

Introduction to Supervised Learning

Pietro Gori

Associate Professor
Equipe IMAGES - Télécom Paris - IPParis
pietro.gori@telecom-paris.fr



Summary

1 Introduction - Supervised Learning

- Feature pre-processing
- Function space, Risk and Loss functions
- The Bias-Complexity tradeoff

2 Linear models for regression

- Ordinary Least Squares
- Ridge regression
- LASSO

3 Model selection and assessment

- Information criteria: AIC and BIC
- Cross-validation

4 Linear parametric models for classification

- Linear discriminant analysis (LDA)
- Quadratic discriminant analysis (QDA) and Naive Bayes
- Logistic regression

5 Non-parametric models

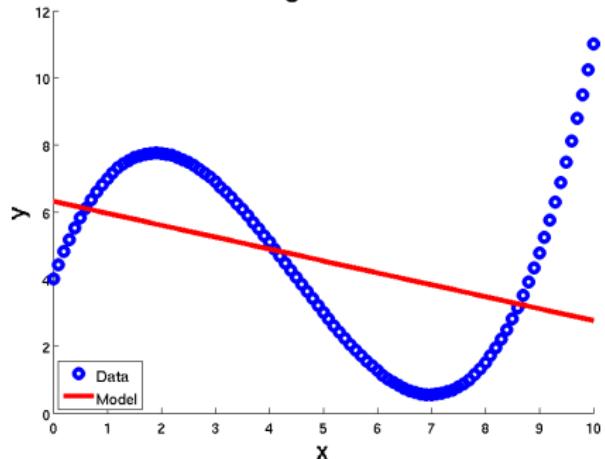
- K-nearest-neighbour

Statistical Machine Learning

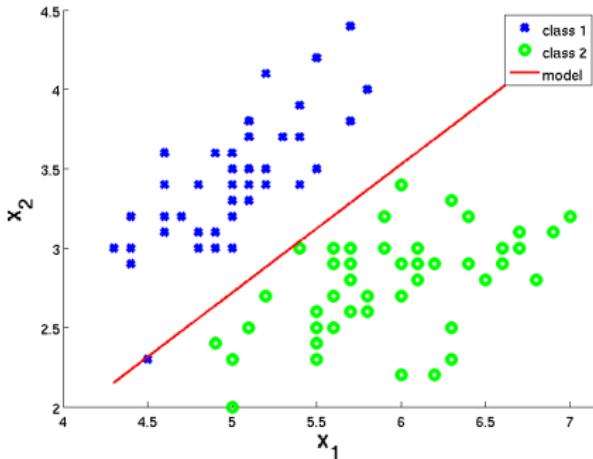
- **Supervised Learning:** inferring a rule f from a labeled training set $(X, Y) = \{(x_1, y_1), \dots, (x_N, y_N)\}$ such that x_i is the vector of features describing the i -th observation and y_i is its label. The rule f will map any new observation t described by x_t to its corresponding y_t .
 - **Regression:** y_i is a set of continuous variables (e.g., $y_i \in \mathbb{R}$)
 - **Classification:** y_i is a set of discrete categories/classes
- **Unsupervised Learning:** No labels are given, we only have X . Possible goals are:
 - **Clustering:** divide X into homogeneous groups
 - **Dimensionality reduction:** simplify X by mapping it into a lower dimensional-space (e.g., PCA)
 - **Outlier detection**

Supervised Learning

Regression



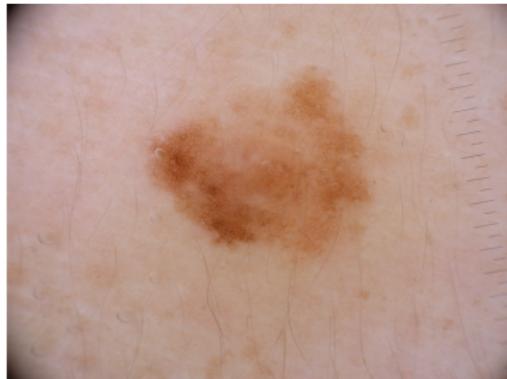
Classification



- Every blue dot is an observation described by a scalar feature $x_i \in \mathbb{R}$ and a scalar *continuous* output $y_i \in \mathbb{R}$
- The goal is to model the relationship between y and x
- Every dot is an observation described by a vector feature $x_i \in \mathbb{R}^2$ and a binary *class* y_i
- The goal is to estimate a model f that divides the input space into two classes

