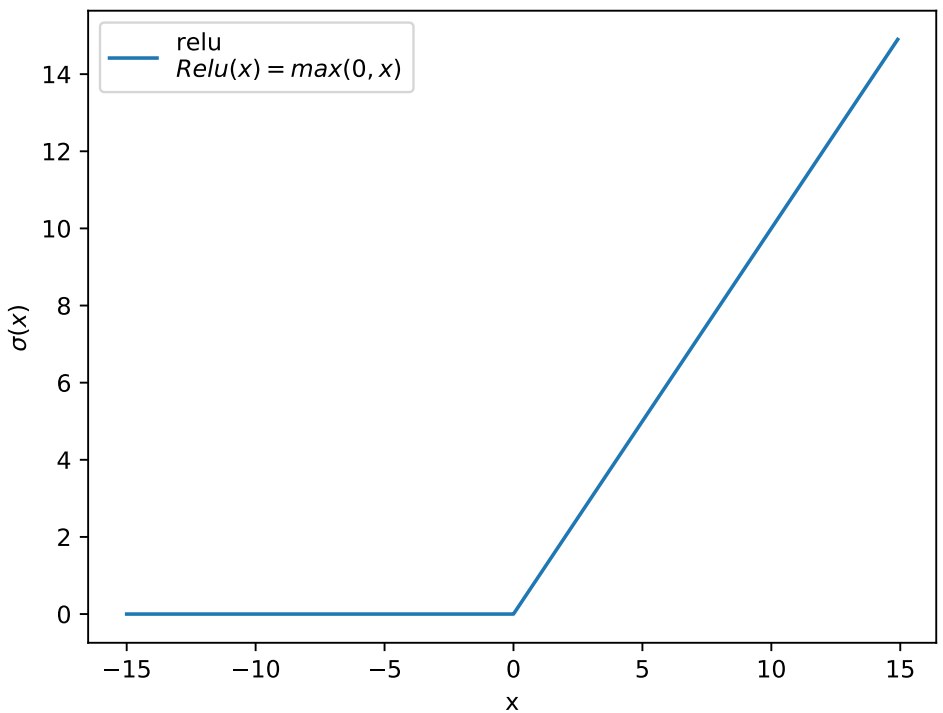
Fondamentaux théoriques du machine learning



Overview of lecture 9

Course summary

SAG (TP5)

Scoring, multiclass problems, cross validation

- Regression
- Binary classification
- Multi-class classification
- Cross-validation

Neural networks

- A space of functions
- Chain rule

Project

PTML project description.

lecture notes.pdf

Updated yesterday (added Bayes predictor for binary classification and other edits).

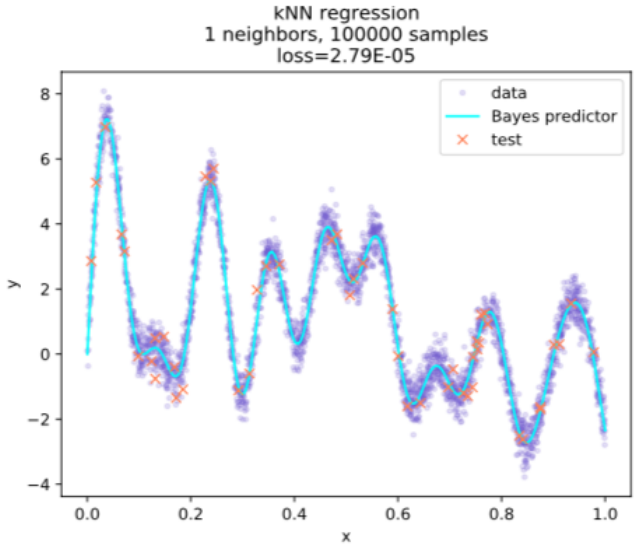


TABLE DES MATIÈRES

1 General mathematical results 3
1.1 Linear algebra . . . 3
1.1.1 Matricial writing of inner products . . . 3
1.2 Statistics and Probability theory . . . 4
1.2.1 Expected value, variance . . . 4
1.2.2 Sample mean, sample variance, sample covariance . 6
1.2.3 Markov and Chebychev inequality . . . 7
1.2.4 Concentration inequalities . . . 8
1.3 Differential calculus 8

Lecture notes.pdf

$$H_x^f = \begin{pmatrix} \frac{\partial^2 f}{\partial x_1^2}(x) & \frac{\partial^2 f}{\partial x_2 \partial x_1}(x) & \dots & \frac{\partial^2 f}{\partial x_d \partial x_1}(x) \\ \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots \\ \frac{\partial^2 f}{\partial x_1 \partial x_d}(x) & \frac{\partial^2 f}{\partial x_2 \partial x_d}(x) & \dots & \frac{\partial^2 f}{\partial x_d^2}(x) \end{pmatrix}$$

Remark. *By Swhartz theorem, if f is C^2, then the Hessian is symmetrical.*

1.3.3 *Lipshitz continuity*

Definition 10. L-Lipschitz continuous function
Let f be a differentiable function and $L > 0$. We say that f is L-Lipschitz continuous if $\forall x, y \in \mathbb{R}^d$,

$$||f(x) - f(y)|| \leqslant L||x - y||$$

Definition 11. L-Lipschitz continuous gradients
Let f be a differentiable function and $L > 0$. We say that f has L-Lipschitz continuous gradients if $\forall x, y \in \mathbb{R}^d$,

$$\|\nabla_x f - \nabla_y f\| \leqslant L||x - y||$$

1.3.4 *Smoothness*

Definition 12. Smoothness
A differentiable function f with real values is said L-smooth if and only if

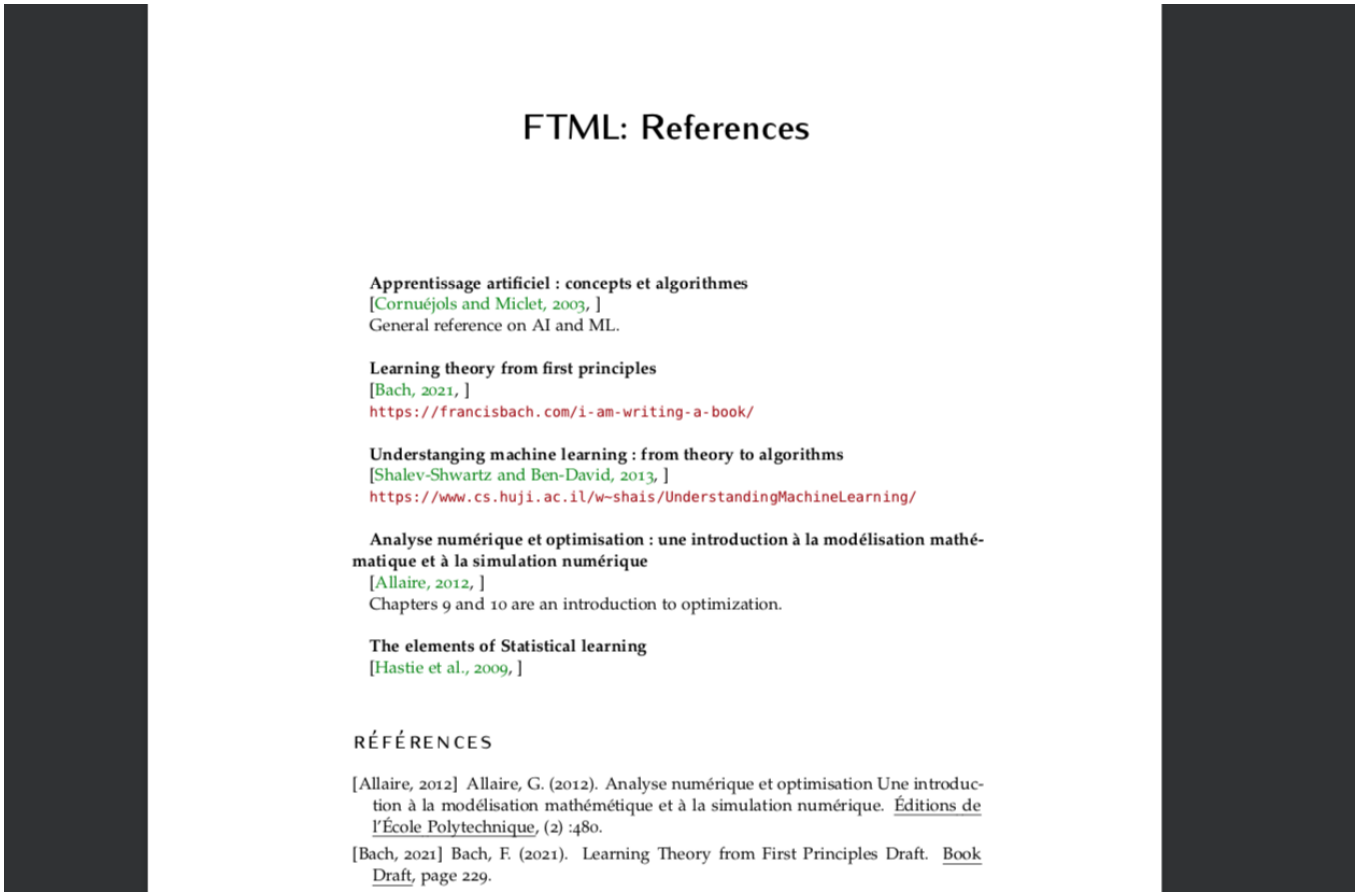
$$\forall x, y \in \mathbb{R}^d, |f(y) - f(x) - \nabla_x f(y - x)| \leqslant \frac{L}{2}||y - x||^2$$

