Foundations

Course of Machine Learning
Master Degree in Computer Science

University of Rome "Tor Vergata"

a.a. 2020-2021

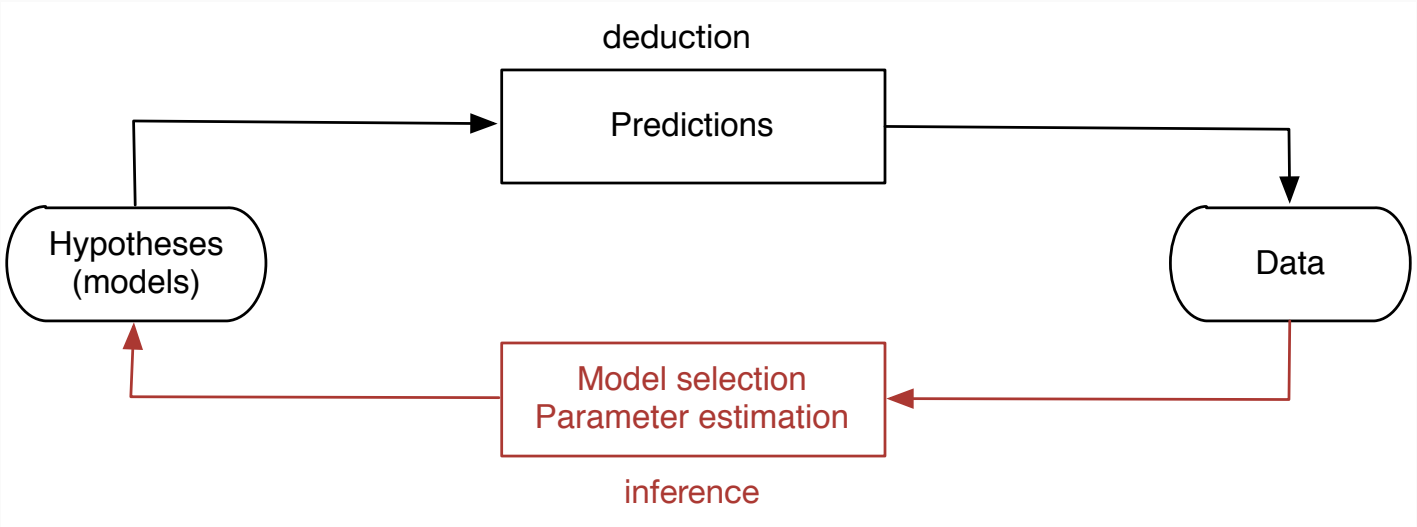
Giorgio Gambosi

Objectives

Machine learning: inductive approach

Learning of commonalities through analysis of a set of examples (training set), which is assumed to be available.

- A training set of n items is represented as a set of input vectors $\mathbf{x}_1, \ldots, \mathbf{x}_n$, used to derive a model.
- If the purpose is item classification with respect to a collection of predefined classes, the training set also includes a target vector $\mathbf{t} = \{t_1, \ldots, t_n\}$, where the class of each training set item is specified.



Types of problems

Supervised learning

- We want to predict, given the values of a set (features) of an item $\mathbf{x}$, the unknown value of an additional feature target of the item
  - Target in $\mathbb{R}$: regression. Target in $\{1, \ldots, K\}$: classification.
- General approach: defined (by means of learning from a set of examples) a model of the relation between feature and target values.
- The training set $\mathbf{X}, \mathbf{t}$ includes a feature vector $\mathbf{x}_i = \{x_{i1}, \ldots, x_{im}\}$ and the corresponding target t_i for each item.
- The model could be:
  1. a function $y()$ which, for any item $\mathbf{x}$, returns a value $y(\mathbf{x})$ as an estimate of t
  2. a probability distribution which associates to each possible value $\overline{y}$ in the target domain, the corresponding probability $p(y = \overline{y}|\mathbf{x})$

Types of problems

Unsupervised learning

- We wish to extract, froma a given collection of items dataset) $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$, with no target associated, some synthetic information, such as:
  - subsets of similar items clustering)
  - the distribution of items in their domain (density estimation)
  - the projection, as informative as possible, of items on lower dimensional subspaces, that is, their characterization by means of a smaller set of features feature selection, feature extraction)
- A suitable model, of just the data features, is usually defined and applied also in the case of unsupervised learning.

