample” (i.e. *training*) error

$$E_{\text{in}}(\boldsymbol{\theta}) = \frac{1}{m}\sum_{i=1}^{m} e(h_{\boldsymbol{\theta}}(\mathbf{x_i}), y_i)$$

where *e()* measures how “far” the chosen hypothesis is from the true observed value

How we can “transform” MLE to an expression similar to the “in-sample” error above?

From MLE to In-Sample Error

$$\mathtt{argmax}_{\boldsymbol{\theta}}\left(\prod_{i=1}^{m}\ell(y_i\boldsymbol{\theta}^T\mathbf{x_i})\right) \qquad \mathtt{argmax}_{\boldsymbol{\theta}}\left(\boxed{\frac{1}{m}\ln}\left(\prod_{i=1}^{m}\ell(y_i\boldsymbol{\theta}^T\mathbf{x_i})\right)\right)$$

$$\mathtt{argmax}_{\boldsymbol{\theta}}\left(\frac{1}{m}\ln\left(\prod_{i=1}^{m}\ell(y_i\boldsymbol{\theta}^T\mathbf{x_i})\right)\right) = \boxed{\mathtt{argmin}_{\boldsymbol{\theta}}\left(-\frac{1}{m}\ln\right.}\left(\prod_{i=1}^{m}\ell(y_i\boldsymbol{\theta}^T\mathbf{x_i})\right)\Bigg)$$

$$= \mathtt{argmin}_{\boldsymbol{\theta}}\left(-\frac{1}{m}\ln\left(\ell(y_1\boldsymbol{\theta}^T\mathbf{x_1})\right)-\ldots-\frac{1}{m}\ln\left(\ell(y_m\boldsymbol{\theta}^T\mathbf{x_m})\right)\right)$$

as $k\ \ln(a\cdot b) = k(\ln(a) + \ \ln(b)) = k\ \ln(a) + k\ \ln(b)$.

$$= \mathtt{argmin}_{\boldsymbol{\theta}}\left(\frac{1}{m}\ \sum_{i=1}^{m}\ -\ln(\ell(y_i\boldsymbol{\theta}^T\mathbf{x_i}))\right)$$

$$= \mathtt{argmin}_{\boldsymbol{\theta}}\left(\frac{1}{m}\ \sum_{i=1}^{m}\ \ln\left(\frac{1}{\ell(y_i\boldsymbol{\theta}^T\mathbf{x_i})}\right)\right)$$

as $-\ln(a) = \ln(\frac{1}{a})$.

From MLE to In-Sample Error

$$\boxed{\texttt{argmin}_{\boldsymbol{\theta}}\left(\frac{1}{m}\sum_{i=1}^{m}\ln\left(\frac{1}{\ell(y_i\boldsymbol{\theta}^T\mathbf{x_i})}\right)\right)}$$

By noticing that logistic function can be rewritten as follows:

$$\ell(z)=\frac{e^z}{1+e^z}=\frac{1}{e^{-z}+1}$$

We can finally write the "in-sample" error to be minimised:

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

Cross-Entropy Error

The Learning Algorithm

Picking the Best Hypothesis

So far we have defined:

- The model
- The error measure (cross-entropy)

To actually select the best hypothesis, we have to pick the vector of parameters so that the error measure is minimised

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

The usual way of achieving this is to compute the **gradient** with respect to **θ** (i.e. the vector of partial derivatives), set it to 0, and solve it for **θ**