Lemme 5. *f is L-smooth if and only if it has L-Lipshitz continuous gradients.*

Démonstration. We consider, for x and y fixed $\in \mathbb{R}^d$, the map :

$$u : \begin{cases} t \to f(x + t(y - x)) \\ \mathbb{R} \mapsto \mathbb{R} \end{cases}$$

References



FTML: References

Apprentissage artificiel : concepts et algorithmes
[Cornuéjols and Miclet, 2003,]
General reference on AI and ML.

Learning theory from first principles
[Bach, 2021,]
https://francisbach.com/i-am-writing-a-book/

Understanging machine learning : from theory to algorithms
[Shalev-Shwartz and Ben-David, 2013,]
https://www.cs.huji.ac.il/w~shais/UnderstandingMachineLearning/

Analyse numérique et optimisation : une introduction à la modélisation mathématique et à la simulation numérique
[Allaire, 2012,]
Chapters 9 and 10 are an introduction to optimization.

The elements of Statistical learning
[Hastie et al., 2009,]

RÉFÉRENCES

[Allaire, 2012] Allaire, G. (2012). Analyse numérique et optimisation Une introduction à la modélisation mathématique et à la simulation numérique. Éditions de l'École Polytechnique, (2) :480.

[Bach, 2021] Bach, F. (2021). Learning Theory from First Principles Draft. Book Draft, page 229.

Introduction

- What is IA ? What is ML ?
- Presentation of learning paradigms :
  - supervised learning
  - unsupervised learning
  - reinforcement learning
- Several examples of input spaces $\mathcal{X}$ and output spaces $\mathcal{Y}$.
- Presentation of the problem of overfitting and of the curse of dimensionality

Example I

Predict the winning team of an NBA game at half-time.

- Dataset : 15 years of games (comments, text) : approximately 17000 games.
- The dataset is preprocessed to have as an input a time-series : each time contains the score **and** 10 technical features (rebounds, etc.). So for each time the dimension is 11. Each game is a matrix of size 1440×11, reorganized as a line vector.
- Output : Receiving team wins or looses (classification)
- Evaluation metric : classification error ("0-1" loss).

Example II

Predict the quantity of oil in a rock.

- Input : tomographic image of a rock.
- Output : material of the rock, average presence of residual oil in the rock (regression).

Example III

Detect issues in wind farms.

- Input : sensors on the wind turbine (wind direction, air temperature, electric tension, rotation speed, component temperature, etc.) as a time series. Each step represents 10 minutes (several years).
- Output : Power generated by the turbine (regression)
- Evaluation metric : MAE (mean absolute error).

Maths

- linear algebra
- statistics / probabilities
- differentials
- optimization, convex functions

Most important results and tools are in lecture notes.pdf.

Supervised learning

- The dataset D_n is a collection of n samples $\{(x_i, y_i)\}_{1 \leq i \leq n}$, that are **independent and identically distributed** draws of a joint random variable (X, Y).
- the law of (X, Y) is unknown, we can note it ρ. We assume there exists an unknown function f that relates X and Y (not necessary deterministic).
- we look for an estimator $\tilde{f}_n$ of f. n refers to the fact that we have n samples.

Risks

Let l be a loss.
The **risk** (or **statistical risk, generalization error, test error)** of estimator f writes

$$E_{(X,Y)\sim\rho}[l(Y, f(X)]$$

The **empirical risk (ER)** of an estimator f writes

$$R_n(f) = \frac{1}{n}\sum_{i=1}^{n} l(y_i, f(x_i))$$

The risks depend on the loss l.

Excess risk

We define the **target function / bayes estimator** f^* by

$$f^* \in \underset{f:X\to Y}{\arg\min}\, R(f)$$

with $f : X \to Y$ set of measurable functions. Notation : $R(f^*) = R^*$.

Definition

Fundamental problem of Supervised Learning
Estimate f^* given only D_n and I.

$\tilde{f}_n$ is the minimizer of the empirical risk.

Definition

Excess risk

The **excess risk** $\mathcal{R}(\tilde{f}_n)$ measures how close $\tilde{f}_n$ is to the best possible f^*, in terms of expected risk (average / expected) error on new examples.

$$\mathcal{R}(\tilde{f}_n) = R(\tilde{f}_n) - R(f^*)$$

Bayes estimator

We saw that in two cases, it is possible to express f^* as a funciton of the law of probablities.

- Regression with square loss :

$$f^*(x) = E(Y|X = x) \tag{1}$$

- Classification with "0-1" loss :

$$f^*(x) = \arg\max_{z \in \mathcal{Y}} P(Y = z|X = x) \tag{2}$$

  This one is intuitive.

However, in practical situations we do not know the law $P(X, Y)$ of $P(Y|X = x)$ for all $x \in \mathcal{X}$.

Bayes estimator

We saw that in two cases, it is possible to express f^* as a funciton of the law of probablities.

- Regression with square loss :

$$f^*(x) = E(Y|X = x) \tag{3}$$

- Classification with "0-1" loss :

$$f^*(x) = \arg\max_{z \in \mathcal{Y}} P(Y = z|X = x) \tag{4}$$

- Regression with absolute loss ($l(y, z) = |y - z|$) ? FTML Project.

Risk decomposition into Bias and variance

- f^* : Bayes predictor
- F : Hypothesis space
- $\tilde{f}_n$: estimated predictor ($\in F$).

$$R(\tilde{f}_n) - R^* = \left(R(\tilde{f}_n) - \inf_{f \in F} R(f)\right) + \left(\inf_{f \in F} R(f) - R^*\right) \quad (5)$$

However : $\tilde{f}_n$ is a **random variable**, and so is $R(\tilde{f}_n)$. We can also consider the expected value of this quantity.

Risk decomposition

- approximation error
- esitmation error
- optimization eror

Ordinary least squares

- Linear regression, square loss, $\mathcal{X} = \mathbb{R}^d$, $\mathcal{Y} = \mathbb{R}$.
- Hypothesis space F

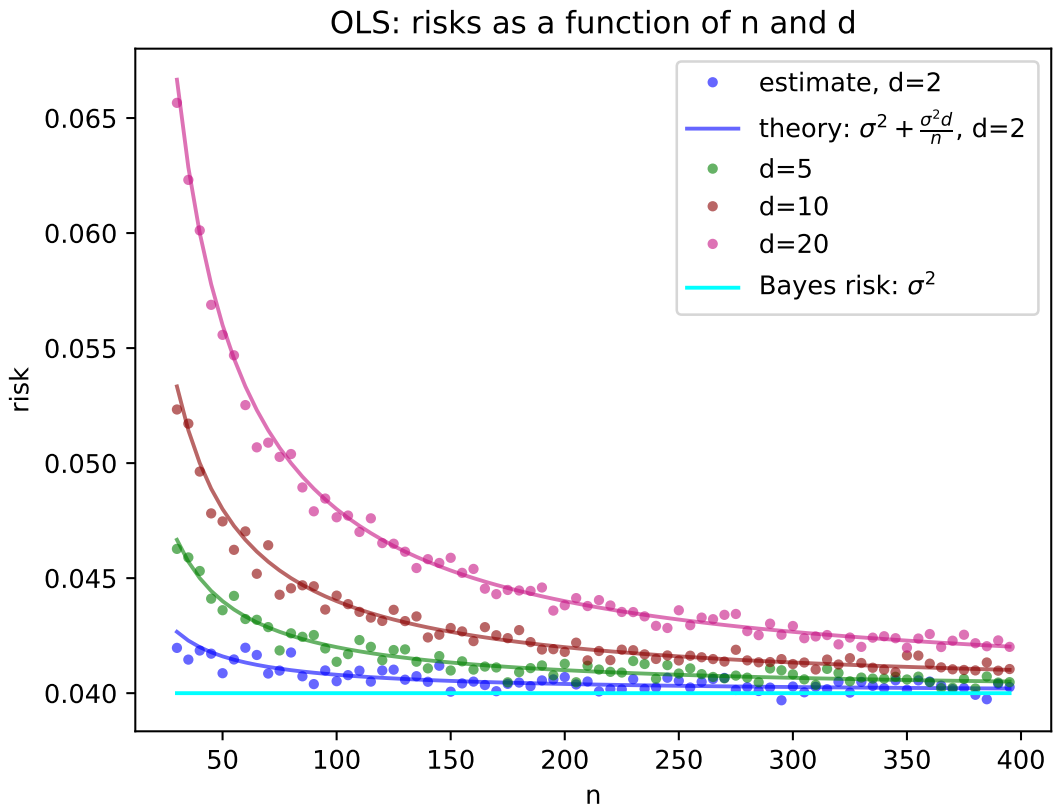
$$F = \{x \mapsto \theta^T x, \theta \in \mathbb{R}^d\}$$

