Practical introduction to machine learning

Part 3 : Supervised learning

Rémi Flamary - CMAP, École Polytechnique

Master Data Science, Institut Polytechnique de Paris

October 11, 2023

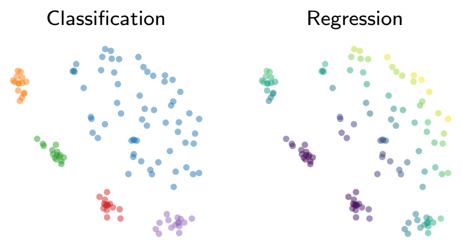
Overview of MAP654I

1. **Data and Machine Learning problems**
   - Data properties and visualization
   - Pre-processing
   - Finding your Machine Learning problem
2. **Unsupervised learning**
   - Clustering
   - Density estimation and generative modeling
   - Dictionary learning and collaborative filtering
   - Dimensionality reduction and manifold learning
3. **Supervised learning**
   - Bayesian decision and Nearest neighbors
   - Linear models nonlinear methods for regression and classification
   - Trees, forest and ensemble methods
4. **Validation and interpretation**
   - Performance measures
   - Models and parameter selection (validation)
   - Interpretation of the methods

Overview for the current part

Supervised dataset

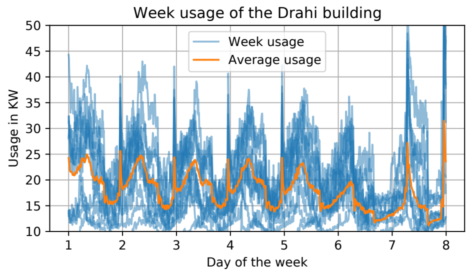
$$\mathbf{X} = \begin{bmatrix} \mathbf{x}_1^\top \\ \mathbf{x}_2^\top \\ \vdots \\ \mathbf{x}_i^\top \\ \vdots \\ \mathbf{x}_n^\top \end{bmatrix}, \quad \mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_i \\ \vdots \\ y_n \end{bmatrix}$$



Supervised learning

- The dataset contains the samples $\{\mathbf{x}_i, y_i\}_{i=1}^n$ where $\mathbf{x}_i$ is the feature sample and $y_i \in \mathcal{Y}$ its label.
- The values to predict (label) can be concatenated in a vector $\mathbf{y} \in \mathcal{Y}^n$
- Prediction space $\mathcal{Y}$ can be:
  - $\mathcal{Y} = \{-1, 1\}$ or $\mathcal{Y} = \{1, \ldots, p\}$ for classification problems.
  - $\mathcal{Y} = \mathbb{R}$ for regression problems ($\mathbb{R}^p$ for multi-output regression).
  - Structured for structured prediction (graphs,...).
- Scatter plots for supervised data (`plt.scatter`) use color for the label.

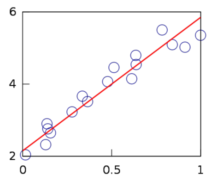
Example of real life dataset

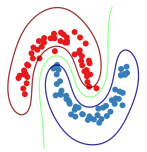


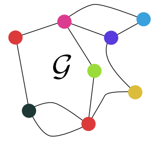
Electrical usage of the Drahi X-Novation Center

- Demonstrator of Energy4Climate of IP Paris.
- Recording of the electrical usage of the building during 1.5 years.
- Each day of energy usage contains 144 measurements.
- Supervised learning problem from the measurements of the last two days ($d = 288$) predict:
  - If the energy usage will lower of increase on the next day (classification)
  - The energy usage for the next day (regression).

Supervised learning







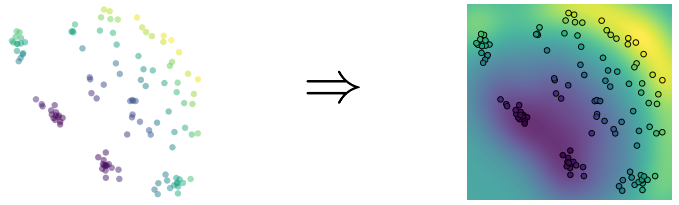
Objective

- Training dataset : $\{\mathbf{x}_i, y_i\}_{i=1}^n$ with observations $\mathbf{x}_i \in \mathbb{R}^d$ and labels $y_i \in \mathcal{Y}$.
- Train a function $f(\cdot) : \mathbb{R}^d \to \mathcal{Y}$ on the dataset.

Types of supervised prediction

- **Classification** $f(\cdot)$ predicts a class (discrete output) either binary $\mathcal{Y} = \{-1, 1\}$ or multiclass $\mathcal{Y} = \{1, \dots, p\}$.
- **Regression** $f(\cdot)$ predicts a continuous value ($\mathcal{Y} = \mathbb{R}$) or several ($\mathcal{Y} = \mathbb{R}^p$).
- **Structured prediction** $f(\cdot)$ predicts a structured object (graph, tree, molecule) (not discussed here).

Regression



Objective

$$\{\mathbf{x}_i, y_i\}_{i=1}^n \quad \Rightarrow \quad f : \mathbb{R}^d \to \mathbb{R}$$

- Train a function $f(\mathbf{x}) = y \in \mathcal{Y}$ predicting a continuous value ($\mathcal{Y} = \mathbb{R}$).
- Can be extended to multi-value prediction ($\mathcal{Y} = \mathbb{R}^p$).

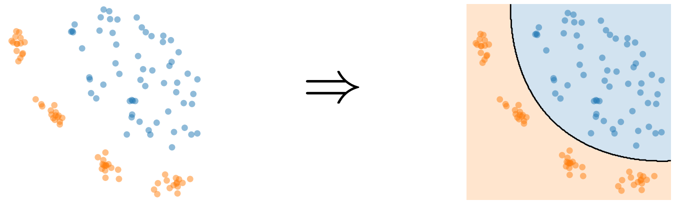
Parameters

- Type of function (linear, kernel, neural network).
- Performance measure.
- Regularization.

Methods

- Least Square (LS).
- Ridge regression, Lasso.
- Kernel regression.
- Deep learning.

Binary classification



Objective

$$\{\mathbf{x}_i, y_i\}_{i=1}^n \quad \Rightarrow \quad f : \mathbb{R}^d \to \{-1, 1\}$$

- Train a function $f(\mathbf{x}) = y \in \mathcal{Y}$ predicting a binary value ($\mathcal{Y} = \{-1, 1\}$).
- In practice, train a continuous function $f : \mathbb{R}^d \to \mathbb{R}$ and predict with sign(f).
- $f(\mathbf{x}) = 0$ defines the boundary on the partition of the feature space.
- Optional: provide uncertainty information such as probabilities of each class.

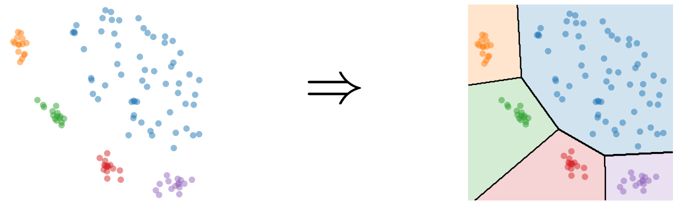
Parameters

- Type of function (linear, kernel, neural network).
- Performance measure.
- Regularization.

Methods

- Bayesian classifier (LDA, QDA)
- Linear and kernel discrimination
- Decision trees, random forests.
- Deep learning.

Multiclass classification



Objective

$$\{\mathbf{x}_i, y_i\}_{i=1}^n \quad \Rightarrow \quad f : \mathbb{R}^d \rightarrow \{1, \ldots, p\}$$

- Train a function $f(\mathbf{x}) = y \in \mathcal{Y}$ predicting an integer value ($\mathcal{Y} = \{1, \ldots, p\}$).
- In practice p continuous score functions f_k are estimated and the prediction is

$$f(\mathbf{x}) = \arg\max_k f_k(\mathbf{x}) \qquad (1)$$

- Softmax can be used instead of argmax to get probability estimates.

Parameters

- Type of function (linear, kernel, neural network).
- Performance measure.
- Regularization.

Methods

- Bayesian classifier (LDA, QDA)
- Linear and kernel discrimination
- Decision trees, random forests.
- Deep learning.

Scikit-learn estimator for supervised learning

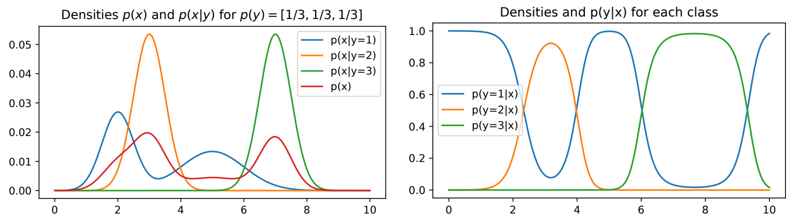
Scikit-learn object API

- Scikit-learn and its API became in recent years a standard for ML in Python.
- The estimator is usually used in 2 steps:
  1. Creation of the estimator :
     ```
     est = Estimator(param='parameter value',param2=10)
     ```
  2. Fitting of the estimator to the data:
     ```
     est.fit(X,y)
     ```
- After the fitting step, new attributes from the algorithms have been added to the object.

Using the estimator in supervised learning

- **Prediction**
  Predict the labels (for regression and classification) with `est.predict(X)` or `est.fit_predict(X)`
- **Probability prediction**
  On some classification methods the probability of belonging to the classes is computed with `est.predict_proba(X)` (`predict_log_proba` also available).
- **Decision functions**
  On some classification methods the score of belonging to the classes is computed by `est.decision_function(X)`.

Probability distribution of the data



Probability distributions for classification problem

We suppose here that the data is generated following a joint feature/label distribution.

- $p(\mathbf{x}, y)$ is the joint feature/label probability.
- $p(\mathbf{x}) = \int p(\mathbf{x}, y)dy$ is the feature probability (marginal on the feature)
- $p(y) = \int p(\mathbf{x}, y)d\mathbf{x}$ is the discrete label probability (marginal on the labels)
- $p(\mathbf{x}|y) = \frac{p(\mathbf{x},y)}{p(y)}$ is the conditional probability of $\mathbf{x}$ for a given class.
- $p(y|\mathbf{x}) = \frac{p(\mathbf{x},y)}{p(\mathbf{x})}$ is the conditional probability of y for a given observation $\mathbf{x}$.

Bayes Theorem : $p(x, y) = p(x|y)p(y) = p(y|x)p(x)$

Probability of error and Bayes risk

Error rate of a classifier

$$P_{err}(f) = E_{\mathbf{x}}\left[\sum_{k=1}^{p}(1_{f(\mathbf{x})\neq k})p(y = k|\mathbf{x})\right] = 1 - \int \sum_{k=1}^{p} 1_{f(\mathbf{x})=k}p(y = k|\mathbf{x})p(\mathbf{x})d\mathbf{x} \quad (2)$$

- 1_{cond} has value 1 when the condition $cond$ is true else 0.
- For a given classifier f the error rate is the probability that the classifier makes a mistake.
- Standard measure of performance for a classifier, often estimated empirically and called accuracy (`sklearn.metrics.accuracy_score`).

