MAC 0460 / 5832
Introduction to Machine Learning

12 – Random topics

• Linear × non-linear • ✓
• Multiclass classification • softmax function • ✓
• Underfitting/Overfitting • Validation loss • ✓

IME/USP (23/05/2021)

Linear × Non-linear

↓

Perceptron

Regressão linear

Logistic Regression

$\underline{\sum_{i=1}^{d} w_i x_i + b}$

Linear × non-linear

$$\mathbf{x} = (x_1, x_2, \ldots, x_d)$$
$$\mathbf{w} = (w_1, w_2, \ldots, w_d) \in \mathbb{R}^d, \; b \in \mathbb{R}$$

Linear function:

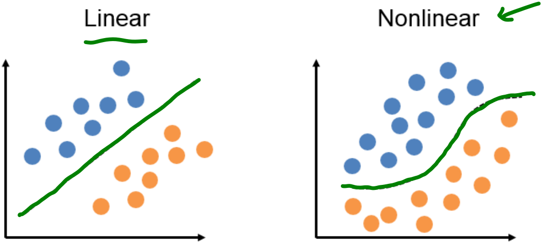
$$s = \underbrace{w_1 x_1 + w_2 x_2 + \ldots + w_d x_d + b}$$

Non-linear function – some examples:

$$s = w_1 x_1 + w_2 x_2 + w_3 \underline{x_1 x_2} + w_4 \underline{x_1^2} + w_5 \underline{x_2^2} + b$$
$$s = w_1 x_1^2 + w_2 x_2^2 + b$$

Any function $s : \mathbb{R}^d \to \mathbb{R}$ can be used for classification:

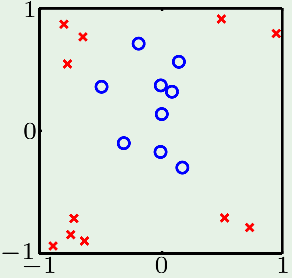
- $s < 0 \implies$ class := negative
- $s > 0 \implies$ class := positive
- $s = 0 \implies$ decision boundary



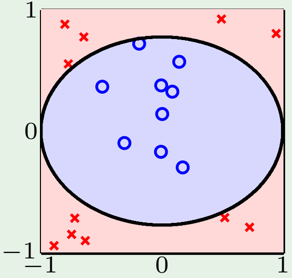
Fonte: https://jtsulliv.github.io/perceptron/

Linear is limited

Data:



Hypothesis:



Another example

Credit line is affected by 'years in residence'

but **not** in a linear way!

Nonlinear $[[x_i < 1]]$ and $[[x_i > 5]]$ are better.

Can we do that with linear models?

Linear in what?

Linear regression implements

$$\sum_{i=0}^{d} w_i \; x_i$$

fixos

variable.

Linear classification implements

$$\text{sign}\left(\sum_{i=0}^{d} w_i \; x_i\right)$$

Algorithms work because of linearity in the weights

$$f_2(x_1, x_2) = \underline{w_0 + w_1\, x_1^2 + w_2\, x_1\, x_2}$$

Non-linear with respect to $\underline{\underline{x_i}}$

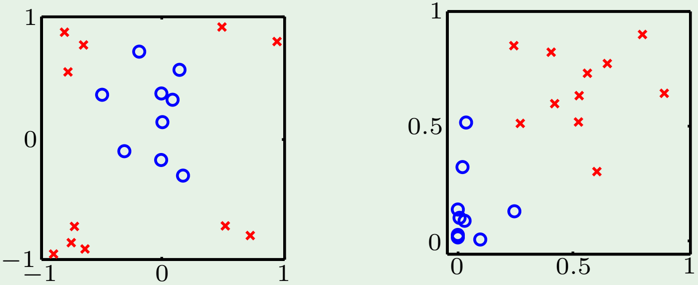
"constants"