- closed-form solution
- statistical properties of the OLS estimator in the fixed design setting. It is unbiased and

$$E\left[R(\hat{\theta})\right] - R(\theta^*) = \frac{\sigma^2 d}{n} \tag{6}$$

OLS, fixed design

$$E\left[R(\hat{\theta})\right] - R(\theta^*) = \frac{\sigma^2 d}{n} \tag{7}$$

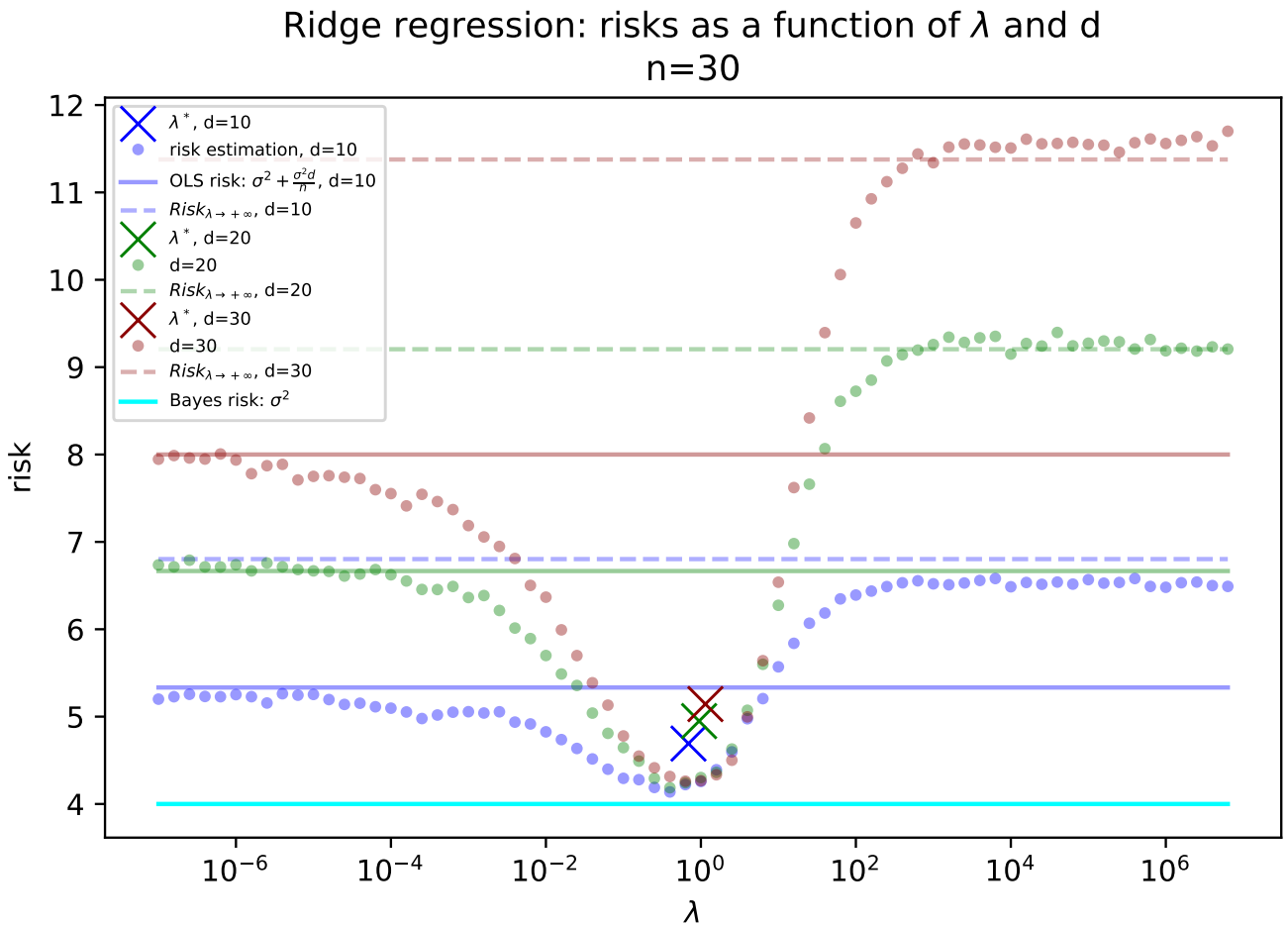


Ridge regression

- When d is large, we have statistical problems.
  - bad generalization error
  - high variance in the OLS estimator
  - no closed form solution if $d > n$.
- To tackle this problem, **regularization** is used. Ridge regression is the regularization with the square norm.
- closed-form solution, strong convexity, **but** a bias is introduced.

Ridge regression

With a good choice of λ (add $\lambda||\theta||^2$) to the empirical risk, we can obtain a better generalization error than OLS.



Convex surrogates for classification

- For classification, the empirical risk with binary loss is not differentiable nor convex.
- Instead of minimizing it, we use a proxy (convex surrogate), such as the logistic loss.
- We have some guarantees on the consistency of this process (calibration).
- logistic regression : differentiable, convex, computation of the gradient. Link with maximum likelihood estimation.

Unsupervised learning

From a number of samples x_i, you want to retrieve information on their structure : **modelisation**. The three main unsupervised learning problems are :

- clustering
- density estimation
- dimensionality reduction

Unsupervised learning : clustering

- Notion of partition
- vector quantization, kmeans
- Heuristics for clustering (choice of the number of clusters)

Unsupervised learning : dimensionality reduction

- Motivation : compress the data while keeping as much information as possible.
- **Principal component analysis** : linear dimensionality reduction. Link with the maximisation of a well defined random variable, and with the reconstruction error. Notion of explained variance in order to use a relevant number of principal components.

Empirical risk minimization

We want to minimize

$$g(\theta) = \frac{1}{n} \sum_{i=1}^{n} g_i(\theta) \tag{8}$$

on $\theta \in \mathbb{R}^d$.
With for instance

$$g_i(\theta) = l(x_i^T \theta, y_i) + \frac{\mu}{2} ||\theta||^2 \tag{9}$$

$x_i \in \mathbb{R}^d$, l being a loss, such as square loss or logistic loss.

ERM

We want to minimize

$$g(\theta) = \frac{1}{n} \sum_{i=1}^{n} g_i(\theta) \tag{10}$$

With for instance

$$g_i(\theta) = l(x_i^T \theta, y_i) + \frac{\mu}{2} ||\theta||^2 \tag{11}$$

l being a loss, such as square loss or logistic loss.
Often, no closed form solution is avaible, or is too expensive to compute it (large n, large d). Hence we use iterative methods. On convex problems, we have some convergence guarantees in some cases.

ERM

Gradient descent :

$$\theta^{t+1} = \theta^t - \gamma \nabla_\theta g(\theta) \tag{12}$$

What convergence guarantees do we have ?

ERM

Gradient descent :

$$\theta^{t+1} = \theta^t - \gamma \nabla_\theta g(\theta) \tag{13}$$

We assume there exists a minimizer θ^*.

- convex g : $g(\theta^t) - g(\theta^*)$ decreases as $\mathcal{O}(\frac{1}{t})$
- strongly-convex g : $g(\theta^t) - g(\theta^*)$ decreases as $\mathcal{O}\big(\exp(-\frac{t}{\kappa})\big)$

ERM

Gradient descent (GD) :

$$\theta^{t+1} = \theta^t - \gamma \nabla_\theta g(\theta) \tag{14}$$

We assume there exists a minimizer θ^*.

- convex g : $g(\theta^t) - g(\theta^*)$ decreases as $\mathcal{O}(\frac{1}{t})$
- strongly-convex g : $g(\theta^t) - g(\theta^*)$ decreases as $\mathcal{O}\big(\exp(-\frac{t}{\kappa})\big)$

However, each GD iteration costs $\mathcal{O}(nd)$ operations. If n is very large, and/or d, this can become prohibitive.