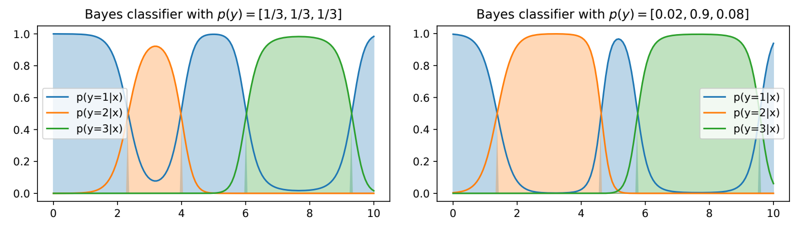
Bayes Risk

A more general expression of the error of a classifier is the Bayes risk expressed as

$$R(f) = E_{(\mathbf{x},y)}[L(y, f(\mathbf{x}))] \qquad (3)$$

- $L(i, j)$ is the cost of predicting class j when the true class is i.
- When $L(i, j) = 1_{i\neq j}$ we recover the error rate where all mistakes cost the same.
- The Bayes risk can be used to encode asymmetry between the errors of a classifier (some errors are more serious than others).

Bayesian decision



Bayes Classifier

- The Bayes classifier is the one minimizing the error rate

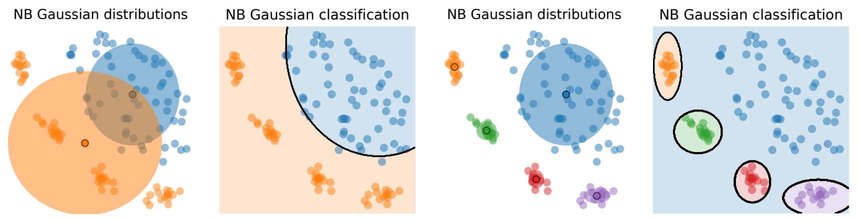
$$\min_f \quad 1 - \int \sum_{k=1}^{K} 1_{f(\mathbf{x})=k} p(y=k|\mathbf{x}) p(\mathbf{x}) d\mathbf{x}$$

- We can see above that for a given $\mathbf{x}$ the $f(\mathbf{x})$ that minimize the error is the one with maximum probability $p(y|\mathbf{x}) = p(y)p(\mathbf{x}|y)/p(\mathbf{x})$ ($p(\mathbf{x})$ indep. from y).
- The Bayes classifier minimizing the problem above is then

$$f^\star(\mathbf{x}) = \arg\max_k \quad p(y=k|\mathbf{x}) \tag{4}$$

- This is exactly the multiclass classifier formula (1) with $f_k(\mathbf{x}) = p(y=k|\mathbf{x})$.
- In practice the probability distributions are unknown so they have to be estimated.

Naive Bayes Classifier (NB)



Principle (Tutorial [Murphy et al., 2006])

- Assumption in NB Classification is that all variables are independent:

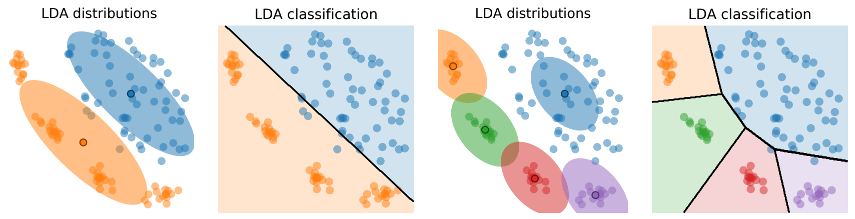
$$p(y|\mathbf{x}) = p(y) \prod_{i=1}^{d} p(x_i|y)$$

- Probabilities $\hat{p}(x_i|y)$ are estimated independently in 1D for each variable x_i with distributions depending on data prior (Gaussian, Bernoulli, Multinomial).
- Simple model that works very well on many applications [Zhang, 2004].
- Used a lot on textual data with bag of words (binary data for many spam filters)

Gaussian Naive Bayes (sklearn.naive_bayes.GaussianNB)

- Classes follow Gaussian distributions with diagonal covariances (indep. variables).
- The data is modeled as a GMM illustrated above for 2 and 5 classes.

Linear Discriminant Analysis (LDA)



Principle [Fisher, 1936]

- Model the conditional probabilities for each class as

$$p(\mathbf{x}|y=k) = \mathcal{N}(\boldsymbol{\mu}_k, \boldsymbol{\Sigma})$$

- The covariance matrix $\boldsymbol{\Sigma}$ is shared across all classes (Homoscedasticity).
- The proportions of classes are $\phi_k = p(y=k) \geq 0$ such that $\sum_k \phi_k = 1$.
- The Bayes decision function is taken as

$$f_k(\mathbf{x}) = \log(p(\mathbf{x}|y=k)p(y=k)).$$

- LDA is also known as Fisher Discriminant Analysis (FDA).
- Scikit-learn : sklearn.discriminant_analysis.LinearDiscriminantAnalysis

LDA discriminant functions

Score functions and simplifications

- The score function $f_k(\mathbf{x})$ can be expressed as

$$\begin{aligned} f_k(\mathbf{x}) &= \log(\phi_k) + \log(p(\mathbf{x}|y=k)) \\ &= \log(\phi_k) - \frac{d}{2}\log(2\pi) - \frac{1}{2}\log\det(\boldsymbol{\Sigma}) - \frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_k)^T\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu}_k) \end{aligned}$$

- Removing the terms that do not depend on k and do not change the decision we get the following equivalent score function

$$f_k(\mathbf{x}) = \boldsymbol{\mu}_k^\top\boldsymbol{\Sigma}^{-1}\mathbf{x} + \log(\phi_k) - \frac{1}{2}\boldsymbol{\mu}_k^\top\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_k = \mathbf{w}_k^\top\mathbf{x} + b_k \tag{5}$$

- The decision function is linear because the quadratic terms are constant wrt k when the Gaussians have the same covariance $\boldsymbol{\Sigma}$.

LDA for binary classification

- Parameters for the Gaussian distributions are: $\phi, \boldsymbol{\Sigma}, \boldsymbol{\mu}_1, \boldsymbol{\mu}_{-1}$,
- Decision function f can be expressed as : $f(\mathbf{x}) = sign(f_1(\mathbf{x}) - f_{-1}(\mathbf{x}))$
- It can be expressed as $f(\mathbf{x}) = sign(\mathbf{w}^\top\mathbf{x} + b) = sign(\sum_k w_k x_k + b)$ with

$$\mathbf{w} = \boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_{-1}), \qquad b = -\frac{1}{2}\mathbf{w}^\top(\boldsymbol{\mu}_1 + \boldsymbol{\mu}_{-1}) + \log\phi_1 - \log\phi_{-1}$$

LDA in practice

LDA as dimensionality reduction [Rao, 1948]

- We want to find a subspace that maximizes the distance between the means of the classes in the projected space while minimizing the variance of each class.
- The optimization problem can be expressed as

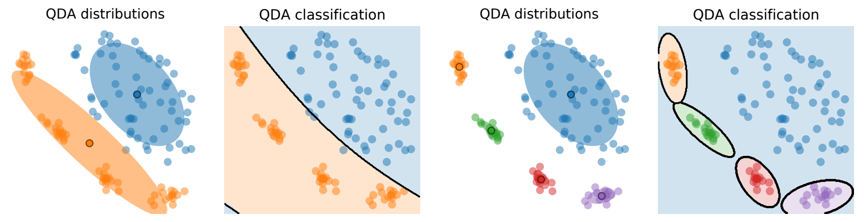
$$\max_{\mathbf{D},\mathbf{D}^\top\mathbf{D}=\mathbf{I}_{K-1}} \frac{\langle \mathbf{\Sigma}_b, \mathbf{D}\mathbf{D}^\top \rangle}{\langle \mathbf{\Sigma}, \mathbf{D}\mathbf{D}^\top \rangle}$$

  where $\mathbf{\Sigma}_b = \sum_k \phi_k(\boldsymbol{\mu}_k - \bar{\boldsymbol{\mu}})(\boldsymbol{\mu}_k - \bar{\boldsymbol{\mu}})^\top$ with $\bar{\boldsymbol{\mu}} = \sum_k \phi_k \boldsymbol{\mu}_k$.
- The solution $\mathbf{D}^\star$ contains the eigenvector with largest eigenvalues for the generalized eigen-decomposition of $\mathbf{\Sigma}^{-1}\mathbf{\Sigma}_b$.

Estimating the parameters

- Gaussian distributions for each class can be estimated by their empirical mean $\hat{\boldsymbol{\mu}}$ and covariance $\hat{\mathbf{\Sigma}}$ estimators.
- In high dimension good estimators for covariances require a large number of samples but still might lead do degenerate matrices (with numerical problems).
- In this case a good strategy is to perform a shrinkage of the matrix toward the identity by using $(1-\alpha)\hat{\mathbf{\Sigma}} + \alpha\mathbf{I}_d$ instead of $\hat{\mathbf{\Sigma}}$.

Quadratic Discriminant Analysis (QDA)



Principle (Tutorial [Tharwat, 2016])

- Bayesian decision similar to LDA but where the conditional probabilities are :

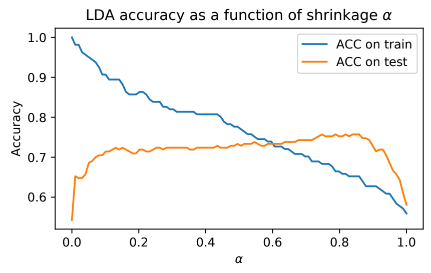
$$p(\mathbf{x}|y=k) = \mathcal{N}(\boldsymbol{\mu}_k, \mathbf{\Sigma}_k)$$

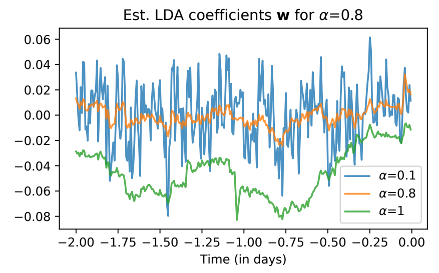
- The score functions $f_k(\mathbf{x})$ can be expressed as

$$f_k(\mathbf{x}) = \log(\phi_k) - \frac{d}{2}\log(2\pi) - \frac{1}{2}\log\det(\mathbf{\Sigma}_k) - \frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_k)^{\mathrm{T}}\mathbf{\Sigma}_k{}^{-1}(\mathbf{x}-\boldsymbol{\mu}_k)$$

- When the covariances $\mathbf{\Sigma}_k$ are different the quadratic terms do not cancel each other and the final decision is quadratic.
- More sensitive to the curse of dimensionality than LDA.
- Scikit-learn : sklearn.discriminant_analysis.QuadraticDiscriminantAnalysis

Bayesian decision on energy usage dataset

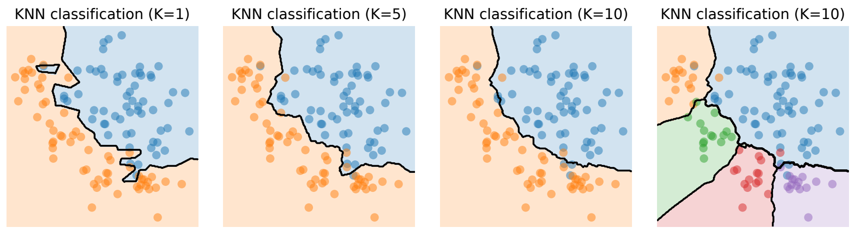




Application to energy usage classification

- Objective : predict if the energy usage will increase on the next day using the previous two days $(n=161, d=288)$.
- Gaussian Naive Bayes classifier provided an accuracy on test data of : 0.59
- LDA with no shrinkage of the covariance gives an accuracy of : 0.54
- LDA with a shrinkage of $\alpha=0.8$ gives an accuracy of : 0.75
- QDA with no shrinkage gives an accuracy of : 0.48
- QDA with a shrinkage of $\alpha=0.8$ gives an accuracy of : 0.74
- Warning : in high dimension probability density estimation is hard, regularize/shrink your covariances.