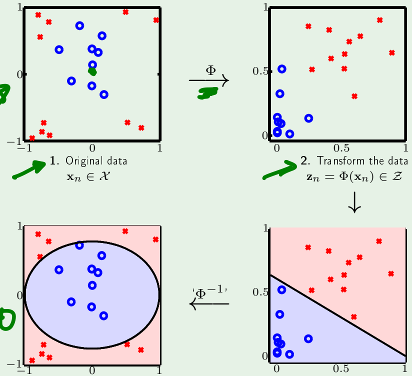
$$f_2(x_1, x_2) = \underline{w_0 + w_1 \boxed{x_1^2} + w_2 \boxed{x_1\, x_2}}$$

Linear with respect to w_i

Transform the data nonlinearly

$$(x_1, x_2) \xrightarrow{\Phi} (x_1^2, x_2^2)$$





1. Original data
$\mathbf{x}_n \in \mathcal{X}$

2. Transform the data
$\mathbf{z}_n = \Phi(\mathbf{x}_n) \in \mathcal{Z}$

3. Separate data in $\mathcal{Z}$-space
$\tilde{g}(\mathbf{z}) = \text{sign}(\tilde{\mathbf{w}}^{\mathsf{T}}\mathbf{z})$

4. Classify in $\mathcal{X}$-space
$g(\mathbf{x}) = \tilde{g}(\Phi(\mathbf{x})) = \text{sign}(\tilde{\mathbf{w}}^{\mathsf{T}}\Phi(\mathbf{x}))$

Nonlinear transforms

$$\tilde{d}$$

$$\underbrace{\mathbf{x} = (x_0, x_1, \cdots, x_d)}_{d} \xrightarrow{\Phi} \mathbf{z} = \underbrace{(z_0, z_1, \cdots\cdots\cdots\cdots, z_{\tilde{d}})}$$

Each $z_i = \phi_i(\mathbf{x})$ $\qquad$ $\underline{\mathbf{z} = \Phi(\mathbf{x})}$

Example: $\underbrace{\mathbf{z} = (1, x_1, x_2, x_1x_2, x_1^2, x_2^2)}$

Final hypothesis $g(\mathbf{x})$ in $\mathcal{X}$ space:

$$\underbrace{\operatorname{sign}\left(\tilde{\mathbf{w}}^{\mathsf{T}}\Phi(\mathbf{x})\right)} \qquad \text{or} \qquad \tilde{\mathbf{w}}^{\mathsf{T}}\Phi(\mathbf{x})$$

The price we pay

$$\mathbf{x} = (x_0, x_1, \cdots, x_d) \quad \xrightarrow{\Phi} \quad \mathbf{z} = (z_0, z_1, \cdots\cdots\cdots\cdots, z_{\tilde{d}})$$

$$\downarrow \qquad\qquad\qquad\qquad \downarrow$$

$$\mathbf{w} \qquad\qquad\qquad\qquad \tilde{\mathbf{w}}$$

$$d_{\mathrm{VC}} = d + 1 \qquad\qquad\qquad\qquad d_{\mathrm{VC}} \leq \tilde{d} + 1$$

- Linear models are simple but have limited ability to discriminate classes
- There are many non-linear algorithms

  Neural networks, decision trees, etc
- Non-linear transformation applied on the data

  A kind of feature transformation

  Some algorithms such as SVM explicitly explore this fact

  Neural net layers can be interpreted as input feature transformers

Multiclass classification

Multiclass classification

C classes problem

Approach 1: Combine multiple binary classifiers

- **OVA** (*One versus All*)
- **OVO** (*One versus One*)

OVA scheme (*One versus All*)

h_j

- one classifier for each class: h_j is a binary classifier designed to recognize objects of class j amongst all obejcts
- total of C binary classifiers: h_j, $j = 1, 2, \ldots, C$
- assume each classifier returns a score in $[0, 1]$
- **Decision**: given $\mathbf{x}$, let $\hat{y} = \arg\max_j \{h_j(\mathbf{x})\}$