SGD

$$g(\theta) = \frac{1}{n} \sum_{i=1}^{n} g_i(\theta) \tag{15}$$

SGD : replace $\nabla_\theta g(\theta)$ by $\nabla_\theta g_i(\theta)$, for a i random in $[1, n]$.

$$\theta^{t+1} = \theta^t - \gamma \nabla_\theta g_i(\theta) \tag{16}$$

$\nabla_\theta g_i(\theta)$ is an **unbiased estimation** of the full gradient $\nabla_\theta g(\theta)$.
Given θ,

$$E_i\left[\nabla_\theta g_i(\theta)\right] = \nabla_\theta g(\theta) \tag{17}$$

Each uptate is $\mathcal{O}(d)$ (independent on n).

SAG

SAG : different estimation g^t_{SAG} of the full gradient $\nabla_\theta g(\theta)$.

$$g^t_{SAG} = \frac{1}{n}\sum_{i=1}^{n} y_i^k \tag{18}$$

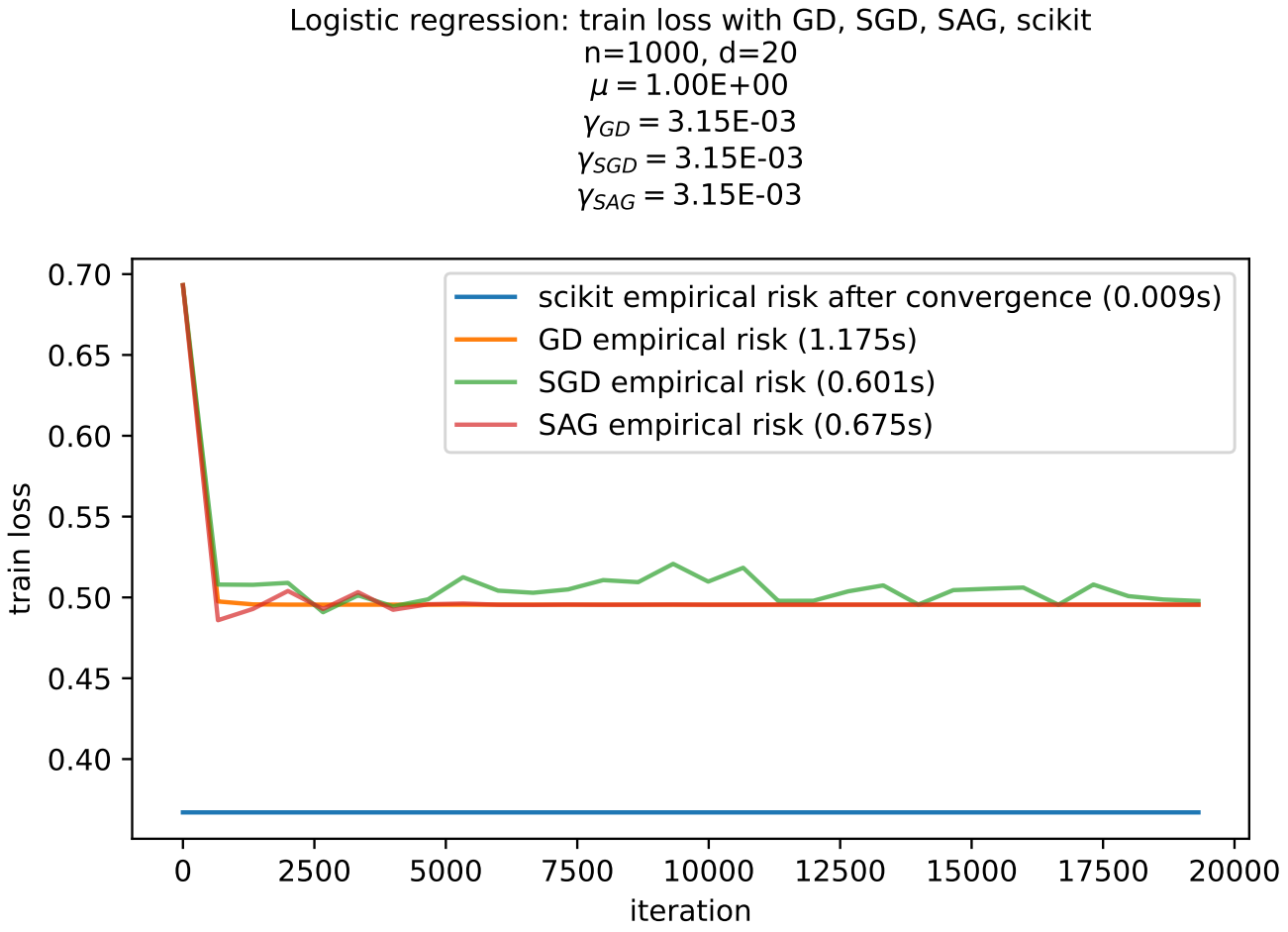
where $y_i^k = \nabla_\theta g_i(\theta^{t_i})$, and θ^{t_i} is the most recent iterate when we sampled i.

SAG

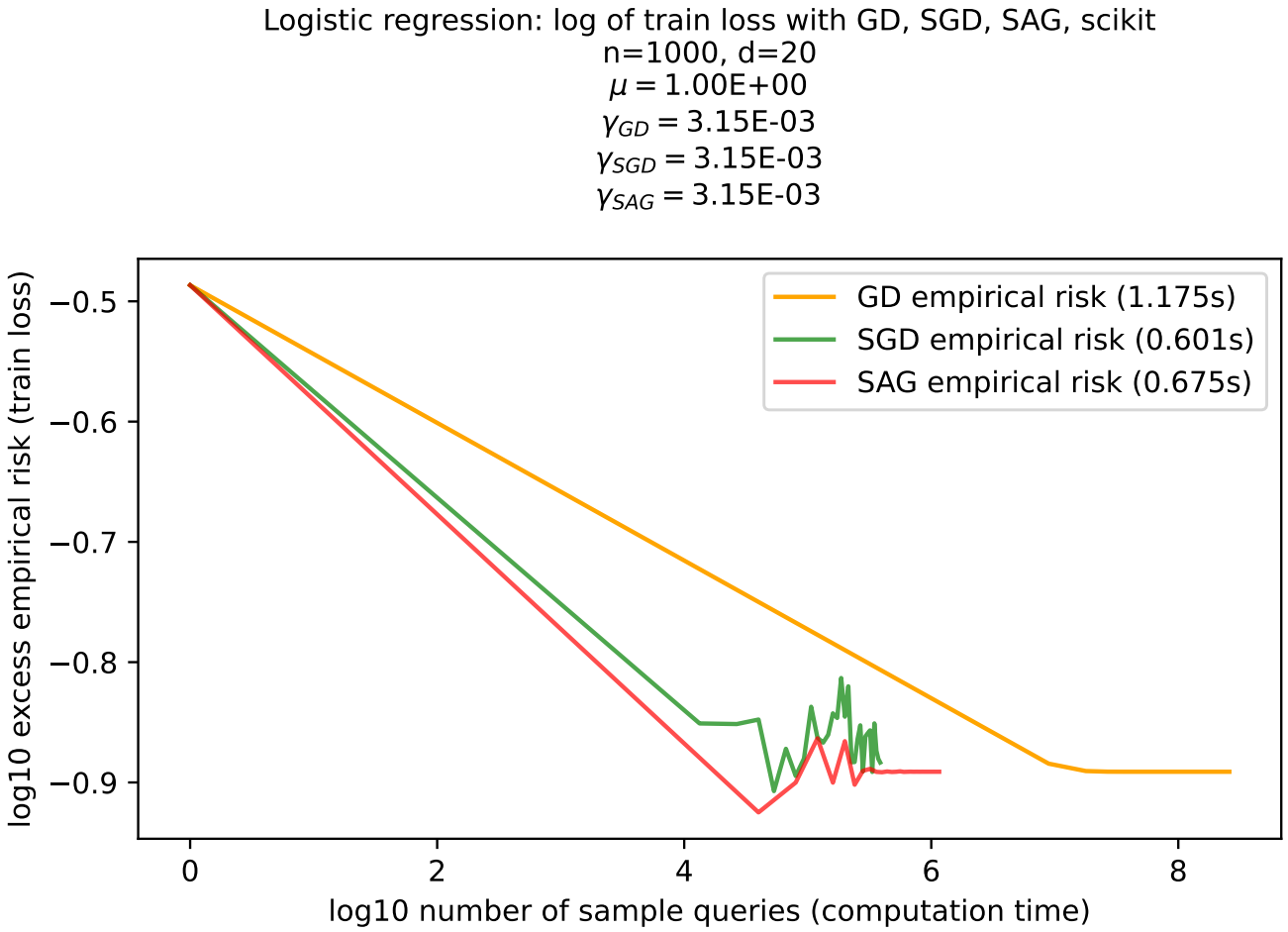
The paper [Schmidt et al., 2013,] shows that under some conditions, SAG combines the advantages of GD and of SGD :