Supervised Learning - Binary Classification

Observation 1



$$x_1 = \begin{cases} \text{Asymmetry} \\ \text{Border} \\ \text{Color} \\ \text{Dimension} \end{cases} \quad y_1 = \text{Nevus}$$

⋮

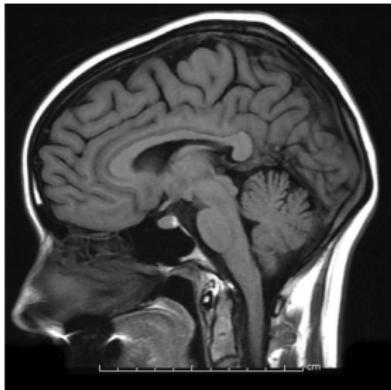
Observation N



$$x_N = \begin{cases} \text{Asymmetry} \\ \text{Border} \\ \text{Color} \\ \text{Dimension} \end{cases} \quad y_N = \text{Melanoma}$$

Supervised Learning - Regression

Subject 1



$$x_1 = \begin{cases} \text{Atrophy hippocampus} \\ \text{Temporal lobe thickness} \\ \text{Volume ventricles} \\ \text{Thickness Corpus Callosum} \end{cases} \quad y_1 = \text{Age subject 1}$$

Subject N



$$x_N = \begin{cases} \text{Atrophy hippocampus} \\ \text{Temporal lobe thickness} \\ \text{Volume ventricles} \\ \text{Thickness Corpus Callosum} \end{cases} \quad y_N = \text{Age subject } N$$

Supervised Learning - Probabilistic framework

- X : real valued random input vector taking values x in $\mathcal{X} = \mathbb{R}^d$

Supervised Learning - Probabilistic framework

- X : real valued random input vector taking values x in $\mathcal{X} = \mathbb{R}^d$
- Y : real valued random output variable taking values y in $\mathcal{Y} = \{C_1, \dots, C_K\}$ (classification with K classes) or in $\mathcal{Y} = \mathbb{R}$ (regression)

Supervised Learning - Probabilistic framework

- X : real valued random input vector taking values x in $\mathcal{X} = \mathbb{R}^d$
- Y : real valued random output variable taking values y in $\mathcal{Y} = \{C_1, \dots, C_K\}$ (classification with K classes) or in $\mathcal{Y} = \mathbb{R}$ (regression)
- p_{XY} : joint probability distribution of (X, Y) , fixed but unknown

Supervised Learning - Probabilistic framework

- X : real valued random input vector taking values x in $\mathcal{X} = \mathbb{R}^d$
- Y : real valued random output variable taking values y in $\mathcal{Y} = \{C_1, \dots, C_K\}$ (classification with K classes) or in $\mathcal{Y} = \mathbb{R}$ (regression)
- p_{XY} : joint probability distribution of (X, Y) , fixed but unknown
- $\mathcal{D}_n = \{(x_i, y_i) \in \mathcal{X} \times \mathcal{Y}\}$: n i.i.d.samples drawn from p_{XY} .

Supervised Learning - Probabilistic framework

- X : real valued random input vector taking values x in $\mathcal{X} = \mathbb{R}^d$
- Y : real valued random output variable taking values y in $\mathcal{Y} = \{C_1, \dots, C_K\}$ (classification with K classes) or in $\mathcal{Y} = \mathbb{R}$ (regression)
- p_{XY} : joint probability distribution of (X, Y) , fixed but unknown
- $\mathcal{D}_n = \{(x_i, y_i) \in \mathcal{X} \times \mathcal{Y}\}$: n i.i.d. samples drawn from p_{XY} .
- $f(x_t) : \mathcal{X} \rightarrow \mathcal{Y}$: function estimated from \mathcal{D}_n to predict the output y_t of a new observation t described by x_t

Supervised Learning - Probabilistic framework

- X : real valued random input vector taking values x in $\mathcal{X} = \mathbb{R}^d$
- Y : real valued random output variable taking values y in $\mathcal{Y} = \{C_1, \dots, C_K\}$ (classification with K classes) or in $\mathcal{Y} = \mathbb{R}$ (regression)
- p_{XY} : joint probability distribution of (X, Y) , fixed but unknown
- $\mathcal{D}_n = \{(x_i, y_i) \in \mathcal{X} \times \mathcal{Y}\}$: n i.i.d. samples drawn from p_{XY} .
- $f(x_t) : \mathcal{X} \rightarrow \mathcal{Y}$: function estimated from \mathcal{D}_n to predict the output y_t of a new observation t described by x_t
- $L(y, f(x)) : \mathcal{Y} \times \mathcal{Y} \rightarrow \mathbb{R}^+$: loss function which measures the prediction error, namely the difference between y and $f(x)$

Supervised Learning - Practical point of view

- From a practical point of view, we usually have a set of n images characterized by one or more channels (c) with p pixels and one vector of classes/outputs \mathbf{y} of size $[n \times 1]$. Images can then be stored as an array \mathbf{x} of size $[n \times p \times c]$ or $[n \times pc]$ depending on the problem.