Reinforcement learning

- We want to identify, in a given framework, a sequence of actions to be performed in order to maximize a certain profit
- As in supervised learning, no examples are given, but an environment is available which returns a profit in correspondance to the execution of any action

Machine learning framework: domains

Domain set $\mathcal{X}$: Set of objects we may wish to label. Each object is modeled as a vector of features. The number of features is the dimensionality of the problem

Label set $\mathcal{Y}$: Set of possible label values associated to objects in $\mathcal{X}$.

- $\mathcal{Y}$ continuous: regression
- $\mathcal{Y}$ discrete: classification

Assume for the moment that $|\mathcal{Y}| = 2$, this is case of binary classification

Machine learning framework: learner input

Training set $\mathcal{T}$: A set of object-label pairs: $\mathcal{T} = \{(\mathbf{x}_1, t_1), \ldots, (\mathbf{x}_n, t_n)\}$. We shall usually denote as $\mathbf{X}$ the matrix of objects (feature matrix), that is

$$\mathbf{X} = \begin{pmatrix} - & \mathbf{x}_1 & - \\ & \vdots & \\ - & \mathbf{x}_n & - \end{pmatrix}$$

and as $\mathbf{t}$ the vector of labels (target vector), that is

$$\mathbf{t} = \begin{pmatrix} t_1 \\ \vdots \\ t_n \end{pmatrix}$$

Machine learning framework: learner output

Learner output: The learner (an algorithm A) is requested to return, for a given training set $\mathcal{T}$, a prediction rule (classifier, regressor) $A(\mathcal{T}) = h : \mathcal{X} \mapsto \mathcal{Y}$

Machine learning framework: learner evaluation

Training objects generation model: We assume that the objects observed in the training set is a sample from $\mathcal{X}$, generated according to some probability distribution $\mathcal{D}$. That is, for any $\mathbf{x} \in \mathcal{X}$, $p_{\mathcal{D}}(\mathbf{x})$ is the probability that $\mathbf{x}$ is the next object sampled in the training set

Training targets generation model: In the general case, we assume the labels associated to the objects in the training set are generated according to a probability distribution $\mathcal{D}'$ conditional on $\mathcal{X}$. That is, for any $t \in \mathcal{Y}$, $p_{\mathcal{D}'}(t|\mathbf{x})$ is the probability that the observed label of object $\mathbf{x}$ in the training set is t. For the moment, we shall assume that the relation between object and label is deterministic, that is it exists an unknown function f such that $t_i = f(\mathbf{x}_i)$

Machine learning framework: learner evaluation

Error: The error of a predictor h derives in general from the comparison of its predictions $h(\mathbf{x})$ and the given labels y for all objects $\mathbf{x}_i$ in the domain $\mathcal{X}$.

Loss: The comparison is performed by applying a predefined loss function $L : \mathcal{Y} \times \mathcal{Y} \mapsto \mathbb{R}$ to all pairs *predicted label - target*.

Risk: The error of a predictor h is defined in terms of risk (or generalization error), the expected loss on all objects in $\mathcal{X}$

$$\mathcal{R}(h) = E_{\mathcal{D},f}[L(h(\mathbf{x}), f(\mathbf{x}))] = L(h(\mathbf{x}), f(\mathbf{x})) \cdot p_{\mathcal{D}}(\mathbf{x})$$

In the general case, this is defined as

$$\mathcal{R}(h) = E_{\mathcal{D},\mathcal{D}'}[L(h(\mathbf{x}), y)] = L(h(\mathbf{x}), y) \cdot p_{\mathcal{D}}(\mathbf{x}) \cdot p_{\mathcal{D}'}(y|\mathbf{x})$$

Machine learning framework: learner evaluation

Since $\mathcal{D}$ and $\mathcal{D}'$ (or f) are not known, the risk can only be estimated from the data available (the training set $\mathcal{T}$).