- convergence rate in terms of algorithm iterations
- cost of one iteration

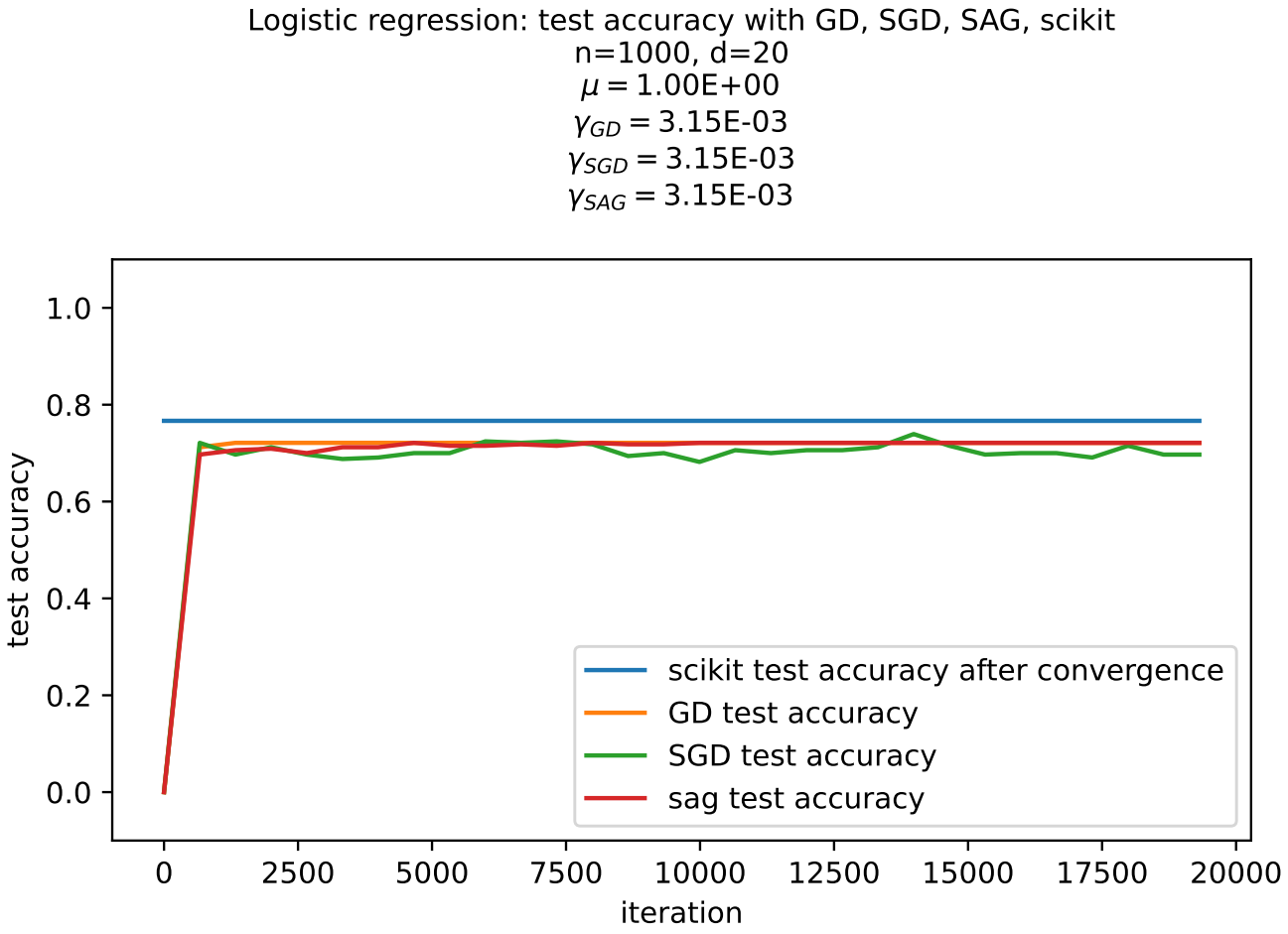
Basic implementation



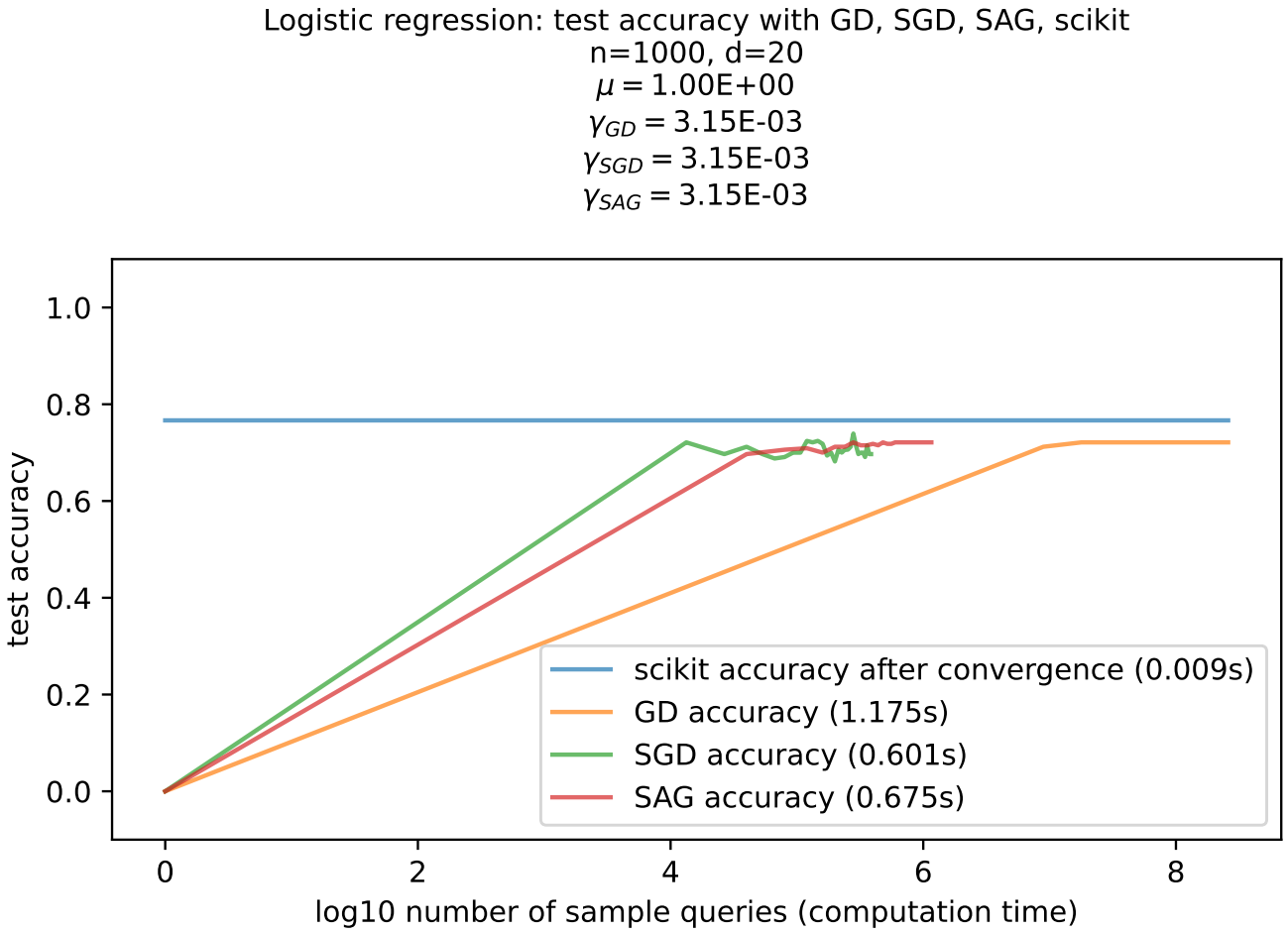
Basic implementation



Basic implementation



Basic implementation



Better implementation

- Larger step size (e.g. $\frac{1}{L}$ instead of $\frac{1}{16L}$, although the theoretical result is only proven for the latter).
- Line search
- non uniform sample selection
- **many** other details : you can read entire sections of the paper.