K-nearest neighbors classification (KNN)



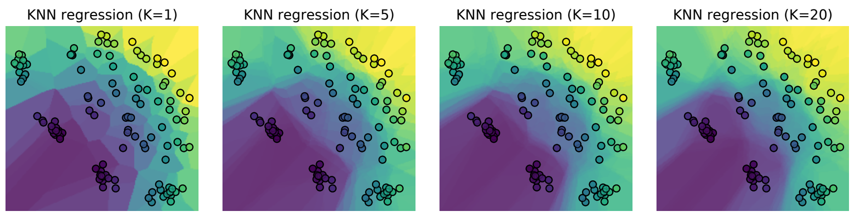
Principle [Fix and Hodges, 1989]

- Estimate locally the conditional densities $\hat{p}(\mathbf{x}|y=k)$ as

$$\hat{p}(\mathbf{x}|y=k) = \frac{1}{K}\sum_{i\in\mathcal{N}^K(\mathbf{x})} 1_{y_i=k} \tag{6}$$

  where $\mathcal{N}^K(\mathbf{x})$ contains the index of the K nearest samples to $\mathbf{x}$ in the dataset.
- The density estimation is a special case of KDE with adaptive kernel bandwidth.
- Instead of uniform voting one can use : $\hat{p}(\mathbf{x}|y=k) = \frac{\sum_{i\in\mathcal{N}^K(\mathbf{x})} \frac{1}{\|\mathbf{x}-\mathbf{x}_i\|} 1_{y_1=k}}{\sum_{i\in\mathcal{N}^K(\mathbf{x})} \frac{1}{\|\mathbf{x}-\mathbf{x}_i\|}}$
- Consistent estimator but requires the whole dataset for prediction (complexity).
- Scikit-learn implementation : sklearn.neighbors.KNeighborsClassifier

K-nearest neighbors for regression



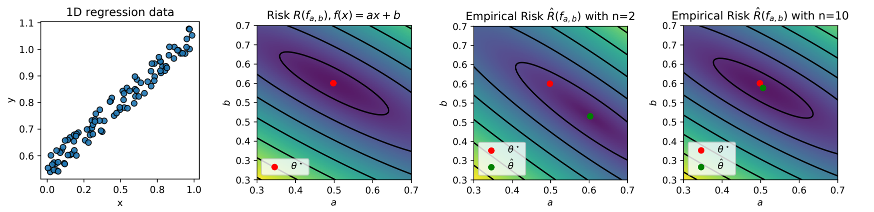
Principle

- The predicted value for a given samples $\mathbf{x}$ an be computed as:

$$\hat{f}(\mathbf{x}) = \frac{1}{K}\sum_{i\in\mathcal{N}^K(\mathbf{x})} y_i \tag{7}$$

- This is the expected value of y on the distribution in the neighborhood $\mathcal{N}^K$
- For $K=1$ the partition of the space is a Voronoi Diagram with prediction piecewise constant in each cell (for regression and classification).
- Smoother prediction using kernel or distance-based weighting similar to KNN classification with $\hat{f}(\mathbf{x}) = \frac{\sum_{i\in\mathcal{N}^K(\mathbf{x})} k(\mathbf{x}_i,\mathbf{x})y_i}{\sum_{i\in\mathcal{N}^K(\mathbf{x})} k(\mathbf{x}_i,\mathbf{x})}$.
- Scikit-learn implementation : sklearn.neighbors.KNeighborsRegressor

Empirical Risk Minimization



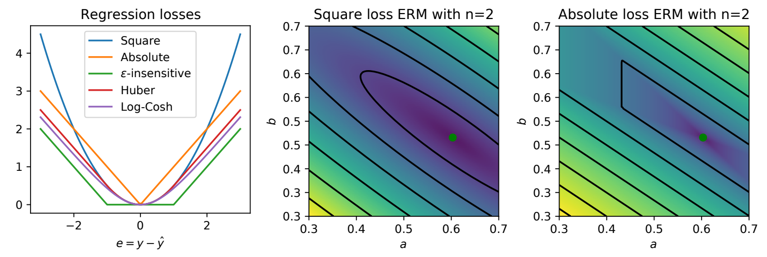
Principle

- In practice the Bayes risk is not known, we only have access to a sampling $\{\mathbf{x}_i, y_i\}_i$ of the true distribution.
- We search for a prediction function f that minimize the expected loss over the empirical distribution (training data):

$$\min_f \quad \left\{\hat{R}(f) = \frac{1}{n}\sum_{i=1}^{n} L(y_i, f(\mathbf{x}_i))\right\} \tag{8}$$

- L is a measure of discrepancy between the true and predicted values.
- The empirical risk $\hat{R}(f)$ is a good approximation of $R(f)$ for large n.
- Usually we use a parametric function $f_{\boldsymbol{\theta}}$ and optimize its parameters $\boldsymbol{\theta}$.

Losses for regression

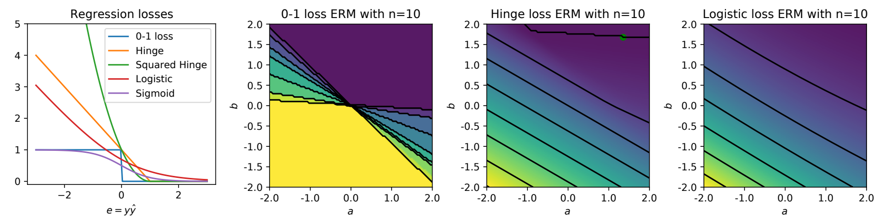


Penalizing prediction error

- For regression the error can be defined as : $e = y - f(\mathbf{x}) = y - \hat{y}$
- Typical losses :

Loss	$L(y,\hat{y})$	Smooth	Convex
Square loss (MSE, L2)	$\frac{1}{2}(y-\hat{y})^2$	$++$	$++$
Absolute deviation (MAE)	$\|y-\hat{y}\|$	-	$+$
ϵ-insensitive	$\max(0, \|y-\hat{y}\| - \epsilon)$	-	$+$
Huber loss	$\begin{cases}\frac{1}{2}(y-\hat{y})^2 & \text{for } \|y-\hat{y}\| \leq \delta, \\ \delta(\|y-\hat{y}\| - \frac{1}{2}\delta), & \text{otherwise.}\end{cases}$	$+$	$++$
Log-Cosh	$\log(\cosh(y-\hat{y}))$	$++$	$++$

Losses for classification



Penalizing prediction error

- For binary classification ($\{-1, 1\}$) the error can be defined using : $e = yf(\mathbf{x}) = y\hat{y}$
- Typical losses are asymmetric wrt 0 :

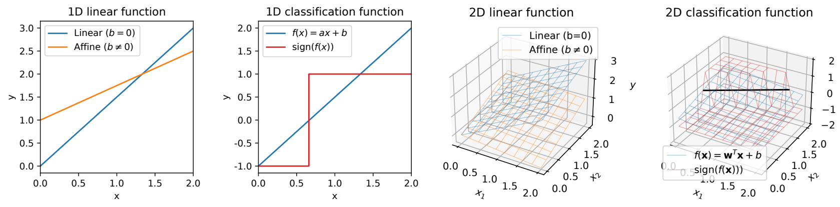
Loss	$L(y,\hat{y})$	Smooth	Convex
0-1 loss	$\frac{1}{2}(1 - \text{sign}(y\hat{y}))$	$-$	$-$
Hinge	$\max(0, 1 - y\hat{y})$	-	$+$
Squared Hinge	$\max(0, 1 - y\hat{y})^2$	$+$	$+$
Logistic	$\log(1 + \exp(-y\hat{y}))$	$+$	$+$
Sigmoid	$(1 - \tanh(y\hat{y}))/2$	$+$	-

- For multiclass classification the classical loss is the categorical cross-entropy :

$$L(y, f(\mathbf{x})) = -\sum_{k=1}^{p} \delta_{k=y}\log(f_k(\mathbf{x}))$$

where the output of f contains probability estimates (softmax).

Linear prediction model

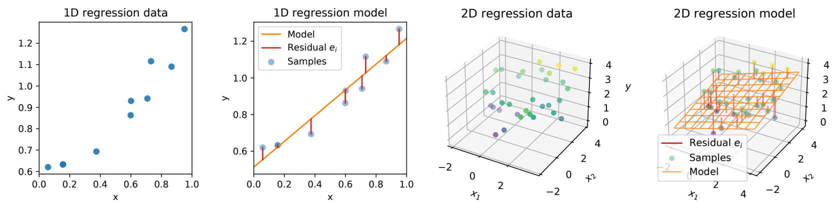


Linear (affine) function

$$f_{\boldsymbol\theta}(\mathbf{x})=\sum_{i=1}^{d}w_ix_i+b=\mathbf{x}^\top\mathbf{w}+b=[\mathbf{x}^\top,1]\boldsymbol\theta \qquad (9)$$

- $\mathbf{w}\in\mathbb{R}^d$ a vector defining an hyperplane in $\mathbb{R}^d$ ($\mathbf{w}$ orthogonal to the hyperplane).
- $b\in\mathbb{R}$ a bias term displacing the function along the normal $\mathbf{w}$ of the hyperplane.
- All the parameters can be stored in a unique vector $\boldsymbol\theta^\top=\left[\mathbf{w}^\top,b\right]$.
- Linear models are interpretable (look at the weights w_i and their sign).
- Estimating the bias b can be done using the data matrix $\tilde{\mathbf{X}}=[\mathbf{X},\mathbf{1}_n]$.
- Linear models from `sklearn.linear_model` have the following attributes after fitting
  - `model.coef_` : contains the weight coefficients $\mathbf{w}\in\mathbb{R}^d$ of the variables.
  - `model.intercept_` : contains the bias b (also called the intersect).