Empirical risk: The risk can be estimated from the training set by estimating the expectation of the loss function as the average loss on the set.

$$\overline{\mathcal{R}}_{\mathcal{T}}(h) = \frac{1}{|\mathcal{T}|} \sum_{(x,t)\in\mathcal{T}} L(h(x), t)$$

Machine learning framework: from learning to optimization

The fundamental approach in machine learning is deriving a predictor h which (at least approximately) minimizes the empirical risk computed on the available training set.

A learning problem is then reduced to a minimization problem in some functional space $\mathcal{H}$, the set of all possible predictors h.

$$h^* = \underset{h \in \mathcal{H}}{\operatorname{argmin}} \, \overline{\mathcal{R}}_{\mathcal{T}}(h)$$

Here, $\mathcal{H}$ is the set of hypotheses or inductive bias

Issues related to the inductive bias

The choice of the set of hypotheses is an important issue in ML:

- what is the effect of the structure and size of $\mathcal{H}$?
- how to define $\mathcal{H}$ in such a way to make it feasible to compute h^*?

Choice of $\mathcal{H}$

- The hypotheses class $\mathcal{H}$ can be viewed as reflecting some prior knowledge that the learner has about the task
  - a belief that one of the members of the class $\mathcal{H}$ is a low-error model for the task
- A trivial way of pursuing this goal would be to define a very rich class, that is assuming that many possible functions belong to $\mathcal{H}$
- As a limit, $\mathcal{H}$ could be defined just as the set of all functions $f : \mathcal{X} \mapsto \mathcal{Y}$

Choice of $\mathcal{H}$

Problem with large $\mathcal{H}$:

- Assume a binary classification problem with training set $\mathcal{T} = (\mathbf{X}, \mathbf{t})$, with $0/1$ loss

$$L(y, t) = \begin{cases} 0 & \text{if } y = t \\ 1 & \text{otherwise} \end{cases}$$

  that is, the loss is 1 if the item is misclassified, 0 otherwise. As a consequence, the risk is the expected number of classification errors, while the empirical risk is the fraction of items in the training set which are misclassified.
- Assume $p(t = 1|\mathbf{x}) = \frac{1}{2}$ for $\mathbf{x} \in \mathcal{X}$, that is, the two classes have same size in the population

Choice of $\mathcal{H}$

Consider the classification function defined as:

$$h(x) = \begin{cases} 1 & \text{if } \mathbf{x} = \mathbf{x}_i \in \mathbf{X}, t_i = 1 \\ 0 & \text{otherwise} \end{cases}$$

that is, h assigns to class 1 all items labeled as 1 in the training set. All other items are classified as 0.

Clearly, the empirical risk here is 0 by definition, but the risk is $\approx \frac{1}{2}$. When applied to a dataset randomly sampled from the population, the quality of h^* is the same of a function which randomly assigns items to classes.

This is called overfitting: the classification method behaves well on the training set, but poorly on new data from the population.

Choice of $\mathcal{H}$

With respect to $\mathcal{H}$, the following considerations can be done:

- If $\mathcal{H}$ is too large (complex), overfitting may occur: a function which behaves very well on the training set may be available which however performs poorly on new data
- If $\mathcal{H}$ is too small (simple), underfitting may occur: no function behaving in a satisfactory way, both on the training set and on new sets of data, is available in $\mathcal{H}$

This is related to the so-called bias variance tradeoff

Bias vs variance

The risk associated to the h^*, the predictor which minimizes the empirical risk, can be decomposed in two parts:

$$\mathcal{R}(h^*) = \epsilon_B + \epsilon_V$$

where:

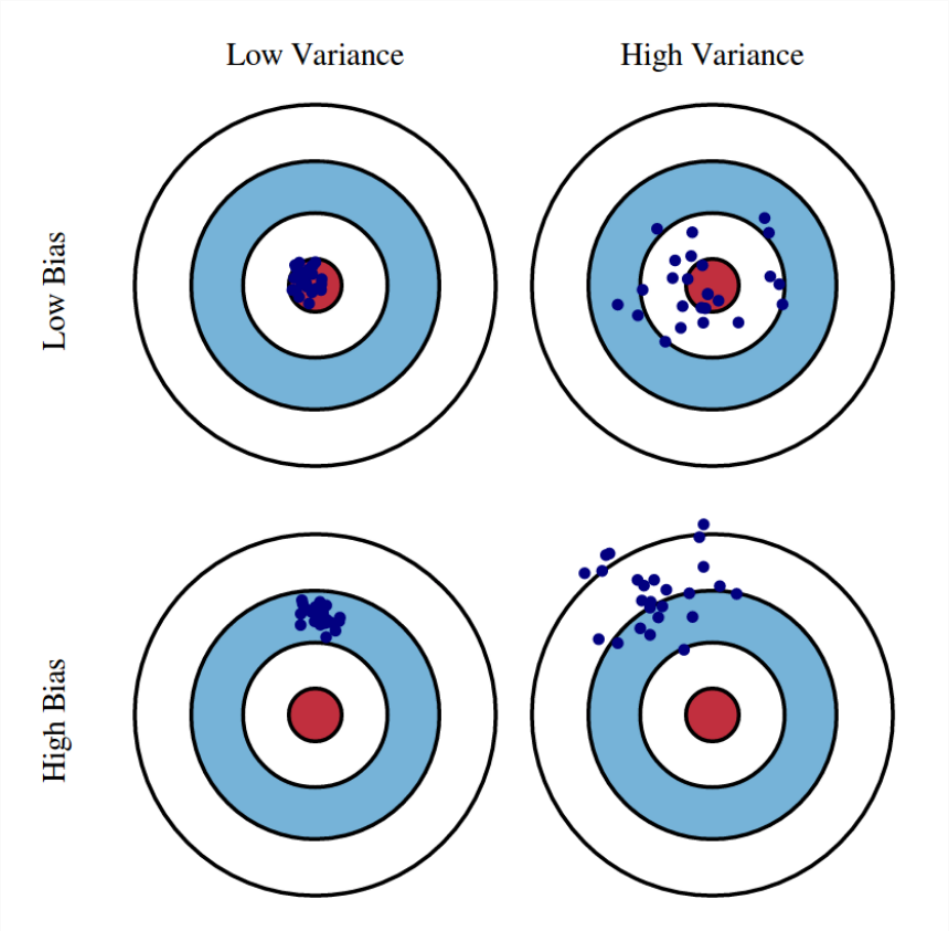
- ϵ_B is the minimum risk achievable by any $h \in \mathcal{H}$: this is only determined by the inductive bias, and independent from the training set. It is a property of the class of hypotheses considered with respect to the prediction task. This is called bias
- ϵ_V is the difference between the above minimum risk in $\mathcal{H}$ and the risk associated to the best predictor in $\mathcal{H}$ with respect to the training set: it is related to the fact that empirical risk minimization only provides an estimate of the best predictor achievable for the given inductive bias. It is a measure of how well the predictor computed from a particular training set approximates the best possible one. Its expectation with respect to all possible training sets is a measure of how much a predictor derived from a random training set may result in poorer performances with respect to the best possible one. This is called variance