OVO scheme (*One versus One*)

$j\ k$

- one classifier for each pair of classes: h_{jk} is a binary classifier trained using only examples from class j (positive) and k (negative)

- total of $\boxed{\frac{C(C-1)}{2}}$ binary classifiers: h_{jk}, $j < k$, $j, k = 1, 2, \ldots, C$ (note that for $k > j$, we have $h_{kj} = 1 - h_{jk}$)

- assume each classifier returns a score in $[0, 1]$ ✓

- **Decision**: given $\mathbf{x}$, let $\boxed{\hat{y} = \underset{j \in \{1,2,\ldots,C\}}{\arg\max} \left\{ \sum_{\substack{k=1 \\ j\neq k}}^{C} h_{jk}(\mathbf{x}) \right\}}$

Example of a binary classifier that outputs a score in $[0, 1]$

Logistic regression (sigmoid)

$$\hat{p}_1 = \hat{P}(y = 1|\mathbf{x}) = \theta(\mathbf{w}^T\mathbf{x}) = \frac{1}{1 + e^{-\mathbf{w}^T\mathbf{x}}}$$

Its output ($\hat{p}_1 = \hat{P}(y = 1|\mathbf{x})$) is interpreted as a probability

Note that OVA and OVO can be based on any type of binary classifiers. If the classifiers return a score value (that is, an estimate of $P(y|\mathbf{x})$), then the rules given earlier can be used.

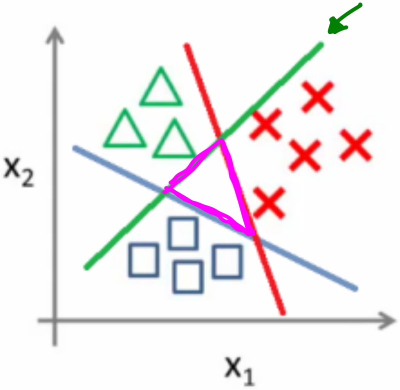
What if we use **hard classifiers** instead of **soft classifiers** ?

Hard classifier: output in $\{0, 1\}$ (class label y)

Soft classifier: output in $[0, 1]$ (conditional probability $P(y|\mathbf{x})$)

We can use, for instance, the **majority vote**

Voting may lead to **regions with undefined classification**



OVA

Fonte: https://utkuufuk.com/2018/06/03/one-vs-all-classification/

The triangular region at the center will receive no classification

There are many ways to **combine multiple binary classifiers** to implement multiclass classification.

(see for instance: *A review on the combination of binary classifiers in multiclass problems*, Ana C. Lorena, André C. P. L. F. de Carvalho, João M. P. Gama)

Similarly, **classifier combination** / **ensemble of classifiers** are topics vastly studied in the field of machine learning

(see for instance: *Combining Pattern Classifiers: Methods and Algorithms*, Ludmila I. Kuncheva)

Random Forest

It is not our goal here to discuss them exhaustively

Approach 2: Inherently multiclass algorithms

Any method that estimates the C conditionals $P(y = j \mid \mathbf{x})$, $j = 1, 2, \ldots, C$ at once

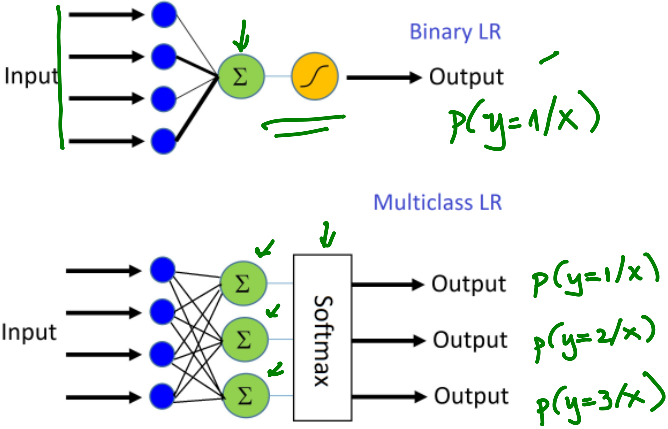
Multinomial logistic regression

The generalization of logistic regression for multiple classes is known as **multinomial logistic regression**