Least Square regression (LS)



Principle

$$\min_{\mathbf{w},b}\quad\frac{1}{n}\sum_{i=1}^{n}(y_i-\mathbf{w}^\top\mathbf{x}_i-b)^2 \qquad (10)$$

- Also called Ordinary Least Squares Linear Regression (OLS).
- Minimize the mean of the squared prediction errors $e_i=y_i-\mathbf{w}^\top\mathbf{x}_i-b$ (MSE).
- Matrix and linear reformulation:
$$\min_{\mathbf{w},b}\quad\frac{1}{n}\|\mathbf{y}-\mathbf{X}\mathbf{w}-b\mathbf{1}_n\|^2\quad\equiv\quad\min_{\boldsymbol\theta}\quad\frac{1}{n}\|\mathbf{y}-\tilde{\mathbf{X}}\boldsymbol\theta\|^2 \qquad (11)$$
where $\tilde{\mathbf{X}}=[\mathbf{X},\mathbf{1}_n]$ is the data matrix with a concatenated column of ones.
- Scikit-learn : `sklearn.linear_model.LinearRegression`

Solving the least square

Minimum of a convex function

Let $J(\boldsymbol\theta)$ be a smooth convex function $\mathbb{R}^{d+1}\to\mathbb{R}$ $\mathbb{R}$. $\boldsymbol\theta^\star$ is a minimum $J(\boldsymbol\theta)$ if and only if

$$\nabla J(\boldsymbol\theta^\star)=\mathbf{0} \qquad (12)$$

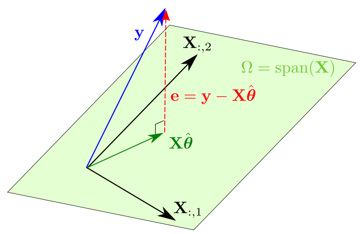
where $\nabla J(\boldsymbol\theta)\in\mathbb{R}^{d+1}$ is the gradient of the function $\nabla J(\boldsymbol\theta)_i=\frac{\partial J(\boldsymbol\theta)}{\partial\theta_i}\quad\forall i$

Gradient and solution for Least Square

- The objective function can be expressed as:
$$J(\boldsymbol\theta)=\frac{1}{n}\|\mathbf{y}-\tilde{\mathbf{X}}\boldsymbol\theta\|^2=\frac{1}{n}\left(\mathbf{y}^\top\mathbf{y}-2\boldsymbol\theta^\top\tilde{\mathbf{X}}^\top\mathbf{y}+\boldsymbol\theta^\top\tilde{\mathbf{X}}^\top\tilde{\mathbf{X}}\boldsymbol\theta\right)$$
- The gradient of the function is
$$\nabla J(\boldsymbol\theta)=\frac{2}{n}\left(-\tilde{\mathbf{X}}^\top\mathbf{y}+\tilde{\mathbf{X}}^\top\tilde{\mathbf{X}}\boldsymbol\theta\right)$$
- Least Square estimator recovered by setting $\nabla J(\boldsymbol\theta)=\mathbf{0}$
$$\tilde{\mathbf{X}}^\top\mathbf{y}=\tilde{\mathbf{X}}^\top\tilde{\mathbf{X}}\hat{\boldsymbol\theta}\qquad\rightarrow\qquad\hat{\boldsymbol\theta}=(\tilde{\mathbf{X}}^\top\tilde{\mathbf{X}})^{-1}\tilde{\mathbf{X}}^\top\mathbf{y} \qquad (13)$$
- Warning : this solution requires that $\tilde{\mathbf{X}}$ be of rank $d+1$ (at least $n\geq d+1$).

Geometric interpretation of Ordinary Least Square

- We search for a vector $\tilde{\mathbf{X}}\hat{\boldsymbol\theta}\in\mathbb{R}^n$ in the span $\Omega=\text{span}(\tilde{\mathbf{X}})$.
- Minimizing the norm of the error $\mathbf{e}=\mathbf{y}-\tilde{\mathbf{X}}\hat{\boldsymbol\theta}$ corresponds to finding the orthogonal projection on Ω.
- For an optimal solution $\hat{\boldsymbol\theta}$, $\mathbf{e}$ is orthogonal to any vector in Ω.


$\mathbf{y}$
$\mathbf{X}_{:,2}$
$\Omega=\text{span}(\mathbf{X})$
$\mathbf{e}=\mathbf{y}-\mathbf{X}\hat\theta$
$\mathbf{X}\hat\theta$
$\mathbf{X}_{:,1}$

- This means that the residual $\mathbf{e}=\mathbf{y}-\tilde{\mathbf{X}}\hat{\boldsymbol\theta}$ should be orthogonal to any of the columns in $\tilde{\mathbf{X}}$ which implies that
$$\tilde{\mathbf{X}}^\top(\mathbf{y}-\tilde{\mathbf{X}}\hat{\boldsymbol\theta})=\mathbf{0}$$
- This orthogonality conditions allows to recover geometrically the solution
$$\hat{\boldsymbol\theta}=(\tilde{\mathbf{X}}^\top\tilde{\mathbf{X}})^{-1}\tilde{\mathbf{X}}^\top\mathbf{y}$$
that is the solution of the LS optimization problem (10).

Probabilistic interpretation of Least Square

Observation model and likelihood

- The model is supposed to be linear with Gaussian IID noise, that is

$$p(\mathbf{y}|\tilde{\mathbf{X}}) = \mathcal{N}(\tilde{\mathbf{X}}\boldsymbol{\theta}, \sigma^2\mathbf{I}_n)$$

- The log-likelihood for parameters $\boldsymbol{\theta}$ and σ^2 can be expressed as

$$\mathcal{L}(\boldsymbol{\theta}, \sigma^2) = -\frac{n}{2}\log(2\pi) - n\log(\sigma) - \frac{1}{2\sigma^2}\|\mathbf{y} - \tilde{\mathbf{X}}\boldsymbol{\theta}\|^2$$

Maximum likelihood

- Estimating the parameters $\boldsymbol{\theta}$ and σ^2 is done by maximum likelihood that is by solving

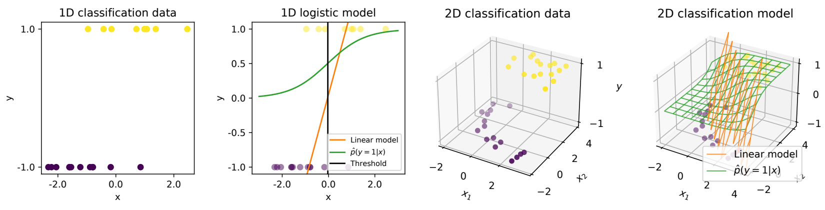
$$\max_{\boldsymbol{\theta}, \sigma^2} \quad \mathcal{L}(\boldsymbol{\theta}, \sigma^2)$$

- The solution is recovered by computing the gradients *w.r.t.* $\boldsymbol{\theta}$ and σ^2 and setting them to 0 (and checking that it is a maximum with the Hessian). The optimal values are :

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

$$\hat{\sigma}^2 = \frac{1}{n}\|\mathbf{y} - \tilde{\mathbf{X}}\boldsymbol{\theta}\|^2$$

Logistic regression



Principle

$$\min_{\mathbf{w},b} \quad \frac{1}{n}\sum_{i=1}^{n}\log(1+\exp(-y_i(\mathbf{w}^\top\mathbf{x}_i + b))) \tag{14}$$

- Model the conditional probabilities for binary classes $\{-1, 1\}$ with

$$p(y=1|\mathbf{x}) = \frac{1}{1+\exp(-\mathbf{w}^\top\mathbf{x} - b)}, \qquad p(y=-1|\mathbf{x}) = \frac{1}{1+\exp(\mathbf{w}^\top\mathbf{x} + b)} \tag{15}$$

- Bayes decision : $f(\mathbf{x}) = \text{sign}(p(y=1|\mathbf{x}) - p(y=-1|\mathbf{x}))$ that is equivalent to
$$f(\mathbf{x}) = \text{sign}(\mathbf{w}^\top\mathbf{x} + b)$$
- Parameters $\hat{\mathbf{w}}, \hat{b}$ are optimized by maximum likelihood corresponding to the optimization problem (14).
- Scikit-learn : `sklearn.linear_model.LogisticRegression`

Solving the logistic regression

Recovering the MLE optimization problem

- The log-likelihood from the logistic regression can be expressed as:

$$\mathcal{L}(\mathbf{w}, b) = \sum_{i, y_i=1} -\log(1+\exp(-\mathbf{w}^\top\mathbf{x}_i - b)) + \sum_{i,y_i=-1} -\log(1+\exp(\mathbf{w}^\top\mathbf{x}_i + b))$$
$$= -\sum_{i=1}^{n}\log(1+\exp(-y_i(\mathbf{w}^\top\mathbf{x}_i - b)))$$

- So maximizing the likelihood above is equivalent to minimizing its negative in Equation (14).

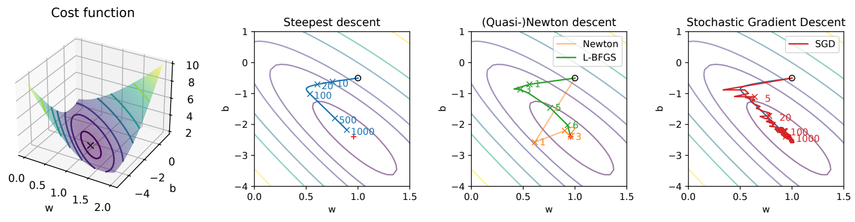
Gradient of the objective function

- The gradient of $J(\boldsymbol{\theta})$ defined in (14) can be expressed as

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = -\frac{1}{n}\tilde{\mathbf{X}}^\top\mathbf{P}\mathbf{y} \tag{16}$$

where $\mathbf{P}$ is a diagonal matrix of elements $\frac{p_i}{1+p_i}$ with $p_i = \exp(-y_i(\mathbf{w}^\top\mathbf{x}_i + b))$.
- Setting the gradients $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \mathbf{0}$ leads to a highly nonlinear equations so there is no close form solution as in LS.

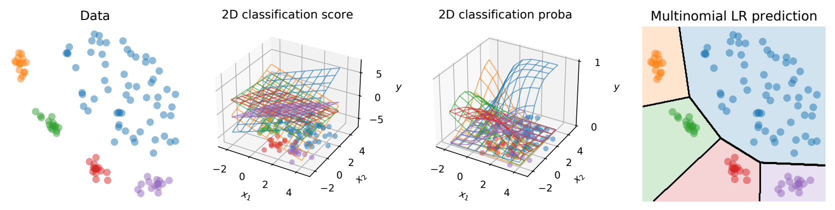
Numerical optimization with gradient descent



Principle

- Optimize a smooth function $J(\boldsymbol{\theta})$ using its gradient (or its approximation).
- Initialize a vector $\boldsymbol{\theta}^{(0)}$ and update it at each iteration k as:
$$\boldsymbol{\theta}^{(k+1)} = \boldsymbol{\theta}^{(k)} + \mu_k\mathbf{d}_k \tag{17}$$
where μ_k is a step and $\mathbf{d}_k$ is a descent direction ($\mathbf{d}_k^\top\nabla J(\boldsymbol{\theta}^{(k)}) < 0$).
- Classical descent directions are :
  - **Steepest descent** : $\mathbf{d}_k = -\nabla J(\boldsymbol{\theta}^{(k)})$
  - **Newton**: $\mathbf{d}_k = -(\nabla^2 J(\boldsymbol{\theta}^{(k)}))^{-1}\nabla J(\boldsymbol{\theta}^{(k)})$ where $\nabla^2 J(\boldsymbol{\theta}^{(k)})$ is the Hessian.
  - **Quasi-Newton (QN)** : $\mathbf{d}_k = -\mathbf{B}\nabla J(\boldsymbol{\theta}^{(k)})$ where $\mathbf{B}$ is an approximation of the inverse of the Hessian.
  - **Stochastic Gradient Descent (SGD)** : $\mathbf{d}_k = -\nabla\tilde{J}(\boldsymbol{\theta}^{(k)})$ with approx. gradient.