Supervised Learning - Practical point of view

- From a practical point of view, we usually have a set of n images characterized by one or more channels (c) with p pixels and one vector of classes/outputs \mathbf{y} of size $[n \times 1]$. Images can then be stored as an array \mathbf{x} of size $[n \times p \times c]$ or $[n \times pc]$ depending on the problem.
- The goal is then to predict for a new observation t , described by x_t , its output y_t

Supervised Learning - Practical point of view

- From a practical point of view, we usually have a set of n images characterized by one or more channels (c) with p pixels and one vector of classes/outputs \mathbf{y} of size $[n \times 1]$. Images can then be stored as an array \mathbf{x} of size $[n \times p \times c]$ or $[n \times pc]$ depending on the problem.
- The goal is then to predict for a new observation t , described by \mathbf{x}_t , its output y_t
- If the problem is a binary classification (i.e. $K = 2$), it can be useful to use $\mathcal{Y} = \{-1, 1\}$ in order not to have the zero value. This is important when working with hyperplanes.

Supervised Learning - Practical point of view

- From a practical point of view, we usually have a set of n images characterized by one or more channels (c) with p pixels and one vector of classes/outputs \mathbf{y} of size $[n \times 1]$. Images can then be stored as an array \mathbf{x} of size $[n \times p \times c]$ or $[n \times pc]$ depending on the problem.
- The goal is then to predict for a new observation t , described by x_t , its output y_t
- If the problem is a binary classification (i.e. $K = 2$), it can be useful to use $\mathcal{Y} = \{-1, 1\}$ in order not to have the zero value. This is important when working with hyperplanes.
- If $K > 2$, we can use two strategies:
 - '**One-vs-all**': we estimate K binary classifiers, one for each class. The classification score might be seen as a probability.

Supervised Learning - Practical point of view

- From a practical point of view, we usually have a set of n images characterized by one or more channels (c) with p pixels and one vector of classes/outputs \mathbf{y} of size $[n \times 1]$. Images can then be stored as an array \mathbf{x} of size $[n \times p \times c]$ or $[n \times pc]$ depending on the problem.
- The goal is then to predict for a new observation t , described by \mathbf{x}_t , its output y_t
- If the problem is a binary classification (i.e. $K = 2$), it can be useful to use $\mathcal{Y} = \{-1, 1\}$ in order not to have the zero value. This is important when working with hyperplanes.
- If $K > 2$, we can use two strategies:
 - '**One-vs-allK binary classifiers, one for each class. The classification score might be seen as a probability.**
 - '**One-vs-oneK(K-1)/2 binary classifiers, one for each pair. We choose the class that wins most of the 'duels' (or based on other rules).**

Summary

1 Introduction - Supervised Learning

- Feature pre-processing
- Function space, Risk and Loss functions
- The Bias-Complexity tradeoff

2 Linear models for regression

- Ordinary Least Squares
- Ridge regression
- LASSO

3 Model selection and assessment

- Information criteria: AIC and BIC
- Cross-validation

4 Linear parametric models for classification

- Linear discriminant analysis (LDA)
- Quadratic discriminant analysis (QDA) and Naive Bayes
- Logistic regression

5 Non-parametric models

- K-nearest-neighbour

Features

In many occasions, we want to use both images and categorical scores to solve a problem. How to use both of them within the same machine learning method ?

There are two kinds of features (variables):

- Numerical (quantitative) features. Examples are: image intensities, volumes, surfaces, temperatures, distances, etc.
- Categorical (qualitative) features. Examples are: citizenship, colors, types, sex, cities, perfumes, etc.

As a rule of thumb, if you can add it, it's a numerical variable !

But if we can not add, multiply, divide categorical features. How can we use them ?