To estimate the conditional probabilities we use the **softmax function**:

$$\hat{p}_j = \underbrace{\hat{P}(y = j|\mathbf{x})} = \frac{e^{\mathbf{w}_j^T \mathbf{x}}}{\sum_{i=1}^{C} e^{\mathbf{w}_i^T \mathbf{x}}}, \quad j = 1, 2, \ldots, C$$

$P(y = +1/x)$



Fonte: https://www.cntk.ai/pythondocs/CNTK_103B_MNIST_LogisticRegression.html

Example for $C = 3$ classes:

$$\hat{p}_1 = \hat{P}(y = 1|\mathbf{x}) = \boxed{\frac{e^{\mathbf{w}_1^T\mathbf{x}}}{e^{\mathbf{w}_1^T\mathbf{x}} + e^{\mathbf{w}_2^T\mathbf{x}} + e^{\mathbf{w}_3^T\mathbf{x}}}}$$

$$\hat{p}_2 = \hat{P}(y = 2|\mathbf{x}) = \frac{e^{\mathbf{w}_2^T\mathbf{x}}}{e^{\mathbf{w}_1^T\mathbf{x}} + e^{\mathbf{w}_2^T\mathbf{x}} + e^{\mathbf{w}_3^T\mathbf{x}}}$$

$$\hat{p}_3 = \hat{P}(y = 3|\mathbf{x}) = \frac{e^{\mathbf{w}_3^T\mathbf{x}}}{e^{\mathbf{w}_1^T\mathbf{x}} + e^{\mathbf{w}_2^T\mathbf{x}} + e^{\mathbf{w}_3^T\mathbf{x}}}$$

Clearly $\hat{p}_1 + \hat{p}_2 + \hat{p}_3 = 1$ ✓

Also $0 \leq \hat{p}_j \leq 1$

Observe that in the binary classification case, we used

$$\hat{p}_1 = \hat{P}(y = 1|\mathbf{x}) = \theta(\mathbf{w}^T\mathbf{x}) = \frac{1}{1 + e^{-\mathbf{w}^T\mathbf{x}}}$$

It can be rewritten as:

$$\frac{1}{1 + e^{-\mathbf{w}^T\mathbf{x}}} = \frac{e^{\mathbf{w}^T\mathbf{x}}}{e^{\mathbf{w}^T\mathbf{x}}} \frac{1}{(1 + e^{-\mathbf{w}^T\mathbf{x}})} = \frac{e^{\mathbf{w}^T\mathbf{x}}}{e^{\mathbf{w}^T\mathbf{x}} + 1}$$

Hence:

$$\begin{aligned} \hat{p}_0 &= \hat{P}(y = 0|\mathbf{x}) = 1 - \hat{P}(y = 1|\mathbf{x}) \\ &= 1 - \frac{e^{\mathbf{w}^T\mathbf{x}}}{e^{\mathbf{w}^T\mathbf{x}} + 1} = \frac{1}{1 + e^{\mathbf{w}^T\mathbf{x}}} \end{aligned}$$

and

$$\hat{p}_1 + \hat{p}_0 = \hat{P}(y = 1|\mathbf{x}) + \hat{P}(y = 0|\mathbf{x}) = 1$$

Recall (previous page):

$$\hat{P}(y=0|\mathbf{x}) = \frac{1}{1+e^{\mathbf{w}^T\mathbf{x}}} \qquad \hat{P}(y=1|\mathbf{x}) = \frac{e^{\mathbf{w}^T\mathbf{x}}}{1+e^{\mathbf{w}^T\mathbf{x}}}$$