Multinomial logistic regression



Principle

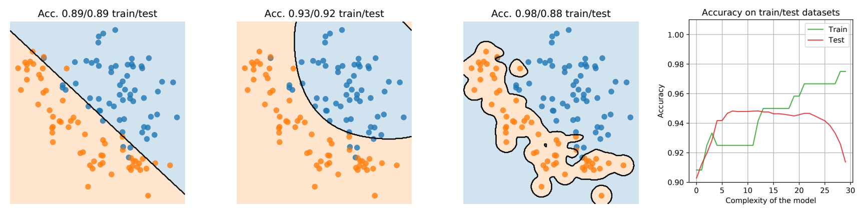
$$\min_{\mathbf{W},\mathbf{b}} \quad -\frac{1}{n}\sum_{i=1}^{n}\sum_{k=1}^{p}\delta_{y_i=k}\log(p_{\mathbf{W},\mathbf{b}}(y=k|\mathbf{x}_i)) \tag{18}$$

- MLE of parameters where $\mathbf{W}=[\mathbf{w}_1,\dots,\mathbf{w}_p]$ and $\mathbf{b}=[b_1,\dots,b_p]^\top$ the linear parameters and the conditional probabilities are modeled as

$$p_{\mathbf{W},\mathbf{b}}(y=k|\mathbf{x})=\frac{\exp(\mathbf{w}_k^\top\mathbf{x}+b_k)}{\sum_{j=1}^{p}\exp(\mathbf{w}_j^\top\mathbf{x}+b_j)} \tag{19}$$

- The operator above is called the softmax of predictions $\mathbf{w}_k^\top\mathbf{x}+b_k$ per classes.
- Problem (18) is and ERM where the loss function is the Kullback-Leibler between the one-hot encoding of the labels and the softmax output.
- Scikit-learn : `sklearn.linear_model.LogisticRegression` (with multiclass labels)

Regularization for supervised learning



Empirical risk minimization with regularization

$$\min_{f} \quad \sum_{i=1}^{n} L(y_i,f(\mathbf{x}_i))+\lambda\Omega(f) \tag{20}$$

- $L(\cdots)$ a loss function measure prediction performance on the training samples.
- $\Omega(\cdot)$ is a measure of complexity of the function weighted by $\lambda\geq 0$.
- For a given λ, (20) is an upper bound on the true expected risk.
- In practice the regularization is often applied on the parameters $\boldsymbol{\theta}$ of the function $f_{\boldsymbol{\theta}}$ leading to the following optimization problem

$$\min_{\boldsymbol{\theta}} \quad \sum_{i=1}^{n} L(y_i,f_{\boldsymbol{\theta}}(\mathbf{x}_i))+\lambda\Omega(\boldsymbol{\theta}) \tag{21}$$

Regularizing linear models

Complexity of a linear model

$$f_{\boldsymbol{\theta}}(\mathbf{x})=\sum_{i=1}^{d}w_ix_i+b=\mathbf{x}^\top\mathbf{w}+b$$

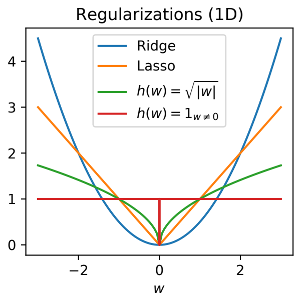
- A measure of complexity of a function is how quick it will change its value.
- This can be measured as the gradient of the function *w.r.t.* its input :

$$\nabla f_\theta(\mathbf{x})=\mathbf{w}$$

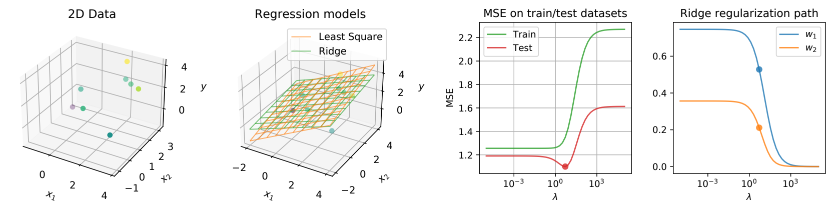
- On measure of complexity is then to use the norm of the linear parameters $\mathbf{w}$.

Common regularizations for linear models

- Ridge : $\Omega(\mathbf{w})=\|\mathbf{w}\|^2=\sum_j w_j^2$
- Lasso : $\Omega(\mathbf{w})=\|\mathbf{w}\|_1=\sum_j |w_j|$
- Mahalanobis : $\Omega(\mathbf{w})=\mathbf{w}^\top\boldsymbol{\Sigma}\mathbf{w}$
- Separable : $\Omega(\mathbf{w})=\sum_j h(|w_j|)$
- Group-Lasso : $\Omega(\mathbf{w})=\sum_{g\in\mathcal{G}}\|\mathbf{w}_g\|$
- L0 pseudo-norm : $\Omega(\mathbf{w})=\|\mathbf{w}\|_0=\sum_j 1_{w_j\neq 0}$



Ridge regression



Principle

$$\min_{\mathbf{w},b} \quad \frac{1}{n}\sum_{i=1}^{n}(y_i-\mathbf{w}^\top\mathbf{x}_i-b)^2+\lambda\|\mathbf{w}\|^2 \tag{22}$$

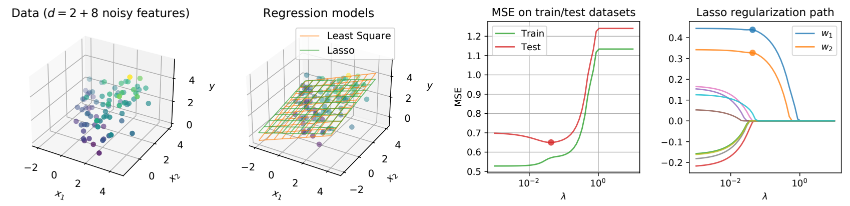
- Quadratic penalization limits the complexity of the model ($\lambda=0$ is LS).
- Makes the optimization problem strictly convex even when $n<d$.
- Solutions without and with bias are

$$\hat{\mathbf{w}}=(\mathbf{X}^\top\mathbf{X}+n\lambda\mathbf{I}_d)^{-1}\mathbf{X}^\top\mathbf{y}, \qquad \hat{\boldsymbol{\theta}}=(\tilde{\mathbf{X}}^\top\tilde{\mathbf{X}}+n\lambda\mathbf{S})^{-1}\mathbf{X}^\top\mathbf{y} \tag{23}$$

Where $\mathbf{S}\in\mathbb{R}^{d+1\times d+1}$ is a matrix defined as $S_{i,j}=1$ if $i=j\leq d$ else 0.

- Ridge with $\lambda=\frac{\sigma_n^2}{\sigma_{\mathbf{w}}^2}$ is actually a MAP with a prior $p(\mathbf{w})\sim\mathcal{N}(0,\sigma_{\mathbf{w}}^2\mathbf{I})$ and a known variance of the additive noise of σ_n^2.
- Scikit-learn implementation (`alpha` is λ) : `sklearn.linear_model.Ridge(alpha=1)`

Lasso regression



Principle [Tibshirani, 1996]

$$\min_{\mathbf{w},b} \quad \frac{1}{n}\sum_{i=1}^{n}(y_i - \mathbf{w}^\top\mathbf{x}_i - b)^2 + \lambda\sum_{j=1}^{d}|w_j| \tag{24}$$

- L1 norm $\|\mathbf{w}\|_1 = \sum_j |w_j|$ regularization is non-smooth in $w_j = 0, \forall j$.
- For a large enough λ the solution of the problem is sparse (some components $\hat{w}_j$ of $\hat{\mathbf{w}}$ are exactly equal to 0).
- Under some conditions, when the true model $\mathbf{w}^\star$ is sparse the true support of $\mathbf{w}^\star$ can be recovered [Zhao and Yu, 2006].
- Lasso regularization can be used for classification [Koh et al., 2007].
- Scikit-learn implementation (`alpha`$=\lambda$) : `sklearn.linear_model.Lasso`
- Efficient solver for large/sparse problems : `celer.Lasso` [Massias et al., 2020]

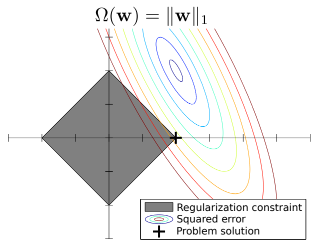
Solution for the Lasso

Why is the Lasso sparse?

- L1 regularization is non-smooth in $w_j = 0, \forall j$ which creates attraction points toward sparsity.
- Lasso Problem (24) is equivalent to

$$\min_{\mathbf{w},b,\|\mathbf{w}\|_1\leq\tau} \quad \frac{1}{n}\sum_{i=1}^{n}(y_i - \mathbf{w}^\top\mathbf{x}_i - b)^2 \tag{25}$$

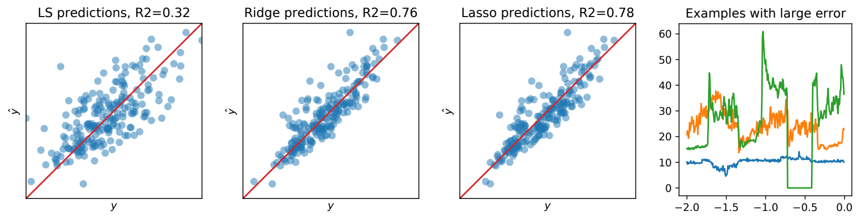
- The geometrical constraints promotes sparse $\mathbf{w}$ on the axis.


$\Omega(\mathbf{w}) = \|\mathbf{w}\|_1$
Regularization constraint
Squared error
Problem solution

Optimization algorithms

- **Coordinate descent** Optimize iteratively each $\mathbf{w}_j$ independently (sklearn).
- **Homotopy Methods** Create iteratively solutions along the regularization path using the fact that it is piece-wise linear (`sklearn.linear_model.lasso_path`).
- **Proximal algorithms** Extension of gradient descent to non-smooth optimization with stochastic solver for large scale datasets (`sklearn.linear_model.SGDRegressor`).

Application on energy usage data



Application to energy usage prediction

- Learn to predict total energy usage for the next day using recordings of usage from the last two days.
- Prediction performance measured with the coefficient of determination R^2 (1 is perfect, 0 is random).
- Comparison for Least Square ($R^2 = 0.32$), Ridge ($R^2 = 0.76$) and Lasso ($R^2 = 0.78$), Ridge and Lasso are far better on large data ($d = 288$).
- Parameters λ selected through cross validation (see next course).
- Plot the predictions and true values (perfect prediction on the red line).
- Plot the linear models $\mathbf{w}$ for all the methods (lasso selects 31/288 features).

Learning nonlinear models

Optimization problem

$$\min_{\boldsymbol{\theta}} \quad \sum_{i=1}^{n} L(y_i, f_{\boldsymbol{\theta}}(\mathbf{x}_i)) + \lambda\Omega(f_{\boldsymbol{\theta}}) \tag{26}$$

- where $f_{\boldsymbol{\theta}}$ is a nonlinear function parametrized by $\boldsymbol{\theta}$.
- Optimization problem can become non-convex and/or non-smooth.
- Different approaches depend on the modeling of the non-linear function $f_{\boldsymbol{\theta}}$.

What kind of nonlinearity ?

- **Non-linear basis :** $\phi_j(\mathbf{x})$ are nonlinear functions and the model is expressed as

$$f_{\boldsymbol{\theta}}(\mathbf{x}) = \sum_{j=1}^{d'}\phi_j(\mathbf{x})w_j + b \tag{27}$$

that can be seen as pre-processing of the data (all linear methods can be applied).
- **Kernel methods :** prediction function lies in a Reproducible Kernel Hilbert Space (RKHS) and non-linearity depends on the choice of the kernel.
- **Neural network :** design the non-linear function as a combination of linear operators and nonlinear transformations. Allows for learning complex feature extraction and taking account the structure of the data.