Categorical features

Ordinal Encoder

- A first naive solution would be to code them as integers
- If we take the example of citizenship: 'European' could be coded as '0', 'American' as '1' and 'Asian' as '2' → function `OrdinalEncoder` in scikit-learn
- **Problem:** why 'European' is more distant from 'Asian' than 'American' ? → Some models would interpret them as ordered even if they are not !

One Hot Encoder

- A better solution is to transform each categorical feature with K possible values into a binary features with K values where one would be 1 and all others 0 → function `OneHotEncoder` in scikit-learn

Categorical features - One Hot Encoder

- Taking the same example as before, using the 'One Hot Encoder' strategy, we would obtain:

Subject	Citizenship
1	American
2	European
3	Asian
4	Asian
5	American



	European	American	Asian
1	0	1	0
2	1	0	0
3	0	0	1
4	0	0	1
5	0	1	0

Numerical features - Standardization

- Standardization means centering the data (i.e. remove the mean value of each feature) and then scale them by the standard deviation
- **Why ?** : Because many models assume that all features are centered around zero and have similar variance. If a feature is way bigger than the others, it might dominate the objective function and the gradient descent and make the model unable to correctly learn from the other features.
- This is very important and common in machine learning → function [StandardScaler](#) in scikit-learn

Numerical features - Normalization

- Normalization (or Min-Max scaling) scales the data to a fixed range, usually $[0, 1]$ by doing:

$$\mathcal{X}_N = \frac{X - X_{min}}{X_{max} - X_{min}} \quad (1)$$

- Why ?** : It is very common in image processing where data pixel intensities should lie within a certain range (i.e. between 0 and 255 or between 0 and 1)
- Cons** : More sensitive to the presence of outliers
- Function `MinMaxScaler` in scikit-learn
- Other scalers at <https://scikit-learn.org>

Features - Missing values

- When we do not know the value for a variable, it is *very important* to indicate it. We usually write 'NaN' (i.e. Not a Number)

The diagram illustrates the concept of missing values. On the left, there is a table with three rows and two columns. The columns are labeled 'Subject' and 'Citizenship'. The first row contains '1' and 'American'. The second row contains '2' and a dash ('-'). The third row contains '3' and 'Asian'. An arrow points from this table to another table on the right. The right table has the same structure: 'Subject' and 'Citizenship' columns. The first row contains '1' and 'American'. The second row contains '2' and 'NaN'. The third row contains '3' and 'Asian'.

Subject	Citizenship
1	American
2	-
3	Asian

→

Subject	Citizenship
1	American
2	NaN
3	Asian

- Possible solutions:
 - Remove** the entire observation (even if it lacks of only one variable)
 - Attribute** a value using simple statistics such as the average or the most common value (usually used for categorical variables)
 - Use **advanced techniques**: clustering, interpolation, robust methods, etc.

Summary

1 Introduction - Supervised Learning

- Feature pre-processing
- Function space, Risk and Loss functions
- The Bias-Complexity tradeoff

2 Linear models for regression

- Ordinary Least Squares
- Ridge regression
- LASSO

3 Model selection and assessment

- Information criteria: AIC and BIC
- Cross-validation

4 Linear parametric models for classification

- Linear discriminant analysis (LDA)
- Quadratic discriminant analysis (QDA) and Naive Bayes
- Logistic regression

5 Non-parametric models

- K-nearest-neighbour

Function space

First question: Which model f should we use ? We do not usually know it a priori. We need to do an assumption, constraining it to a function space \mathcal{F} . We can divide existing algorithms into two categories:

- **Parametric models:** f depends on a set of *fixed* parameters θ :
$$\mathcal{F} = \{f(X; \theta), \theta \in \Theta\}$$
. An example is linear regression.
- **Non-parametric models:** the form of f is not specified. They are more flexible. They use the training data to locally estimate the best f . An example is the k-nearest neighbors algorithm. It estimates f based on the k most similar training data. It only assumes that samples that have a similar x value should also have a similar y value. In this case, the number of parameters is *not fixed* beforehand but it grows with the number of samples (N/k)!