The **softmax formulation for two classes**:

$$\hat{P}(y=0|\mathbf{x}) = \frac{1}{(1+e^{\mathbf{w}^T\mathbf{x}})}\frac{e^{\mathbf{w}_0^T\mathbf{x}}}{e^{\mathbf{w}_0^T\mathbf{x}}} = \frac{e^{\mathbf{w}_0^T\mathbf{x}}}{e^{\mathbf{w}_0^T\mathbf{x}} + e^{(\mathbf{w}+\mathbf{w}_0)^T\mathbf{x}}}$$

$$\hat{P}(y=1|\mathbf{x}) = \frac{e^{\mathbf{w}^T\mathbf{x}}}{(1+e^{\mathbf{w}^T\mathbf{x}})}\frac{e^{\mathbf{w}_0^T\mathbf{x}}}{e^{\mathbf{w}_0^T\mathbf{x}}} = \frac{e^{(\mathbf{w}+\mathbf{w}_0)^T\mathbf{x}}}{e^{\mathbf{w}_0^T\mathbf{x}} + e^{(\mathbf{w}+\mathbf{w}_0)^T\mathbf{x}}}$$

$$\frac{1}{1+e^{-w^T x}}$$

Recall (previous page):

$$\hat{P}(y=0|\mathbf{x}) = \frac{1}{1+e^{\mathbf{w}^T\mathbf{x}}} \qquad \hat{P}(y=1|\mathbf{x}) = \frac{e^{\mathbf{w}^T\mathbf{x}}}{1+e^{\mathbf{w}^T\mathbf{x}}}$$

The **softmax formulation for two classes**:

$$\hat{P}(y=0|\mathbf{x}) = \frac{1}{(1+e^{\mathbf{w}^T\mathbf{x}})}\frac{e^{\mathbf{w}_0^T\mathbf{x}}}{e^{\mathbf{w}_0^T\mathbf{x}}} = \frac{e^{\mathbf{w}_0^T\mathbf{x}}}{e^{\mathbf{w}_0^T\mathbf{x}} + e^{(\underbrace{\boxed{\mathbf{w}+\mathbf{w}_0}}_{\mathbf{w}_1})^T\mathbf{x}}}$$

$$\hat{P}(y=1|\mathbf{x}) = \frac{e^{\mathbf{w}^T\mathbf{x}}}{(1+e^{\mathbf{w}^T\mathbf{x}})}\frac{e^{\mathbf{w}_0^T\mathbf{x}}}{e^{\mathbf{w}_0^T\mathbf{x}}} = \frac{e^{(\overbrace{\boxed{\mathbf{w}+\mathbf{w}_0}}^{\mathbf{w}_1})^T\mathbf{x}}}{e^{\mathbf{w}_0^T\mathbf{x}} + e^{(\underbrace{\boxed{\mathbf{w}+\mathbf{w}_0}}_{\mathbf{w}_1})^T\mathbf{x}}}$$

One-hot encoding of the output:

For each input $\mathbf{x}^{(i)}$, the output is a vector $\mathbf{y}^{(i)} = \underbrace{(y_1^{(i)}, y_2^{(i)}, \ldots, y_C^{(i)})}$ with $y_j^{(i)} = 1 \iff \mathbf{x}^{(i)}$ is from class j, $j = 1, 2, \ldots, C$

$(x^{(i)}, y^{(i)})$

$\in \{-1, +1\}$

Cross-entropy loss (wrt inputs $\mathbf{x}^{(i)} \in D$):

$$\sum_{i=1}^{N} \sum_{j=1}^{C} y_j^{(i)} \log \hat{p}_j^{(i)}$$

$(x^{(i)}, 3)$

Note that: $\hat{p}_j^{(i)} = \hat{P}(y^{(i)} = j \mid \mathbf{x}^{(i)})$, $\sum_{j=1}^{C} \hat{p}_j^{(i)} = 1$, and the parameters to be optimized, $\mathbf{w}_j$, are those in the softmax function $\dfrac{e^{\mathbf{w}_j^T \mathbf{x}}}{\sum_{i=1}^{C} e^{\mathbf{w}_i^T \mathbf{x}}}$