Representation theorem for kernel methods

Theorem (simplified from [Schölkopf et al., 2001, Boser et al., 1992])

Let $\mathcal{H}$ be a Reproducible Kernel Hilbert Space (RKHS) associated to the positive definite kernel k defined on $\mathbb{R}^d \times \mathbb{R}^d$ and a sampling $\{\mathbf{x}_i, y_i\}_i$. Minimizing the following optimization problem

$$\min_{f\in\mathcal{H}} \quad \sum_{i=1}^{n} L(y_i, f(\mathbf{x}_i)) + h(\|f\|_{\mathcal{H}}) \tag{28}$$

where h is a monotonically increasing function leads to an optimal solution that can be expressed as

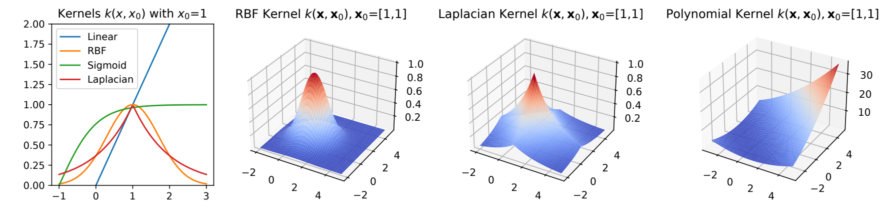
$$\hat{f}(\mathbf{x}) = \sum_{i=1}^{n} \hat{\alpha}_i k(\mathbf{x}, \mathbf{x}_i) + \hat{b} \tag{29}$$

where $\hat{\boldsymbol{\alpha}} \in \mathbb{R}^n$ and $\hat{\mathbf{b}}$ are the parameters of the function.

Discussion on Support Vector Machines (SVM)

- The "kernel trick" used in RKHS allows us to have a non-linear implicit feature extraction $\phi(\mathbf{x}) = k(\mathbf{x}, \cdot)$.
- The norm $\|f\|_{\mathcal{H}}$ in the RKHS can be expressed for a given $f \in \mathcal{H}$ as
$$\|f\|^2_{\mathcal{H}} = \sum_{i,j}^{n,n} \alpha_i \alpha_j k(\mathbf{x}_i, \mathbf{x}_j)$$
- The function f is described through its weight on $k(\mathbf{x}, \mathbf{x}_i)$ the similarity measure with the training samples denoted as support vectors when $\alpha_i \neq 0$.

Kernels as feature extraction

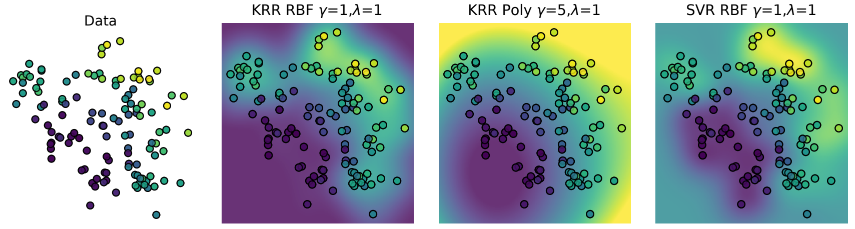


Common kernels (`sklearn.metrics.pairwise`)

- **Linear :** $k(\mathbf{x}, \mathbf{x}') = \mathbf{x}^\top \mathbf{x}'$ (recover linear models where $\mathbf{w} = \sum_i \alpha_i \mathbf{x}_i$)
- **Radial Basis Function (RBF) or Gaussian :** $k(\mathbf{x}, \mathbf{x}') = \exp(-\gamma\|\mathbf{x} - \mathbf{x}'\|^2)$
- **Polynomial :** $k(\mathbf{x}, \mathbf{x}') = (\gamma \mathbf{x}^\top \mathbf{x}' + c_0)^d$
- **Laplacian :** $k(\mathbf{x}, \mathbf{x}') = \exp(-\gamma\|\mathbf{x} - \mathbf{x}'\|)$
- **Cosine :** $k(\mathbf{x}, \mathbf{x}') = \frac{\mathbf{x}^\top \mathbf{x}'}{\|\mathbf{x}\|\|\mathbf{x}'\|}$
- **Sigmoid :** $k(\mathbf{x}, \mathbf{x}') = \tanh(\gamma \mathbf{x}^\top \mathbf{x}' + c_0)$

Numerous kernels have been designed by domain experts for specific applications.

Kernel methods for regression (KRR, SVR)



Kernel Ridge Regression (KRR) ([Murphy, 2012, Chap 14.3])

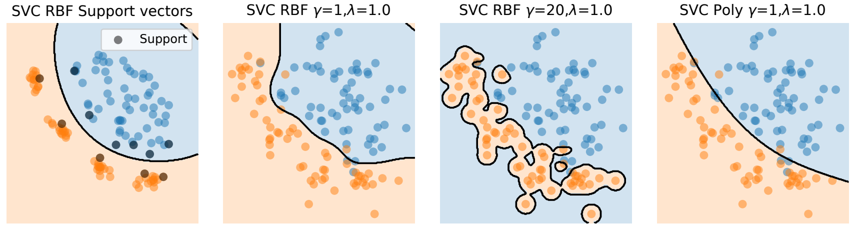
$$\min_{f\in\mathcal{H}} \quad \frac{1}{n}\sum_{i=1}^{n} (y_i - f(\mathbf{x}_i))^2 + \lambda\|f\|^2_{\mathcal{H}} \tag{30}$$

- Optimal parameters $\hat{\boldsymbol{\alpha}} = (\mathbf{K} + n\lambda\mathbf{I}_n)^{-1}\mathbf{y}$ with $\mathbf{K}$ the kernel matrix.
- There exist a Lasso counterpart for sparse $\hat{\boldsymbol{\alpha}}$ [Guigue et al., 2005].
- Scikit-learn implementation (`alpha`=λ) : `sklearn.kernel_ridge.KernelRidge`

Support Vector Regression (SVR) [Drucker et al., 1997]

- Similar to KRR but using the ϵ-invariant loss $L(y, \hat{y}) = \max(0, |y - \hat{y}| - \epsilon)$.
- Solution $\hat{\boldsymbol{\alpha}}$ is sparse (weight on support vectors) and less sensitive to outliers.
- Scikit-learn implementation (`C`=$\frac{1}{\lambda}$): `sklearn.svm.SVR`

Support Vector Classification (SVC)

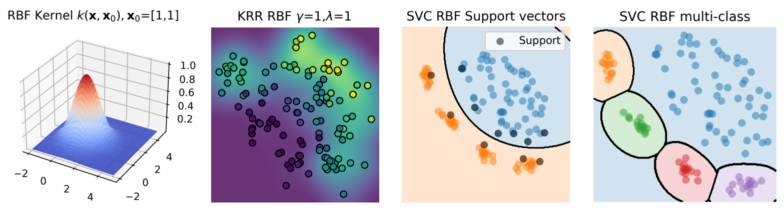


Principle [Boser et al., 1992]

$$\min_{f\in\mathcal{H}} \quad \frac{1}{n}\sum_{i=1}^{n} \max(1 - y_i f(\mathbf{x}_i), 0) + \lambda\|f\|^2_{\mathcal{H}} \tag{31}$$

- The optimization will promote a large margin between the classes.
- The problem (31) can be reformulated as the following convex Quadratic Program
$$\max_{0\leq\boldsymbol{\beta}\leq\frac{1}{2n\lambda}, \boldsymbol{\beta}^\top\mathbf{y}=0} \quad \sum_{j}^{n} \beta_j - \frac{1}{2}\sum_{i,j}^{n,n} \beta_i\beta_j y_i y_j k(\mathbf{x}_i, \mathbf{x}_j) \tag{32}$$
with the solution $\hat{f}(\mathbf{x})$ using the weights $\hat{\alpha}_i = \hat{\beta}_i y_i$.
- Consistent estimator (converges to Bayes for large n) [Steinwart, 2005].
- Scikit-learn implementation: `sklearn.svm.SVC`

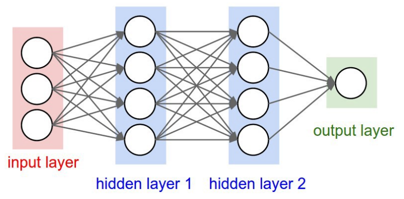
Support Vector Machines

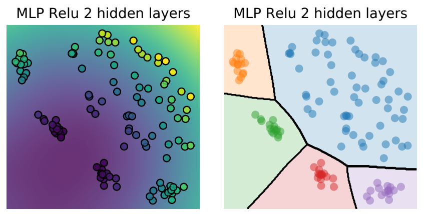


SVM and extensions

- Multi-class classification [Weston and Watkins, 1998] or One-Against-All strategy [Hsu and Lin, 2002] (default in Scikit-learn).
- Estimation of probability of classes done with a logistic regression on the prediction function f [Platt et al., 1999].
- Squared SVM (squared hinge loss) lead to a differentiable problem can be solved with gradient descent [Chapelle, 2007].
- Multiple Kernel Learning allows for learning the feature extraction and selecting the kernel parameter [Bach et al., 2004, Rakotomamonjy et al., 2008].
- Kernels can be approximated using Nyström method [Williams and Seeger, 2001] or Random Fourier Features (RFF) [Rahimi et al., 2007] for learning on large scale datasets (`sklearn.kernel_approximation`).

Neural Networks





Multi-Layer Perceptron (MLP) [Goodfellow et al., 2016, Chapter 6]

$$\min_{\boldsymbol{\theta}} \quad \sum_{i=1}^{n} L(y_i, f_{\boldsymbol{\theta}}(\mathbf{x}_i)) + \lambda\Omega(\boldsymbol{\theta}) \tag{33}$$

- Where the function $f_{\boldsymbol{\theta}}$ is expressed as

$$f_{\boldsymbol{\theta}}(\mathbf{x}) = f_K(f_{K-1}(\dots(f_1(\mathbf{x})))), \quad \text{with} \quad f_k(\mathbf{x}) = \sigma_k(\mathbf{W}_k\mathbf{x} + \mathbf{b}_k) \tag{34}$$

- The parameters are $\boldsymbol{\theta} = \{\mathbf{W}_k, \mathbf{b}_k\}_k$ and σ_k are non-linear activations.
- Highly non-convex and non-smooth optimization problem in general.
- MLP are universal approximators [Hornik et al., 1989].
- Scikit-learn : `sklearn.neural_network.MLPClassifier / MLPRegressor`
- Implementation faster with GPU-compatible toolboxes (Pytorch/tensorflow).

Neural Networks design

Major architectures

- **Convolutional layers** [LeCun et al., 1998] for signal and images use a convolution of the signal $\mathbf{x}$ instead of a general linear operator :

$$f_k(\mathbf{x}) = \sigma_k(\mathbf{w}_k * \mathbf{x} + \mathbf{b}_k)$$

- **Fully convolutional network (U-Nets)** proposed for image segmentation and processing [Long et al., 2015, Ronneberger et al., 2015].
- **Residual Layers** [He et al., 2016] help train deeper network and avoid vanishing gradients (also facilitates recovering the identity function) :

$$f_k(\mathbf{x}) = \sigma_k(\mathbf{W}_k\mathbf{x} + \mathbf{b}_k) + \mathbf{x}$$

- **Recurrent Neural Nets (RNN)** [Rumelhart et al., 1986] and Long short-term memory (LSTM) [Hochreiter and Schmidhuber, 1997] for modeling sequences in signals and Natural Language Processing.
- **Attention models** (transformers) is pointwise product in the layers to focus on some features/parts of the embedding [Vaswani et al., 2017].