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 **θ**

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

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

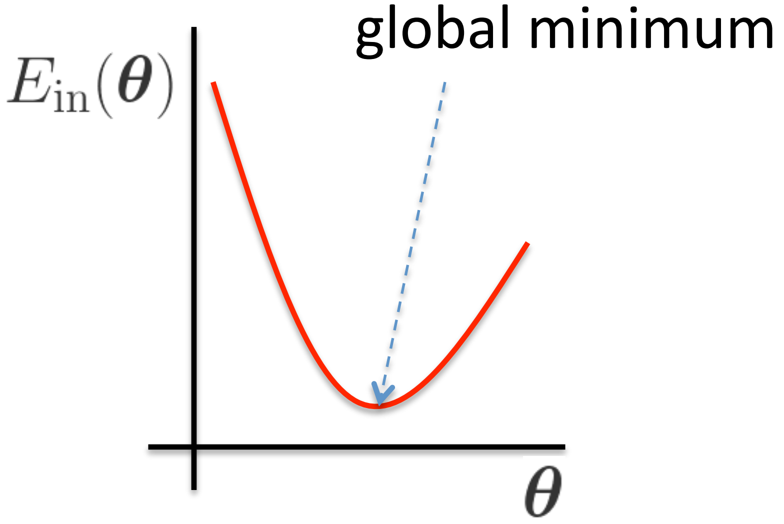
Iterative Solution

(Batch) Gradient Descent

General iterative method for any nonlinear optimization

Under specific assumptions on the function to be minimised and on the learning rate parameter at each iteration, the method **guarantees the convergence to a local minimum**

If the function is **convex** like the cross-entropy error for logistic regression then the local minimum is also the **global minimum**



Gradient Descent: The 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) + η**v**
   b. Return to 2.

Unit vector representing the direction of the steepest slope

$$\boxed{\boldsymbol{\theta}(t+1) = \boldsymbol{\theta}(t) + \underset{\substack{\uparrow \\ \text{step}}}{\eta}\mathbf{v}}$$

Question: How do we compute the direction **v**? Depending on how we solve it we may get different solutions (Gradient Descent, Conjugate Gradient, etc.)

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

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

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. **θ** = θ in R

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

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

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

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

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

Gradient Descent: The Direction **v**

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

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

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

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

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

The greek letter *nabla* indicates the gradient

Gradient Descent: The Direction **v**

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

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

Therefore:

- the **maximum** (i.e. **most positive**) step happens when both the error vector and the direction vector have the same direction
- the **minimum** (i.e. **most negative**) step happens when the two vectors have opposite direction

$$\nabla E_{\text{in}}(\boldsymbol{\theta}(t-1))^T \eta \mathbf{v} \geq -\eta \|\nabla E_{\text{in}}(\boldsymbol{\theta}(t-1))\|$$

Gradient Descent: The Direction **v**

$$\nabla E_{\text{in}}(\boldsymbol{\theta}(t-1))^T \eta \mathbf{v} \geq -\eta \|\nabla E_{\text{in}}(\boldsymbol{\theta}(t-1))\|$$

At each iteration t, we want the unit vector $\hat{\mathbf{v}}$ which makes exactly the most negative step

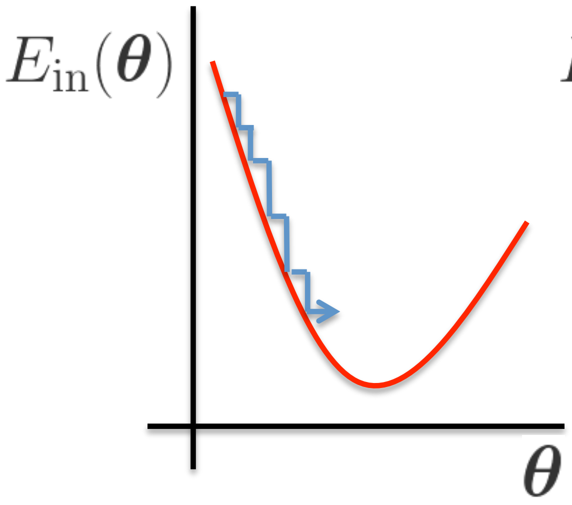
$$\nabla E_{\text{in}}(\boldsymbol{\theta}(t-1))^T \eta \hat{\mathbf{v}} = -\eta \|\nabla E_{\text{in}}(\boldsymbol{\theta}(t-1))\|$$

Therefore:

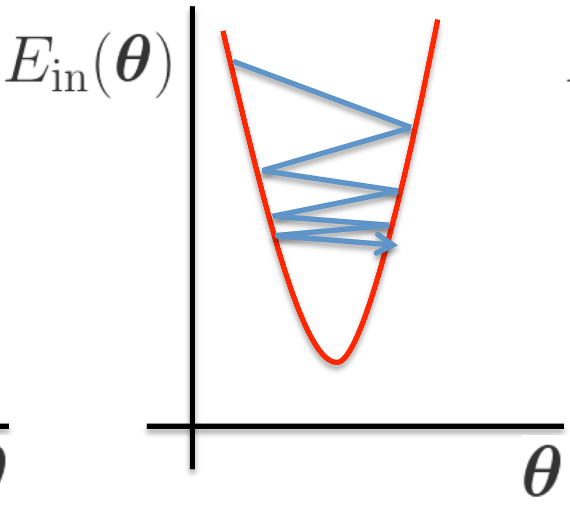
$$\hat{\mathbf{v}} = -\frac{\nabla E_{\text{in}}(\boldsymbol{\theta}(t-1))}{\|\nabla E_{\text{in}}(\boldsymbol{\theta}(t-1))\|}$$

Gradient Descent: The Step η

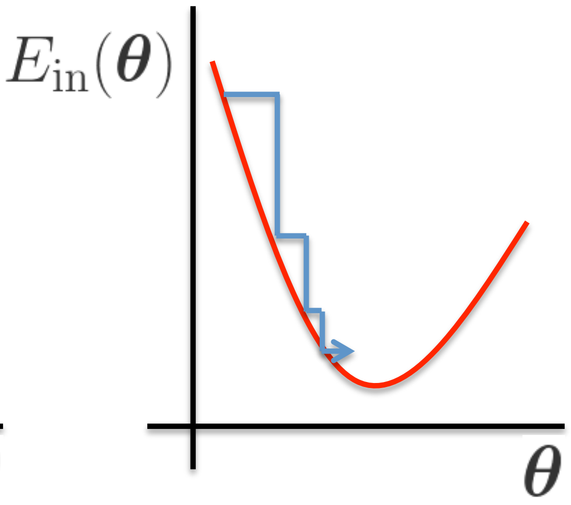
How the step magnitude η affects the convergence?



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:

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

where:

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

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_t

Instead of having a fixed η at each iteration, use a variable η_t as function of η

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

$$\eta_t = \eta k$$

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

If we take

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

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

Gradient Descent: The Algorithm

1. At t=0 initialize the (guessed) vector of parameters **θ** to **θ**(0)
2. For t = 0, 1, 2, … until stop:
   a. Compute the gradient of the cross-entropy error (i.e. the vector of partial derivatives)

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

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

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

Discussion: Initialization

- How do we choose the initial value of the parameters $\boldsymbol{\theta}(0)$?
- 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 $\boldsymbol{\theta}(0)$
  - Problem: we may miss "better" local minima (or even the global if it exists)
  - Solution (heuristic): repeating GD 100÷1,000 times each time with a different $\boldsymbol{\theta}(0)$ may give a sense of what is eventually the global minimum (no guarantees)

Discussion: Termination

- When does the algorithm stop?
- Intuitively, when $\boldsymbol{\theta}(t+1) = \boldsymbol{\theta}(t) \Rightarrow -\eta \nabla E_{in}(\boldsymbol{\theta}(t)) = 0 \Rightarrow \nabla E_{in}(\boldsymbol{\theta}(t)) = 0$
- If the function is convex we are guaranteed to reach the global minimum when $\nabla E_{in}(\boldsymbol{\theta}(t)) = 0$
  - i.e. there exists a unique local minimum which also happens to be the global minimum
- In general we don't know if eventually $\nabla E_{in}(\boldsymbol{\theta}(t)) = 0$ therefore we can use several criteria of termination, e.g.,:
  - stop whenever the difference between two iterations is "small enough" $\rightarrow$ may converge "prematurely"
  - stop when the error equals to $\varepsilon \rightarrow$ may not converge if the target error is not achievable
  - stop after T iterations
  - combinations of the above in practice works…

Advanced Topics

- Gradient Descent using second-order approximation
  - better local approximation than first-order but each step requires computing the second derivative (Hessian matrix)
  - Conjugate Gradient makes second-approximation "faster" as it doesn't require to compute explicitly the full Hessian matrix
- Stochastic Gradient Descent (SGD)
  - At each step only one sample is considered for computing the gradient of the error instead of the full training set
- L1 and L2 regularization to penalize extreme parameter values and deal with overfitting
  - include the L1 or L2 norm of the vector of parameters $\boldsymbol{\theta}$ in the cross-entropy error function to be minimised during learning