$$y^{(i)} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \leftarrow 3$$

Overfitting

Where we are

- We know what machine learning is

- We have learned some supervised learning algorithms

  Linear regression
  Perceptron
  Logistic regression

- We have seen that learning from data is feasible

  $|E_{in}(g) - E_{out}(g)|$ can be made arbitrarily small

What really matters: E_{out} (the error computed over the entire domain) – *out-of-sample error*

Generalization: We minimize E_{in} hoping to also minimize E_{out}.
We would like to have small E_{in} and E_{out} as close as possible to E_{in}

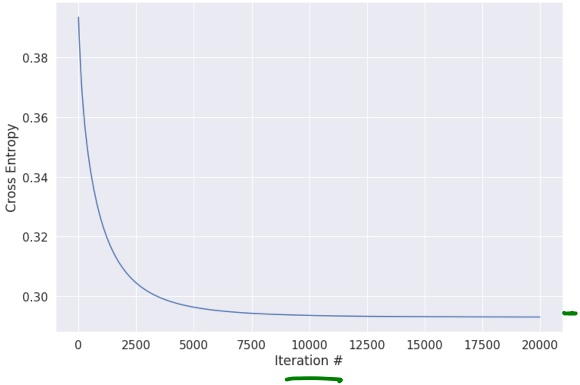
In general, the following equality holds:

$$\boxed{E_{out} \stackrel{\sim}{=} E_{in} + \text{generalization\_error}}$$

E_{in}

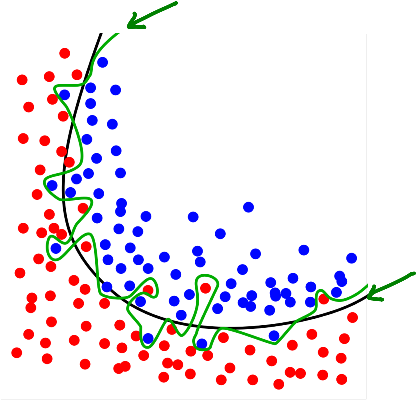
Training loss

E_{in} (loss / cost) usually decreases along the iteration (for instance, when we are employing *gradient descent*)



Overtraining

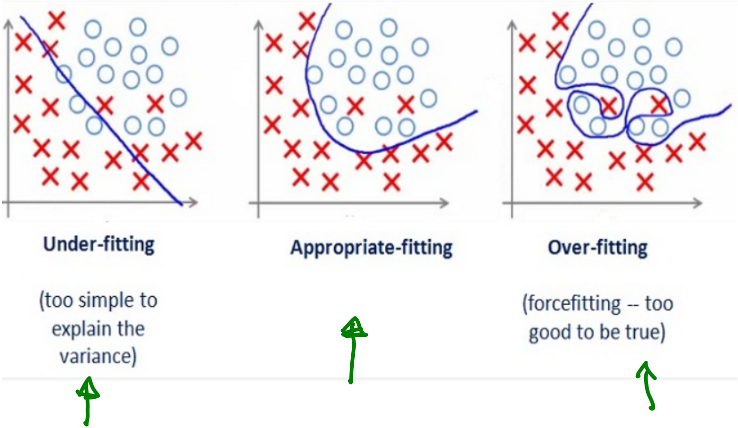
Overtraining may result in **overfitting**



green illustrates overfitting

Fonte: Wikipedia

It is not just about number of iterations. It is also related to model complexity



E_{out}

Since in practice we can not compute E_{out}, we can use an independent set of examples (validation set) and compute the cost on it, E_{val}