Broadcasting

https://numpy.org/doc/stable/user/basics.broadcasting.html

Other variants

https://www.datasciencecentral.com/an-overview-of-gradient-descent-optimization-algorithms/

Course summary

SAG (TP5)

Scoring, multiclass problems, cross validation
- Regression
- Binary classification
- Multi-class classification
- Cross-validation

Neural networks
- A space of functions
- Chain rule

Scoring

Many possibilities are available to evaluate the quality of an estimator.

Regression

- $\mathcal{X} = \mathbb{R}^d$
- $\mathcal{Y} = \mathbb{R}$.

Until now we used the squared error or the mean squared error (MSE) :

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_{pred,i} - y_{label,i})^2 \tag{19}$$

Without normalisation, it is also called **residual sum of squares** (RSS)

$$RSS = \sum_{i=1}^{n}(y_{pred,i} - y_{label,i})^2 \tag{20}$$

Coefficient of determination

Also called $R2$. $R2 \leq 1$. We introduce the Total sum of squares (TSS)

$$TSS = \sum_{i=1}^{n} (y_{label,i} - \bar{y})^2 \tag{21}$$

where $\bar{y}$ is the mean of the obsered data

$$\bar{y} = \frac{1}{n} \sum_{i=1}^{n} y_{label,i} \tag{22}$$

Then

$$R2 = 1 - \frac{RSS}{TSS} \tag{23}$$

Coefficient of determination

$$TSS = \sum_{i=1}^{n} (y_{label,i} - \bar{y})^2 \tag{24}$$

$$RSS = \sum_{i=1}^{n} (y_{pred,i} - y_{label,i})^2 \tag{25}$$

Finally we define the Explained sum of squares ESS :

$$ESS = \sum_{i=1}^{n} (y_{pred,i} - \bar{y})^2 \tag{26}$$

Then if the predicitons are linear, then

$$TSS = ESS + RSS \tag{27}$$

Scikit metrics

https://scikit-learn.org/stable/modules/model_evaluation.html

Binary classification

We now review some metrics for binary classification problems.

Accuracy

Most simple scoring : accuracy.

$$\frac{\text{number of correct predictions}}{\text{number of samples}} \tag{28}$$

Precision and recall

Precision : "Quand tu dis que c'est positif, c'est positif".

$$\frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}} \tag{29}$$

Recall / sensitivity (rappel) : "Quand c'est positif, tu dis que c'est positif".

$$\frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}} \tag{30}$$

See also : specificity and 1-specificity.

F score

Also called $F1$. It quantifies the tradeoff between precision and recall.

$$F1 = 2 \times \frac{\text{sensitivity} \times \text{precision}}{\text{sensitivity} + \text{precision}} \tag{31}$$

Exercice 1 : What are the extremal values of F ?

`https://en.wikipedia.org/wiki/F-score`

Faux amis

In french, accuracy and precision are both translated by "précision".

Binary classification

Standard classifiers such as logistic regression and support vector machines are **binary**.
However, sometimes $|\mathcal{Y}| > 2$.

Number of classifiers

We assume $|\mathcal{Y}| = p$.

Exercice 2 : How many classifiers need to be built :

- with the one-vs-rest scheme ?
- with the one-vs-one scheme ?

Number of classifiers

- one-vs-rest is the standard approach.
- one-vs-one need to build a number of classifiers that is quadratic in p, ($\mathcal{O}(p^2)$).
- However one-vs-one might still be useful since less samples are used by each binary classifiers, since we only need the samples from the two selected classes. If the complexity of the classifier scales badly with the umber of samples n (like for kernel methods), one-vs-one might be faster.

Scikit

Scikit has a builtin implementation of both schemes.
`https://scikit-learn.org/stable/modules/generated/sklearn.multiclass.OneVsRestClassifier.html`

Softmax

It is also possible to directly learn p real outputs and convert the vector of outputs to a probability by normalizing (it is what is done with softmax regression in neural networks).
`https://fr.wikipedia.org/wiki/Fonction_softmax`

Confusion matrix

https://en.wikipedia.org/wiki/Confusion_matrix
https://scikit-learn.org/stable/modules/generated/sklearn.metrics.confusion_matrix.html

Classification report

It is also possible to define precision, recall (and thus F1) for each class. In scikit, classification report prints these quantities for each class
`https://scikit-learn.org/stable/modules/generated/sklearn.metrics.classification_report.html`

Multi-class vs multi-label

Don't mix multi-class problems and multi-label problems.

- multi-class : several output classes are possible
- mutli-label : we have to make several predictions for each input

A problem can be both multi-class and multi-label.

Hyperparameters

All learning algorithms have hyperparameters. Examples :

- regularization parameter
- learning rate schedule
- kernel widths
- tree depth for cart
- number of trees for random forest

Hyperparameter tuning

Sometimes, we have theoretical results that guarantee that a hyperparameter value is a good choice (e.g. the learning rates for GD, SGD, SAG).

However :

- often, these parameters values depend on constants that are problem-dependent and sometimes not available :
  - variance σ^2
  - smoothness constant L
- we may not have a theoretical result at all (true for some aspects of deep learning)
- some values of the hyperparameter might work **better** than the theoretical value.

Hyperparameter tuning

Conclusion : it is most often necessary to experiment and test in order to find relevant hyperparameters.

Validation set

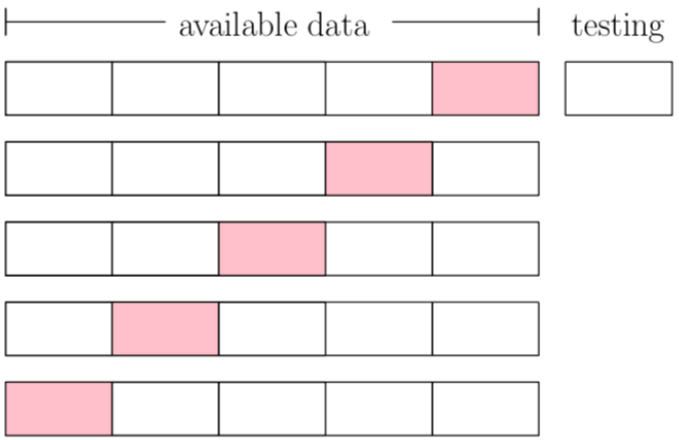
Common practice is to split test several hyperparameters, which means training several models.
The dataset can be split into 3 parts :

- train set : used to optimize each model
- validation set : used to compute a validation error of each model (error on this dataset). The model with lowest validation error can be chosen.
- test set : used to test the final model. We can not use it for validation (choice of the hyperparameters) : otherwise this choice would

Problem : there might be a high variability in the validation procedure. The found hyperparameters might depend a lot on the initial choice of the validation set.

Cross-validation

Cross-validation validation is another method that allows the use of more training data. The train set is split in k folds (often 5 of 10), and k validation errors are computed (one for each fold). The model with the lowest average validation error is chosen, **and then** trained on the whole train set.



Cross-validation

Due the the higher number of computations, cross-validation might be slower than standard train/validation/test split.

Scikit

Grid search is a method for testing hyper parameters (exhaustive search among a fiven list of values).
Grid search and cross validation are builtin in scikit.
`https://scikit-learn.org/stable/modules/cross_validation.html`
`https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html`

Learning curves

https://scikit-learn.org/stable/auto_examples/model_selection/plot_learning_curve.html

Many model selection methods exist :

https://scikit-learn.org/stable/model_selection.html

Course summary

SAG (TP5)

Scoring, multiclass problems, cross validation
- Regression
- Binary classification
- Multi-class classification
- Cross-validation

Neural networks
- A space of functions
- Chain rule

Neural networks

References / tools :

- https://www.deeplearningbook.org/
- https://d2l.ai/
- https://mlelarge.github.io/dataflowr-web/dldiy_ens.html
- https://playground.tensorflow.org/
- http://www.jzliu.net/blog/simple-python-library-visualize-neural-network/

Learning representations / features

- $\mathcal{X} = \mathbb{R}^d$.
- $\mathcal{Y} = \mathbb{R}$.

A neural network with scalar output learns a feature map $\phi : \mathbb{R}^d \to \mathbb{R}^m$ **and** a linear regressor or classifier, $\theta \in \mathbb{R}^m$.

$$\forall x, f(x) = \langle \theta, \phi(x) \rangle \tag{32}$$

We can add a bias by adding a dimension to θ and adding a component with a 1 to each $\phi(x)$.

Learning representations / features

- $\mathcal{X} = \mathbb{R}^d$.
- $\mathcal{Y} = \mathbb{R}$.