Bias vs variance

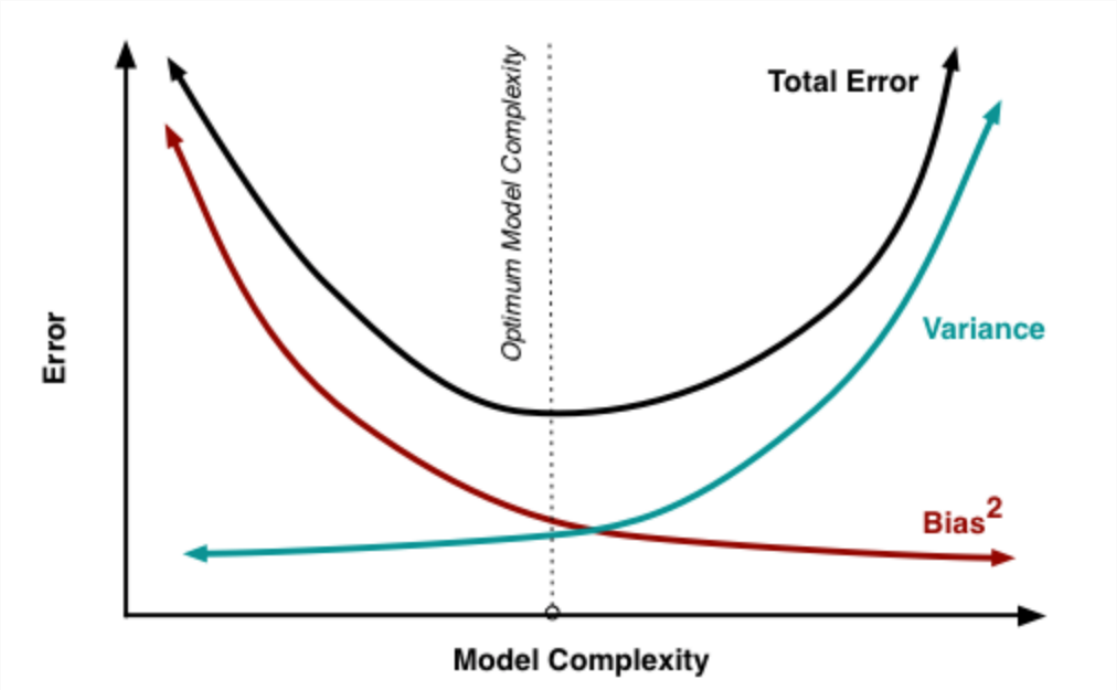
The choice of $\mathcal{H}$ is subject to a bias-variance tradeoff: higher bias tends to induce lower variance, and vice versa.

- High bias and low variance implies that all predictors which can be obtained from different training sets tend to behave similarly, with a similar risk (low variance). However, all of them then to behave poorly (high bias), since $\mathcal{H}$ is too poor to include a satisfactory predictor for the task considered. This results into underfitting
- Low bias and high variance implies that lot of predictors are available in $\mathcal{H}$, and among them a good one is usually avaliable (low bias). However, quite different predictors can be obtained from different training sets, which implies that it may easily happen that, while a very good performance can be obtained on the training set, the resulting predictor can behave quite differently and more poorly that the best possible one, which implies overfitting

Bias vs variance



Bias vs variance



Computing h^*

- The optimization required to derive h^* can be complex in the general case, when a function must be derived in a function space.
- Usually, the situation is made easier by considering $\mathcal{H}$ as a space of functions parameterized by a suitable set of coefficients (for example, all polynomials of degree at most d, for a given d): this results in a minimization to be performed over a set of d-dimensional points.
- That is, $\mathcal{H} = \{h_\theta | \theta \in \Theta\}$, where Θ is the coefficients domain and h is a function template, parameterized by elements in Θ
- Minimizing the Empirical risk results into computing

$$\theta^* = \underset{\theta \in \Theta}{\operatorname{argmin}} \overline{\mathcal{R}}_{\mathcal{T}}(h_\theta)$$

Computing h^*

- In most cases, $\Theta = \mathbb{R}^d$ for some $d > 0$: in this case, the minimization of $\overline{\mathcal{R}}_{\mathcal{T}}(h_\theta)$ is unconstrained and a (at least local) minimum could be computed setting all partial derivatives to 0

$$\frac{\partial}{\partial \theta_i} \overline{\mathcal{R}}_{\mathcal{T}}(h_\theta) = 0$$

  that is, setting to zero the gradient of the empirical risk with respect to the vector of parameters θ

$$\nabla_\theta \overline{\mathcal{R}}_{\mathcal{T}}(h_\theta) = \mathbf{0}$$

- The analytical solution of this set of equations is usually quite hard
- Numerical methods can be applied