E_{val} can be thought as a proxy of E_{out}

E_{val} provides a hint on whether we are doing the right thing or not

Validation error and overfitting

~ Eval

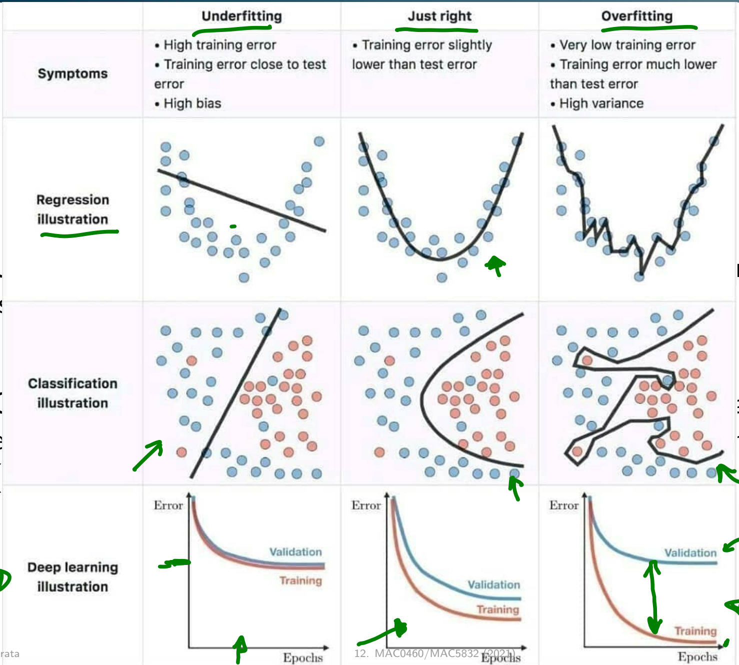
VC dimension

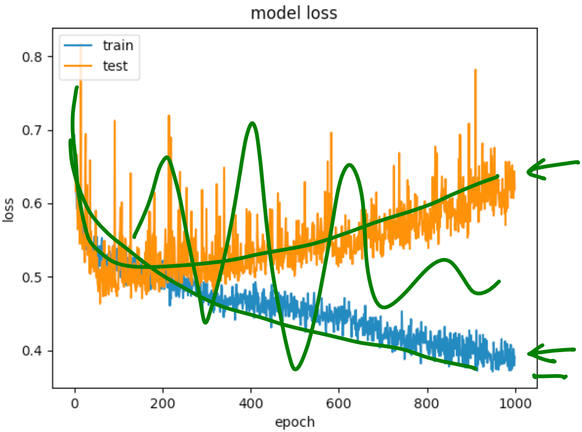
$$\cancel{E_{out}} = E_{in} + \text{generalization\_error}$$

Underfitting: large E_{in} and E_{val} indicate strong model *bias* (model is too constrained)

Overfitting: when the curves of E_{in} and E_{val} start to get separated each other along the iterations, it is an indication of overfitting (model is too sophisticated)

Validation error and overfitting





Learning curve example: deep neural network

(test = error on validation set)

(Learning curve: not the same concept seen in the last class)

Prof. Mostafa's lectures:

Lecture 11: Overfitting
Lecture 12: Regularization
Lecture 13: Validation

How to deal with overfitting :

- **regularization** – add a penalty term in the cost function (to be seen later)
- **validation** – error on the validation set, E_{val}, can be used to choose a family of hypotheses $\mathcal{H}$ of "right complexity"

Validation versus regularization

In one form or another, $\boxed{E_{\text{out}}(h) \;=\; E_{\text{in}}(h) \;+\; \text{overfit penalty}}$

Regularization:

$$E_{\text{out}}(h) \;=\; E_{\text{in}}(h) + \underbrace{\text{overfit penalty}}_{\text{regularization estimates this quantity}}$$

Validation:

$$\underbrace{E_{\text{out}}(h)}_{\text{validation estimates this quantity}} = \; E_{\text{in}}(h) \;+\; \text{overfit penalty}$$

CV cross-validation