- **Parametric models**

- *Benefits:* Simple, Easy to interpret, Fast, Less data
- *Limitations:* Constrained, Accuracy not guaranteed

- **Non-parametric models**

- *Benefits:* Flexibility, Not constrained (more general), more Accurate (if enough data)
- *Limitations:* Many data, Slow, Not always easy to interpret

Risk function

Second question: How do we choose the best f ? We look for the f in the restricted space \mathcal{F} that minimizes the average loss that is called the **risk function** or **expected prediction error** associated to f :

$$\tilde{f}(x) = \arg \min_{f \in \mathcal{F}} R(f) \quad (2)$$

where $R(f)$ is defined as:

$$\begin{aligned} R(f) &= E_{XY}[L(Y, f(X))] = \int_{\mathcal{X}} \int_{\mathcal{Y}} L(y, f(x)) p_{X,Y}(x, y) dx dy \\ &= \int_{\mathcal{X}} \left[\int_{\mathcal{Y}} L(y, f(x)) p_{Y|X}(y|x) dy \right] p_X(x) dx \\ &= E_X [E_{Y|X}[L(Y, f(X))|X]] \end{aligned} \quad (3)$$

Examples of loss functions

Third question: how do we choose $L(y, f(x))$? This is the loss function which measures the error of the classifier/model.

It depends on the application (classification or regression), on the data (presence of outliers) and on the goal (is interpretability important ?)

Here are some examples:

- Squared error loss (L_2 -loss): $L(y, f(x)) = (y - f(x))^2$
- Absolute error loss (L_1 -loss): $L(y, f(x)) = |y - f(x)|$
- L_q -loss: $L(y, f(x)) = (y - f(x))^q \quad q > 0$
- Zero-one loss: $L(y, f(x)) = I(y \neq f(x))$, where I is the Indicator function, namely it is equal to 0 if $y = f(x)$ and 1 otherwise.

Question: Which is the one for classification ?

Risk function - Theory

We look for the function f which minimizes the risk function $R(f)$:

$$f^* = \arg \min_f R(f) = E_X [E_{Y|X} [L(Y, f(X)) | X]] \quad (4)$$

In the discrete case, this can be seen as a sum:

$R(f) = \sum_{x_j \in \mathcal{X}} (E_{Y|X} [L(Y, f(X)) | X]) P(X = x_j)$. We can thus minimize every summand individually, which means solving for every $x_j \in \mathcal{X}$:

$$f^*(x_j) = \arg \min_f E_{Y|X} [L(Y, f(x_j)) | X = x_j] \quad (5)$$

Risk function - L2-loss

For L_2 -losses we obtain:

$$\begin{aligned} f^*(x_j) &= \arg \min_f E_{Y|X}[L(Y, f(x_j))|X = x_j] \\ &= \arg \min_f E_{Y|X}[(Y - f(x_j))^2|X = x_j] \\ &= E_{Y|X}[Y|X = x_j] \end{aligned} \tag{6}$$

That is, under the squared error loss, the best function f^* at every point $X = x_j$ is the conditional expectation $E_{Y|X}[Y|X = x_j]$, which is also called the *regression function*. This is the conditional average of Y ($\int_{y \in \mathcal{Y}} y p_{Y|X}(y|x_j) dy$) at every point $X = x_j$.

Risk function - Zero-one loss

The Zero-one loss is usually used in classification where

$\mathcal{Y} = \{C_1, \dots, C_K\}$. In this case, we can rewrite the risk function:

$$R(f) = E_X \left[\sum_{k=1}^K L(C_k, f(X)) P(Y = C_k | X) \right] \quad (7)$$

Solving again for every $x_j \in \mathcal{X}$, we obtain:

$$f^*(x_j) = \arg \min_{f \in \mathcal{Y}} \sum_{k=1}^K L(C_k, f(x_j)) P(Y = C_k | X = x_j) \quad (8)$$

Being a sum, we want $f(x_j)$ to be equal to the C_k with the greatest $P(Y = C_k | X = x_j)$, so that to minimize the cost function.

That is, under the Zero-one loss, the best function f^* at every point $x = x_j$, is the most probable class C_k . This is known as the *Bayes classifier*.

Bayes classifier

We can rewrite the Bayes classifier in the following way:

$$f^*(x_j) = \arg \max_{f \in \mathcal{Y}} P(Y = f(x_j) | X = x_j) = \arg \max_{C_k} P(X = x_j | Y = C_k) \pi_{C_k} \quad (9)$$

where $P(X = x_j | Y = C_k) = P_k(x)$ and π_{C_k} is the prior of class C_k .

Example: We have two classes $k \in \{1, 2\}$ and we suppose that $P_1(x) = \mathcal{N}(-1, 1)$ and $P_2(x) = \mathcal{N}(1, 1)$ and that the two classes have equal priors $\pi_1 = \pi_2 = 0.5$.