Practical implementation

- ReLU activation $\sigma(x) = \max(0, x)$ allows for deeper networks [Glorot et al., 2011, He et al., 2015]
- Initialization of the parameters is important [Glorot and Bengio, 2010].

Optimization of deep neural networks

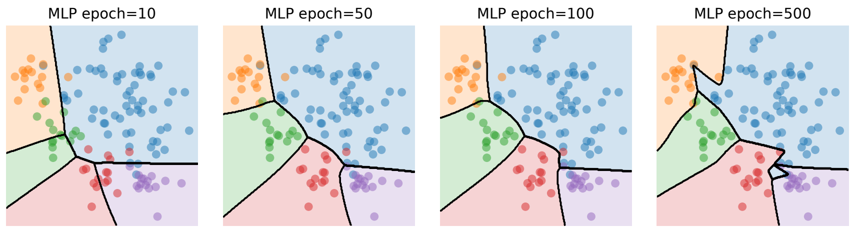
Stochastic Gradient Descent (SGD) on large scale datasets

- Principle : Never compute the full gradient, only on samples (1 or minibatch).
- Going through the whole dataset is called an epoch (numerous gradient steps).
- Fast convergence with averaging (SAG, SRVG, SAGA) [Johnson and Zhang, 2013, Roux et al., 2012, Defazio et al., 2014].
- State of the art algorithm for linear SVM, logistic regression, least square.
- Classification (SVM, Logistic) : `sklearn.linear_model.SGDClassifier`.
- Regression (least square, huber) : `sklearn.linear_model.SGDRegressor`.

Gradient descent for deep learning

- Stochastic Gradient Descent and variants work very well on continuous, non-smooth non-convex problems [Bottou, 2010].
- Use fixed step or change of step size along iterations.
- Several momentum, averaging and adaptive step size strategies:
  - Momentum and Accelerated gradients [Nesterov, 1983]
  - RMSPROP [Tieleman and Hinton, 2012].
  - Adaptive gradient step ADAGRAD [Duchi et al., 2011].
  - Adaptive Moment estimation ADAM [Kingma and Ba, 2014].
- Most optimization strategies implemented in Pytorch/tensorflow.

Regularization of deep neural networks



Regularization strategies [Goodfellow et al., 2016, Chapter 7]

- Ridge (weight decay) or Lasso on the parameters $\mathbf{W}_k$ for smooth prediction.
- Early stopping along the epochs (using validation set) [Yao et al., 2007].
- Dropout shuts down some neurons during training [Srivastava et al., 2014].
- Data Augmentation uses transformation of data (signals) to create new samples and promote invariance [Shorten and Khoshgoftaar, 2019].
- Adversarial regularization penalize the classification error of (virtual) adversarial examples [Miyato et al., 2018].

Ensemble methods

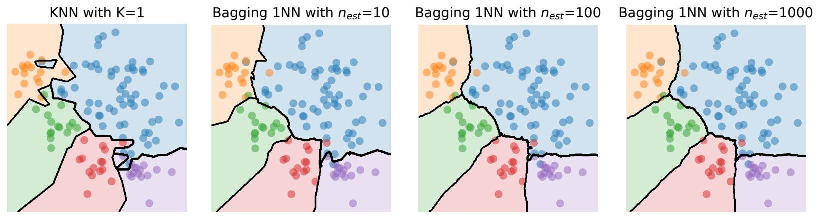
Principle of ensemble methods

- Generalization of a unique predictor is hard to estimate.
- Strength in number (and different opinions).
- Estimate a number of predictors f_k (with some variability).
- Use the predictions of those predictors to reach a consensus that is more robust.
- Theoretical result show that merging prediction from "weak" classifiers can minimize the variance and better generalize.
- Ensemble methods are meta-estimators : they can use existing "black box" estimators.

Two main approaches

- **Averaging methods** Several predictors f_k are build independently and they are averaged for a prediction (Bagging, Random Forests).
- **Boosting** Several predictors f_k are build sequentially to reduce the bias/error of their combination (Adaboost, Gradient Boosting)

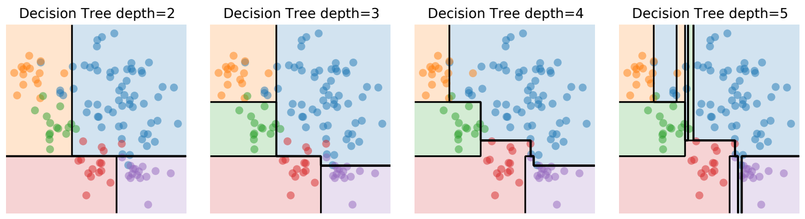
Bagging



Principle [Breiman, 1996]

- Select a supervised estimation method (any supervised predictor).
- Train several predictors on random selection of the train data.
- Sampling subset of samples with replacement is also called **Bootstraping**.
- Predict using majority voting (classification) or average value (regression).
- Several variants where predictors are trained on random subsets :
  - of the features **Random subspaces** [Ho, 1998].
  - of features and samples **Random Patches** [Louppe and Geurts, 2012].
- General implementations can select proportion of selected samples and features.
- Scikit-learn : `sklearn.ensemble.BaggingClassifier` / `BaggingRegressor`

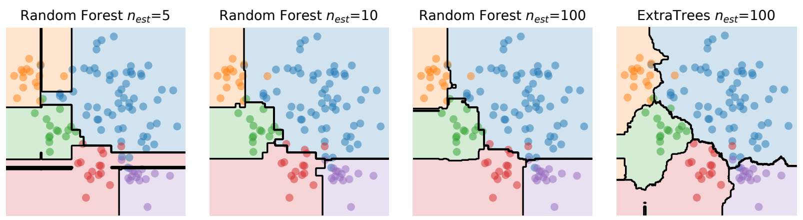
Decision Tree



Principle [Breiman et al., 2017]

- Predictor modeled as a binary tree where each node is a decision based on a threshold of the value of one of the features.
- Standard algorithms are ID3 [Quinlan, 1986] and C4.5 [Quinlan, 1993] and CART that use information entropy to select the variable that will be used on each node.
- Complexity of the tree depends on the depth of the tree.
- Model is very interpretable and explainable : you can express all the reasons for a given decision.
- Rarely used alone in high dimension due to low generalization ability.
- Scikit-learn : `sklearn.tree.DecisionTreeClassifier` / `DecisionTreeRegressor`

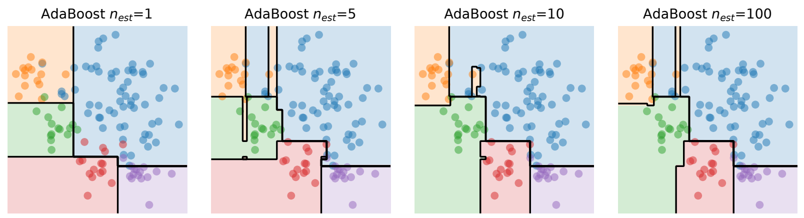
Random Forests (RF)



Principle [Ho, 1995, Breiman, 2001]

- Perform Bagging using Decision Trees as weak classifiers.
- Select only from a random subset of features on each node (similar to random subspaces but on each node).
- Loose some interpretability of the trees but gain generalization performance.
- Similar adaptive neighborhood with RF and KNN [Lin and Jeon, 2006].
- Extremely randomized trees (ExtraTrees) use random thresholds in the trees instead of optimal thresholds as in Decision Trees [Geurts et al., 2006].
- Scikit-learn : `sklearn.tree.RandomForestClassifier / RandomForestRegressor`
  `sklearn.ensemble.ExtraTreesClassifier / ExtraTreesRegressor`

Adaboost



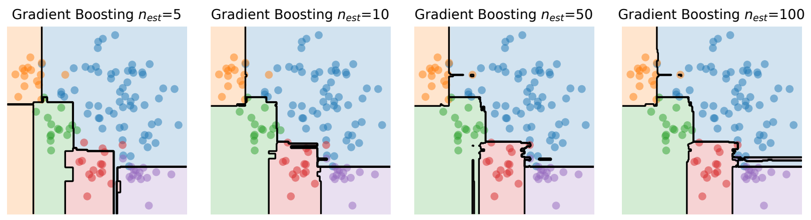
Principle [Freund and Schapire, 1997]

- The predictor is a weighted sum $F_k(\mathbf{x}) = \sum_k \alpha_k f_k$.
- Estimated predictors f_k trained sequentially on weighted training samples.
- At each step k a new predictor is estimated by minimizing :

$$f_k, \alpha_k = \arg\min_{f,\alpha} \sum_{i=1}^{n} e^{-y_i(F_{k-1}(\mathbf{x})+\alpha f(\mathbf{x}))} = \sum_{i=1}^{n} w_i^k e^{-y_i \alpha f(\mathbf{x})} \tag{35}$$

  For binary classification with $f(\mathbf{x}) \in \{-1, 1\}$.
- The weights are updated at each iteration to favor samples miss-predicted by the previous predictors.
- Scikit-learn : `sklearn.ensemble.AdaBoostClassifier / AdaBoostRegressor`

Gradient Boosting (GB)



Principle [Friedman, 2001]

- Generalization of AdaBoost to any differentiable loss L.
- Estimate a predictor $F_k(\mathbf{x}) = \sum_k f_k$ by minimizing iteratively the ERM:

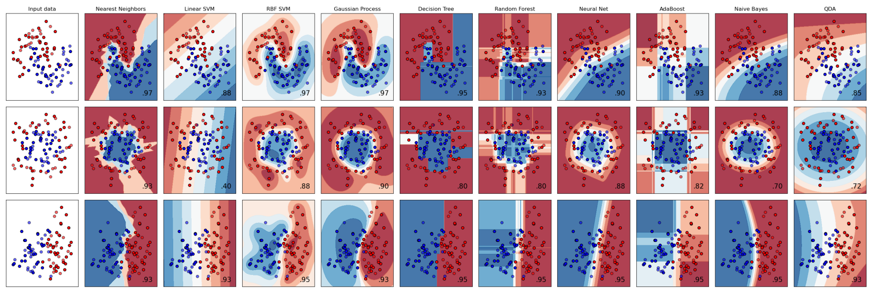
$$f_k = \arg\min_{f} \sum_{i=1}^{n} L(y_i, F_{k-1}(\mathbf{x}) + f(\mathbf{x})) \tag{36}$$

- This is approximated by a gradient descent in the functional space with

$$F_k(\mathbf{x}) = F_{k-1}(\mathbf{x}) - \gamma_m \sum_{i=1}^{n} \nabla_f L(y_i, F_{k-1}(\mathbf{x}) + f(\mathbf{x})) \tag{37}$$

- Stochastic Gradient Boosting use random subsets of samples [Friedman, 2002].
- XGBoost, GB variant, won numerous competitions [Chen and Guestrin, 2016].
- Sklearn : `sklearn.ensemble.GradientBoostingClassifier / GradientBoostingRegressor`

Conclusion



Supervised learning

- Bayesian methods lead to probabilistic predictions but densities can be gard to estimate in high dimension.
- Always try linear models first! They are harder to overfit but use regularization with Ridge or Lasso especially in high dimension.
- For small datasets with nonlinear prediction functions use SVM with hand-crafted kernels (large datasets can use kernel approximation).
- Neural Network can estimate complex functions on large datasets. The different layers can benefit from the structure of the data (convolution on image or signal).
- Gradient Boosting (XGBoost) works in many cases when good features available.