A neural network learns a feature map $\phi : \mathbb{R}^d \to \mathbb{R}^m$ **and** a linear regressor or classifier, $\theta \in \mathbb{R}^m$.

$$\forall x, f(x) = \langle \theta, \phi(x) \rangle \tag{33}$$

We can add a bias by adding a dimension to θ and adding a component with a 1 to each $\phi(x)$.
Remark : kernel methods use hardcoded features ϕ, that can be **implicit** (kernel trick).

Multi output

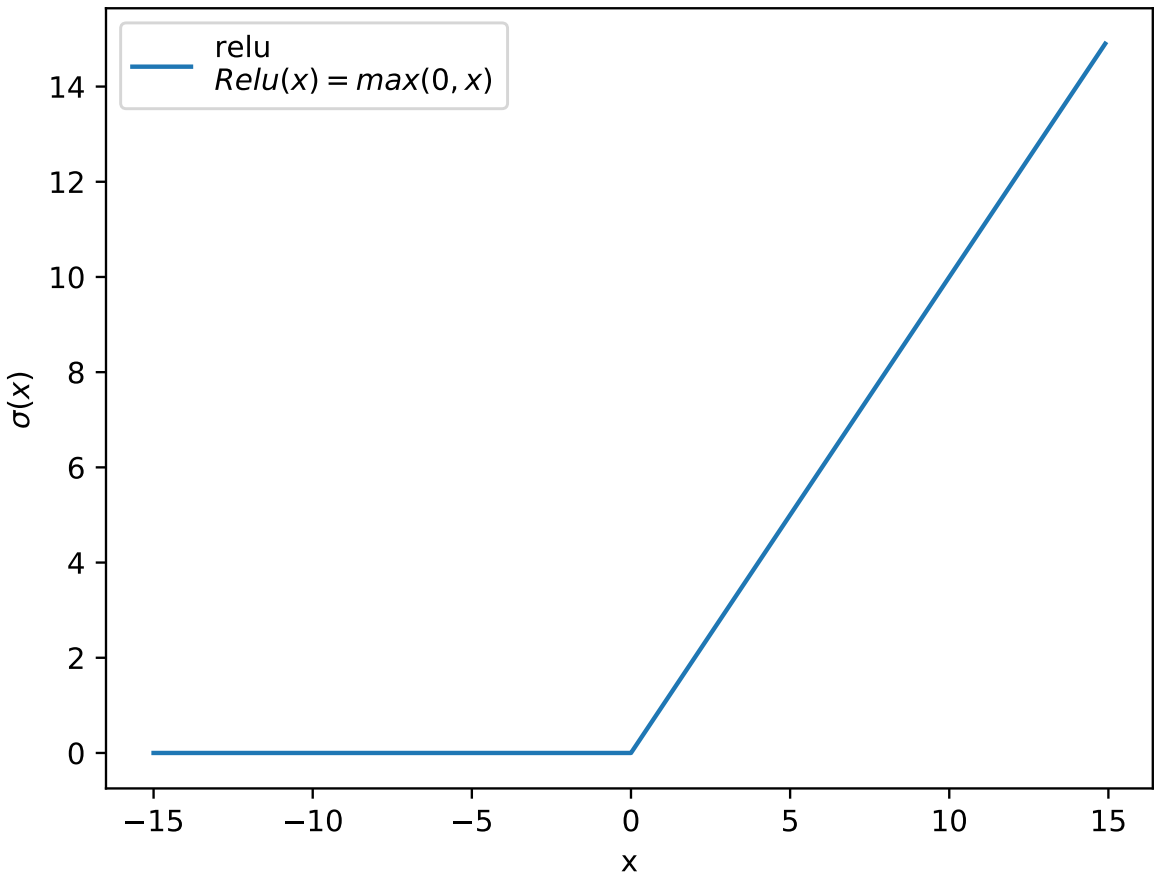
In order to learn a multidimensional output of dimension p ($\mathcal{Y} = \mathbb{R}^p$) , it is sufficient to lean a matrix $\theta \in \mathbb{R}^{m,p}$.

Single layer neural network

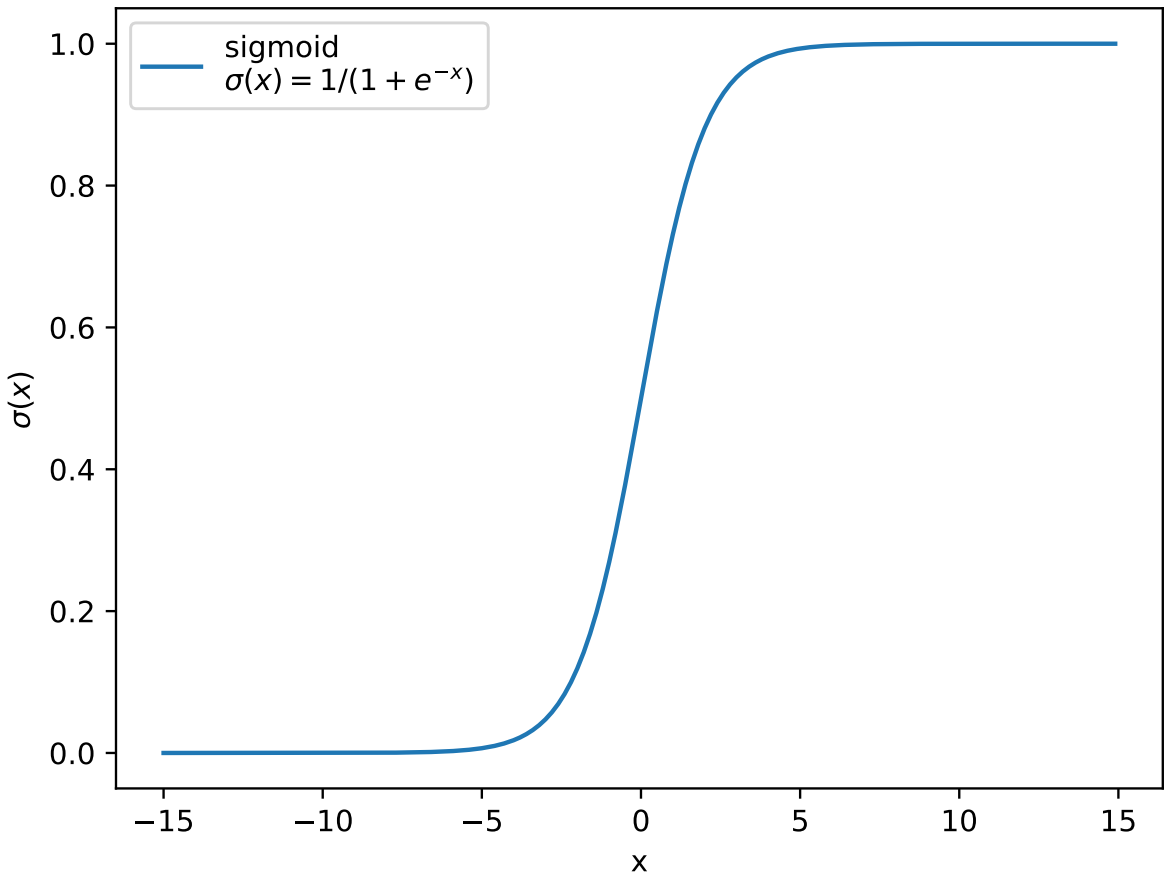
$$f(x) = \sum_{j=1}^{m} \theta_j \sigma(w_j^T x + b_j) \tag{34}$$

σ is an activation function (sigmoid, tanh, ReLu, etc).