Gradient descent

- Gradient descent performs minimization of a function $J(\theta)$ through iterative updates of the current value of θ (starting from an initial value $\theta^{(0)}$) in the opposite direction to the one specified by the current value of the gradient $J'(\theta) = \nabla_\theta J(\theta)$

$$\theta^{(k+1)} = \theta^{(k)} - \eta \nabla_\theta J(\theta) \mid_{\theta=\theta^{(k)}}$$

that is, for each parameter θ_i

$$\theta^{(k+1)} = \theta^{(k)} - \eta \frac{\partial J(\theta)}{\partial \theta_i} \Big|_{\theta=\theta^{(k)}}$$

- η is a tunable parameter, which controls the amount of update performed at each step

Gradient descent and ERM

In Machine learning, minimization of the Empirical Risk is performed, hence gradient descent takes the form

$$\theta_i^{(k+1)} = \theta_i^{(k)} - \eta \frac{\partial}{\partial \theta_i} \frac{1}{|\mathcal{T}|} \sum_{(x,t)\in\mathcal{T}} L(h_\theta(x), t)|_{\theta=\theta^{(k)}}$$

$$= \theta_i^{(k)} - \frac{\eta}{|\mathcal{T}|} \sum_{(x,t)\in\mathcal{T}} \frac{\partial}{\partial \theta_i} L(h_\theta(x), t)|_{\theta=\theta^{(k)}}$$

This is called batch gradient descent: observe that, at each step, all items in the training set must be considered

Stochastic gradient descent

Batch gradient descent can be modified by performing the update, at each step, on the basis of the evaluation at a single item of the training set for single parameters,

$$\theta_i^{(k+1)} = \theta_i^{(k)} - \eta \frac{\partial}{\partial \theta_i} L(h_\theta(x_j), t_j)|_{\theta=\theta^{(k)}}$$

Mini-batch gradient descent

An intermediate case is the one when a subset of the items in the training is considered at each step

$$\theta_i^{(k+1)} = \theta_i^{(k)} - \frac{\eta}{|B_r|} \sum_{(x,t) \in B_r} \frac{\partial}{\partial \theta_i} L(h_\theta(x), t)|_{\theta=\theta^{(k)}}$$

This is called mini-batch gradient descent

Momentum gradient descent

- Based on a physical interpretation of the optimization process: a body of mass $m = 1$ is moving on the surface of a cost function $J(\theta)$, with potential energy $U(\theta) = \eta J(\theta)$ and weight force (or acceleration, since $m = 1$) $F(\theta) = -\nabla U(\theta) = -\eta \nabla J(\theta)$, at any point θ
- In gradient descent, the movement of the body is determined by the acceleration at that point, that is by the gradient $\nabla J(\theta)$
- In momentum gradient descent, the velocity $v(\theta)$ of the body is considered: the movement of the body is determined by the velocity, that is,

$$\theta^{(k+1)} = \theta^{(k)} + v^{(k+1)}$$

with the velocity changing as determined by the acceleration

$$v^{(k+1)} = v^{(k)} - \eta \nabla J(\theta^{(k)})$$

Momentum gradient descent

This results into

$$v^{(k+1)} = -\eta \nabla J(\theta^{(k)}) + v^{(k)} = -\eta \nabla J(\theta^{(k)}) - \eta \nabla J(\theta^{(k-1)}) + v^{(k-1)} = \cdots$$
$$= -\eta \sum_{i=0}^{k} \nabla J(\theta^{(i)}) + v^{(0)}$$

$$\theta^{(k+1)} = \theta^{(k)} + v^{(k+1)} = \theta^{(k)} - \eta \sum_{i=0}^{k} \nabla J(\theta^{(i)}) + v^{(0)}$$