References I

[Bach et al., 2004] Bach, F. R., Lanckriet, G. R., and Jordan, M. I. (2004).
Multiple kernel learning, conic duality, and the smo algorithm.
In *Proceedings of the twenty-first international conference on Machine learning*, page 6.

[Boser et al., 1992] Boser, B. E., Guyon, I. M., and Vapnik, V. N. (1992).
A training algorithm for optimal margin classifiers.
In *Proceedings of the fifth annual workshop on Computational learning theory*, pages 144–152.

[Bottou, 2010] Bottou, L. (2010).
Large-scale machine learning with stochastic gradient descent.
In *Proceedings of COMPSTAT'2010*, pages 177–186. Springer.

[Breiman, 1996] Breiman, L. (1996).
Bagging predictors.
Machine learning, 24(2):123–140.

[Breiman, 2001] Breiman, L. (2001).
Random forests.
Machine learning, 45(1):5–32.

[Breiman et al., 2017] Breiman, L., Friedman, J. H., Olshen, R. A., and Stone, C. J. (2017).
Classification and regression trees.
Routledge.

References II

[Chapelle, 2007] Chapelle, O. (2007).
Training a support vector machine in the primal.
Neural computation, 19(5):1155–1178.

[Chen and Guestrin, 2016] Chen, T. and Guestrin, C. (2016).
Xgboost: A scalable tree boosting system.
In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794.

[Defazio et al., 2014] Defazio, A., Bach, F., and Lacoste-Julien, S. (2014).
Saga: A fast incremental gradient method with support for non-strongly convex composite objectives.
In *Advances in neural information processing systems*, pages 1646–1654.

[Drucker et al., 1997] Drucker, H., Burges, C. J., Kaufman, L., Smola, A., Vapnik, V., et al. (1997).
Support vector regression machines.
Advances in neural information processing systems, 9:155–161.

[Duchi et al., 2011] Duchi, J., Hazan, E., and Singer, Y. (2011).
Adaptive subgradient methods for online learning and stochastic optimization.
Journal of machine learning research, 12(Jul):2121–2159.

[Fisher, 1936] Fisher, R. A. (1936).
The use of multiple measurements in taxonomic problems.
Annals of eugenics, 7(2):179–188.

References III

[Fix and Hodges, 1989] Fix, E. and Hodges, J. L. (1989).
Discriminatory analysis. nonparametric discrimination: Consistency properties.
International Statistical Review/Revue Internationale de Statistique, 57(3):238–247.

[Freund and Schapire, 1997] Freund, Y. and Schapire, R. E. (1997).
A decision-theoretic generalization of on-line learning and an application to boosting.
Journal of computer and system sciences, 55(1):119–139.

[Friedman et al., 2001] Friedman, J., Hastie, T., Tibshirani, R., et al. (2001).
The elements of statistical learning, volume 1.
Springer series in statistics New York.

[Friedman, 2001] Friedman, J. H. (2001).
Greedy function approximation: a gradient boosting machine.
Annals of statistics, pages 1189–1232.

[Friedman, 2002] Friedman, J. H. (2002).
Stochastic gradient boosting.
Computational statistics & data analysis, 38(4):367–378.

[Geurts et al., 2006] Geurts, P., Ernst, D., and Wehenkel, L. (2006).
Extremely randomized trees.
Machine learning, 63(1):3–42.

References IV

[Glorot and Bengio, 2010] Glorot, X. and Bengio, Y. (2010).
Understanding the difficulty of training deep feedforward neural networks.
In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256. JMLR Workshop and Conference Proceedings.

[Glorot et al., 2011] Glorot, X., Bordes, A., and Bengio, Y. (2011).
Deep sparse rectifier neural networks.
In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 315–323. JMLR Workshop and Conference Proceedings.

[Goodfellow et al., 2016] Goodfellow, I., Bengio, Y., and Courville, A. (2016).
Deep learning.
MIT press.

[Guigue et al., 2005] Guigue, V., Rakotomamonjy, A., and Canu, S. (2005).
Kernel basis pursuit.
In *European Conference on Machine Learning*, pages 146–157. Springer.

[He et al., 2015] He, K., Zhang, X., Ren, S., and Sun, J. (2015).
Delving deep into rectifiers: Surpassing human-level performance on imagenet classification.
In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034.

[He et al., 2016] He, K., Zhang, X., Ren, S., and Sun, J. (2016).
Deep residual learning for image recognition.
In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

References V

[Ho, 1995] Ho, T. K. (1995).
Random decision forests.
In *Proceedings of 3rd international conference on document analysis and recognition*, volume 1, pages 278–282. IEEE.

[Ho, 1998] Ho, T. K. (1998).
The random subspace method for constructing decision forests.
IEEE transactions on pattern analysis and machine intelligence, 20(8):832–844.

[Hochreiter and Schmidhuber, 1997] Hochreiter, S. and Schmidhuber, J. (1997).
Long short-term memory.
Neural computation, 9(8):1735–1780.

[Hornik et al., 1989] Hornik, K., Stinchcombe, M., and White, H. (1989).
Multilayer feedforward networks are universal approximators.
Neural networks, 2(5):359–366.

[Hsu and Lin, 2002] Hsu, C.-W. and Lin, C.-J. (2002).
A comparison of methods for multiclass support vector machines.
IEEE transactions on Neural Networks, 13(2):415–425.

[Johnson and Zhang, 2013] Johnson, R. and Zhang, T. (2013).
Accelerating stochastic gradient descent using predictive variance reduction.
In *Advances in neural information processing systems*, pages 315–323.

References VI

[Kingma and Ba, 2014] Kingma, D. P. and Ba, J. (2014).
Adam: A method for stochastic optimization.
arXiv preprint arXiv:1412.6980.

[Koh et al., 2007] Koh, K., Kim, S.-J., and Boyd, S. (2007).
An interior-point method for large-scale l1-regularized logistic regression.
Journal of Machine learning research, 8(Jul):1519–1555.

[LeCun et al., 1998] LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998).
Gradient-based learning applied to document recognition.
Proceedings of the IEEE, 86(11):2278–2324.

[Lin and Jeon, 2006] Lin, Y. and Jeon, Y. (2006).
Random forests and adaptive nearest neighbors.
Journal of the American Statistical Association, 101(474):578–590.

[Long et al., 2015] Long, J., Shelhamer, E., and Darrell, T. (2015).
Fully convolutional networks for semantic segmentation.
In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440.

[Louppe and Geurts, 2012] Louppe, G. and Geurts, P. (2012).
Ensembles on random patches.
In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 346–361. Springer.

References VII

[Massias et al., 2020] Massias, M., Vaiter, S., Gramfort, A., and Salmon, J. (2020).
Dual extrapolation for sparse glms.
Journal of Machine Learning Research, 21(234):1–33.

[Miyato et al., 2018] Miyato, T., Maeda, S.-i., Koyama, M., and Ishii, S. (2018).
Virtual adversarial training: a regularization method for supervised and semi-supervised learning.
IEEE transactions on pattern analysis and machine intelligence, 41(8):1979–1993.

[Murphy, 2012] Murphy, K. P. (2012).
Machine learning: a probabilistic perspective.
MIT press.

[Murphy et al., 2006] Murphy, K. P. et al. (2006).
Naive bayes classifiers.
University of British Columbia, 18(60):1–8.

[Nesterov, 1983] Nesterov, Y. (1983).
A method for unconstrained convex minimization problem with the rate of convergence o (1/kˆ 2).
In *Doklady an ussr*, volume 269, pages 543–547.

[Platt et al., 1999] Platt, J. et al. (1999).
Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods.
Advances in large margin classifiers, 10(3):61–74.

References VIII

[Quinlan, 1986] Quinlan, J. R. (1986).
Induction of decision trees.
Machine learning, 1(1):81–106.

[Quinlan, 1993] Quinlan, J. R. (1993).
C4.5: Programs for Machine Learning.
Morgan Kaufmann Publishers, San Mateo, CA.

[Rahimi et al., 2007] Rahimi, A., Recht, B., et al. (2007).
Random features for large-scale kernel machines.
In *NIPS*, volume 3, page 5. Citeseer.

[Rakotomamonjy et al., 2008] Rakotomamonjy, A., Bach, F., Canu, S., and Grandvalet, Y. (2008).
Simplemkl.
Journal of Machine Learning Research, 9:2491–2521.

[Rao, 1948] Rao, C. R. (1948).
The utilization of multiple measurements in problems of biological classification.
Journal of the Royal Statistical Society. Series B (Methodological), 10(2):159–203.

[Ronneberger et al., 2015] Ronneberger, O., Fischer, P., and Brox, T. (2015).
U-net: Convolutional networks for biomedical image segmentation.
In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer.

References IX

[Roux et al., 2012] Roux, N. L., Schmidt, M., and Bach, F. R. (2012).
A stochastic gradient method with an exponential convergence _rate for finite training sets.
In *Advances in neural information processing systems*, pages 2663–2671.

[Rumelhart et al., 1986] Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986).
Learning representations by back-propagating errors.
nature, 323(6088):533–536.

[Schölkopf et al., 2001] Schölkopf, B., Herbrich, R., and Smola, A. J. (2001).
A generalized representer theorem.
In *International conference on computational learning theory*, pages 416–426. Springer.

[Shorten and Khoshgoftaar, 2019] Shorten, C. and Khoshgoftaar, T. M. (2019).
A survey on image data augmentation for deep learning.
Journal of Big Data, 6(1):1–48.

[Srivastava et al., 2014] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014).
Dropout: a simple way to prevent neural networks from overfitting.
The journal of machine learning research, 15(1):1929–1958.

[Steinwart, 2005] Steinwart, I. (2005).
Consistency of support vector machines and other regularized kernel classifiers.
IEEE transactions on information theory, 51(1):128–142.

References X

[Tharwat, 2016] Tharwat, A. (2016).
Linear vs. quadratic discriminant analysis classifier: a tutorial.
International Journal of Applied Pattern Recognition, 3(2):145–180.

[Tibshirani, 1996] Tibshirani, R. (1996).
Regression shrinkage and selection via the lasso.
Journal of the Royal Statistical Society: Series B (Methodological), 58(1):267–288.

[Tieleman and Hinton, 2012] Tieleman, T. and Hinton, G. (2012).
Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude.
COURSERA: Neural networks for machine learning, 4(2):26–31.

[Vaswani et al., 2017] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017).
Attention is all you need.
In *Advances in neural information processing systems*, pages 5998–6008.

[Weston and Watkins, 1998] Weston, J. and Watkins, C. (1998).
Multi-class support vector machines.
Technical report, Citeseer.

[Williams and Seeger, 2001] Williams, C. and Seeger, M. (2001).
Using the nyström method to speed up kernel machines.
In *Proceedings of the 14th annual conference on neural information processing systems*, number CONF, pages 682–688.

References XI

[Yao et al., 2007] Yao, Y., Rosasco, L., and Caponnetto, A. (2007).
On early stopping in gradient descent learning.
Constructive Approximation, 26(2):289–315.

[Zhang, 2004] Zhang, H. (2004).
The optimality of naive bayes.
AA, 1(2):3.

[Zhao and Yu, 2006] Zhao, P. and Yu, B. (2006).
On model selection consistency of lasso.
The Journal of Machine Learning Research, 7:2541–2563.