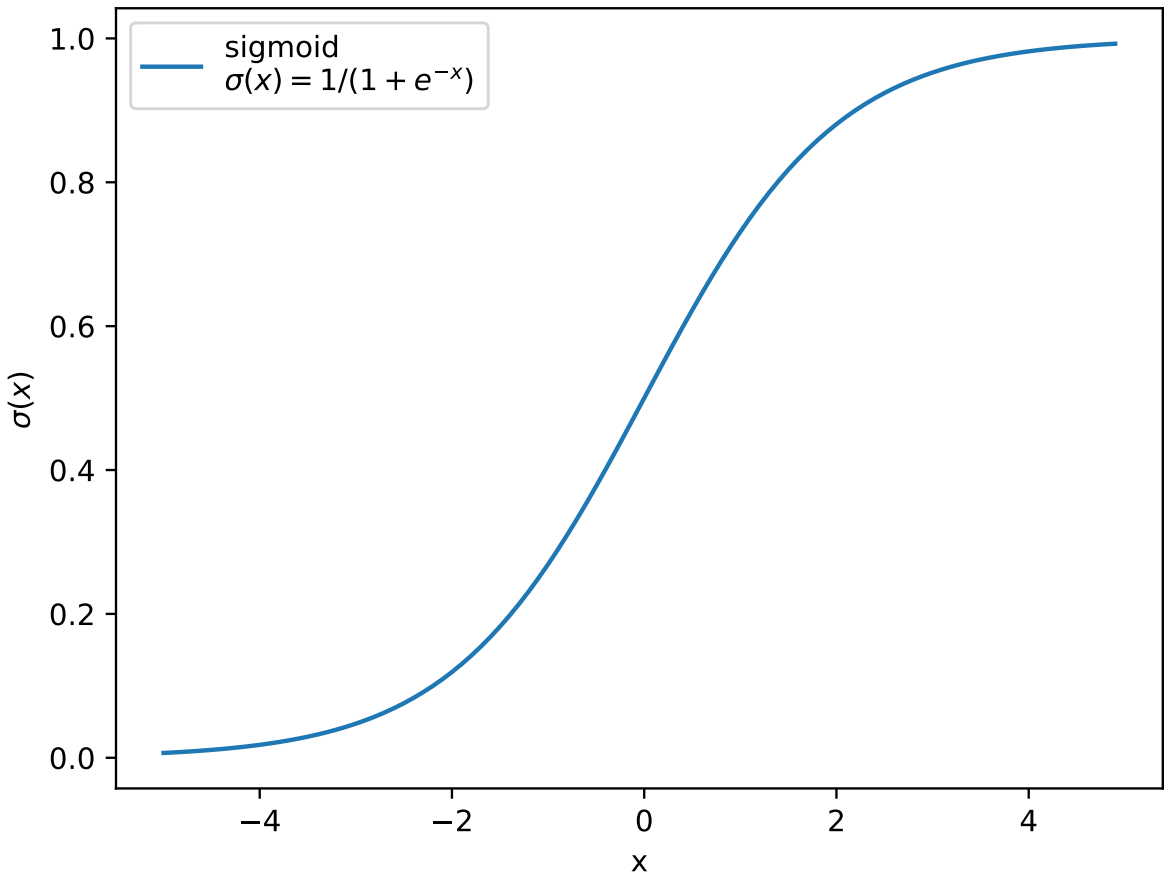
ReLU



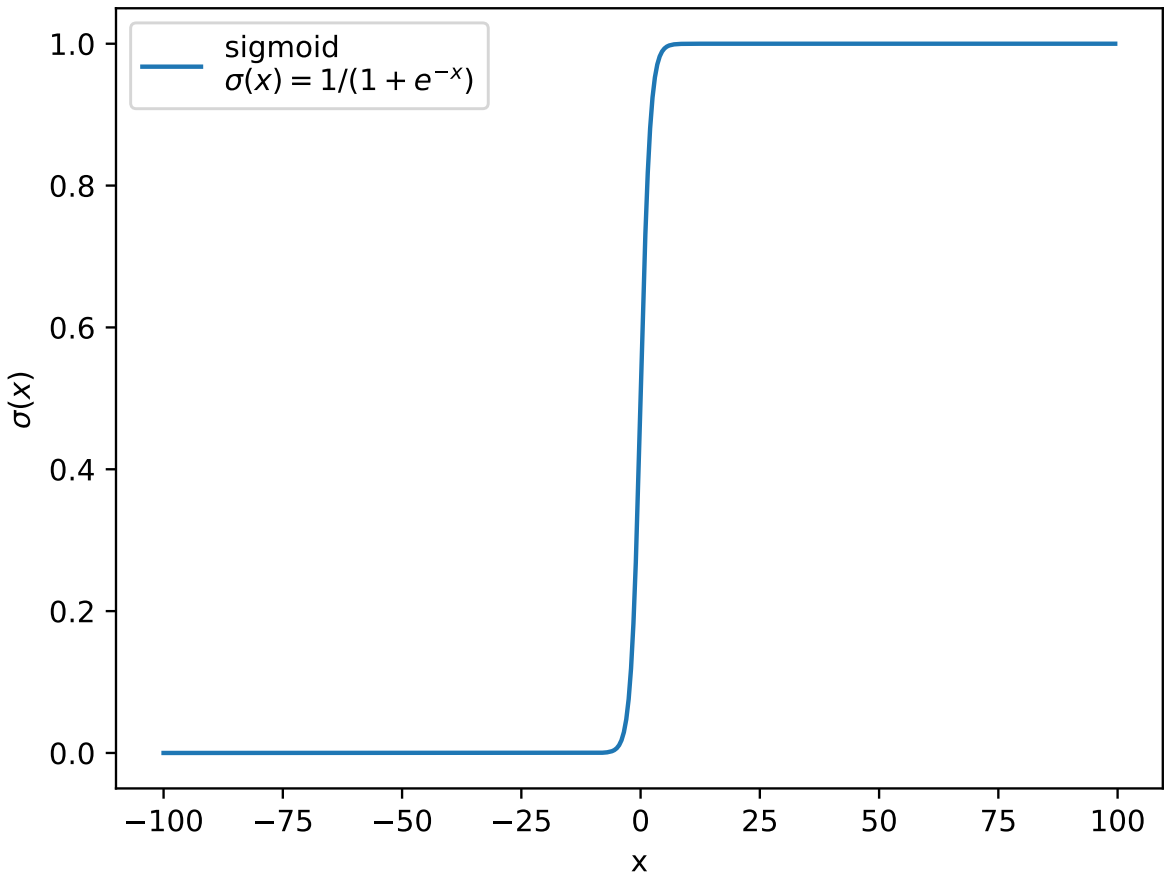
Sigmoid



Sigmoid



Sigmoid


sigmoid
$\sigma(x) = 1/(1 + e^{-x})$
1.0
0.8
0.6
0.4
0.2
0.0
$\sigma(x)$
−100
−75
−50
−25
0
25
50
75
100
x

Chain rule

Backpropagation is the general optimization framework for neural networks. SGD and variants are used.
To understand backpropagation, the main tool is the **chain rule.**

Chain rule

- Simple neural network : $d = 1$, $m = 1$, $n = 1$.
- $w \in \mathbb{R}$, $\theta \in \mathbb{R}$, b.

$$f(x) = \theta\sigma(wx + b) \tag{35}$$

Loss function

$$L(\theta, w, b) = (f(x) - y)^2 \tag{36}$$

We can note $\hat{y} = f(x)$ and $h = \sigma(wx + b)$, then $L(\theta, w, b) = (\hat{y} - y)^2$.

Exercice 3 : Compute and express in terms of y, $\hat{y}$ and h :

- $\frac{\partial h}{\partial w}$, $\frac{\partial h}{\partial b}$
- $\frac{\partial L}{\partial \theta}$, $\frac{\partial L}{\partial w}$, $\frac{\partial L}{\partial b}$

Chain rule

$$\frac{\partial h}{\partial w} = \sigma'(wx + b)x \tag{37}$$

$$\frac{\partial h}{\partial b} = \sigma'(wx + b) \tag{38}$$

$$\frac{\partial L}{\partial \theta} = 2(\hat{y} - y)\sigma(wx + b) \tag{39}$$

$$\frac{\partial L}{\partial w} = 2(\hat{y} - y)\theta\sigma'(wx + b)x \tag{40}$$

$$\frac{\partial L}{\partial b} = 2(\hat{y} - y)\theta\sigma'(wx + b) \tag{41}$$

Chain rule

$$\frac{\partial h}{\partial w} = \sigma'(wx + b)x \tag{42}$$

$$\frac{\partial h}{\partial b} = \sigma'(wx + b) \tag{43}$$

$$\frac{\partial L}{\partial \theta} = 2(\hat{y} - y)\sigma(wx + b) \tag{44}$$

$$\frac{\partial L}{\partial w} = 2(\hat{y} - y)\theta\frac{\partial h}{\partial w} \tag{45}$$

$$\frac{\partial L}{\partial b} = 2(\hat{y} - y)\theta\frac{\partial h}{\partial b} \tag{46}$$

Chain rule

- Simple neural network : $d = 2$, $m = 2$, $n = 1$.
- $w \in \mathbb{R}^{2,2} \simeq \mathbb{R}^4$, $\theta \in \mathbb{R}^2$, $b \in \mathbb{R}^2$.

$$f(x) = \theta_1 \sigma(w_1^T x + b_1) + \theta_2 \sigma(w_2^T x + b_2) \tag{47}$$

Loss function

$$L(\theta, w, b) = (y - f(x))^2 \tag{48}$$

Exercice 4 : Compute :

- $\nabla_w f(w, \theta, b)$
- $\nabla_b f(w, \theta, b)$
- $\nabla_\theta f(w, \theta, b)$

References I

Schmidt, M., Le Roux, N., and Bach, F. (2013).
Minimizing finite sums with the stochastic average gradient.
Mathematical Programming, 162(1-2) :83–112.