Nesterov gradient descent

- The same approach of momentum gradient descent is applied, with the gradient estimation performed not at the current point $\theta^{(k)}$, but approximately at the next point $\theta^{(k+1)}$
- The approximation derives by considering $\theta^{(k)} + \gamma v^{(k)}$ instead of $\theta^{(k+1)}$
- The updates of v and θ are considered in advance with respect to momentum GD

$$v^{(k+1)} = \gamma v^{(k)} + \eta \nabla J(\tilde{\theta}^{(k)}) = \gamma v^{(k)} + \eta \nabla J(\theta^{(k)} + \gamma v^{(k)})$$
$$\theta^{(k+1)} = \theta^{(k)} + v^{(k+1)}$$

Adagrad

- In Adagrad, different learning rates are applied to the different parameters θ_i: larger gradients in the preceding steps results into smaller rates
- in particular, the learning rate of θ_j at step k is defined as

$$\eta_j^{(k)} = \frac{\eta}{\sqrt{G_{j,k} + \varepsilon}}$$

  where η is a constant, $G_{j,k} = \sum_{i=0}^{k} g_{j,i}^2$ is the sum of the squared past gradients $g_{j,i} = \frac{\partial J(\theta, X)}{\partial \theta_j}\Big|_{\theta=\theta^{(i)}}$ and ε is a small *smoothing* constant, to deal with possible null denominators
- The update of θ_j at step $k+1$ is then

$$\theta_j^{(k+1)} = \theta_j^{(k)} - \frac{\eta}{\sqrt{G_{j,k} + \varepsilon}} g_{j,k}$$

- Learning rates decrease at each step, with the ones associated to parameters which had large gradients in the past decreasing more: that is, parameters which tended to large variation at each step are more "pushed" towards a stable value

Adadelta

- In Adadelta, past gradients are considered with a decreasing relevance of long past ones:
- A (decay) is applied my means of a coefficient $0 < \gamma < 1$

$$G_{j,k} = \gamma G_{j,k-1} + (1 - \gamma) g_{j,k}^2$$

Second order methods

- Maxima (or minima) of $J(\theta)$can be found by searching points where the gradient (all partial derivatives) zero
- Any iterative method to compute zeros of a function (such as Newton-Raphson) can the be applied on the gradient $\nabla J(\theta)$
- Such methods rely on the gradient of the function considered, hence, the gradient of $\nabla J(\theta)$ (that is, the Hessian) must be computed

$$H_{ij}(J(\theta)) = \frac{\partial^2 J(\theta)}{\partial\theta_i \partial\theta_j}$$

- At each step, the following iteration is applied (in case Newton-Raphson method is used)

$$\theta^{(k+1)} = \theta^{(k)} - \left(H(J(\theta))^{-1}\nabla J(\theta)\right)\Big|_{\theta_{(k)}}$$

Probabilistic approaches

In this case, a generative approach is followed

1. we assume that the observed dataset (features and target) has been derived by randomly sampling:
   - $\mathcal{X}$ according to the probability distribution $p_{\mathcal{D}}(x)$ (usually the uniform distribution)
   - $\mathcal{Y}$ according to the conditional distribution $p_{\mathcal{D}'}(y|x)$
2. we may then consider a class of possible conditional distributions and
3. select (infer) the "best" conditional distribution $\overline{\mathcal{D}}'$ from the dataset
4. given any new item x, apply $p_{\overline{\mathcal{D}}'}(y|x)$ to provide probabilities for each possible value of the corresponding target
5. an independent decision strategy must be applied to $p_{\mathcal{D}'}(y|x)$ to return a specific prediction $h(x)$

Inferring a best distribution

- how to define the class of possible distributions?
- what is a measure of the quality of the distribution (given the dataset)?
  - this is related to how a dataset generated sampling from $\mathcal{D}$ (usually uniform) and $\mathcal{D}'$ could be similar to the available dataset $\mathcal{T} = (X, t)$

A different approach

- instead of finding a best distribution and use it to predict probabilities, we could:
- compose the predictions provided by all possible distributions to derive a final prediction