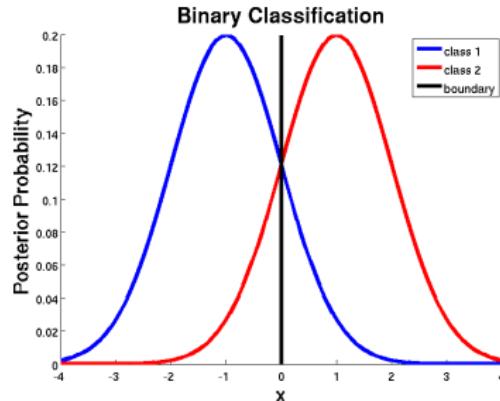
Bayes classifier

How do we classify a new observation $x_j = 0.5$?

$$f^*(x_j) = \arg \max_{C_k} P_k(x_j) \pi_{C_k} \quad (10)$$

With

- $P_1(x_j) \sim \exp\left(-\frac{(x_j - (-1))^2}{2}\right)$
- $P_2(x_j) \sim \exp\left(-\frac{(x_j - 1)^2}{2}\right)$
- $\pi_1 = \pi_2 = 0.5$



Empirical risk minimization

Problem: Since we usually do not know the joint distribution p_{XY} , we can either make assumptions, for instance on $P_k(x_j)$ as in LDA, or compute an approximation of $R(f)$ called *empirical risk* $R_{emp}(f)$ by averaging the loss function L over the training set \mathcal{D}_n :

$$R_{emp}(f) = \frac{1}{n} \sum_{i=1}^n L(y_i, f(x_i)) \quad (11)$$

The *empirical risk minimization principle*¹ states that we should choose the $f \in \mathcal{F}$ that minimizes $R_{emp}(f)$:

$$\hat{f}(x) = \arg \min_{f \in \mathcal{F}} R_{emp}(f) \quad (12)$$

¹Vapnik - Principles of Risk Minimization for Learning Theory - 1992

Empirical risk minimization

Let \mathcal{F} be the restricted space of f :

- f^* is the theoretical optimal model
- \tilde{f} is the theoretical best model in the restricted space \mathcal{F}
- \hat{f} is the best model in the restricted space \mathcal{F} based on the limited data \mathcal{D}_n

We can notice that:

$$\text{err}\{f^* - \hat{f}\} = \text{err}\{ \underbrace{f^* - \tilde{f}}_{\text{approximation error}} \} + \text{err}\{ \underbrace{\tilde{f} - \hat{f}}_{\text{estimation error}} \} \quad (13)$$

- The *approximation error* is due to the restriction of f to \mathcal{F} since it is possible that $f^* \notin \mathcal{F}$.
- The *estimation error* is due to the limited training data. This tends to 0 when $n \rightarrow \infty$, e.g. we observe all possible cases.

Summary

1 Introduction - Supervised Learning

- Feature pre-processing
- Function space, Risk and Loss functions
- The Bias-Complexity tradeoff

2 Linear models for regression

- Ordinary Least Squares
- Ridge regression
- LASSO

3 Model selection and assessment

- Information criteria: AIC and BIC
- Cross-validation

4 Linear parametric models for classification

- Linear discriminant analysis (LDA)
- Quadratic discriminant analysis (QDA) and Naive Bayes
- Logistic regression

5 Non-parametric models

- K-nearest-neighbour

The Bias-Complexity tradeoff

Depending on the choice of \mathcal{F} it can happen that the model f is:

- **Too Rich** (many parameters)
 - Approximation error gets smaller (*lower bias*)
 - Estimation error gets larger (*higher statistical complexity*)
 - *Overfitting*
- **Too simple f** (few parameters)
 - Approximation error gets larger (*higher bias*)
 - Estimation error is likely to reduce (*lower statistical complexity*)
 - *Underfitting*

We should look for a “simple” f suitable for our problem ! How to choose f ?

Training, Validation and Test Set

Let $\{f_i\} \in \mathcal{F}$ be an ensemble of models that we want to test. We first divide the available data into three sets, Training, Validation and Test, which we use in the following way:

- **Training Set \mathcal{T} :** we use it to fit all the models $f_i \in \mathcal{F}$ (i.e. estimate the parameters)
- **Validation Set:** we use it to compare the performances of all f_i and choose the best one \hat{f}_i
- **Test Set:** we use it to test the performance of the best \hat{f}_i using an *evaluation measure* on previously unseen, namely not used, data.

As rule of thumb, the available data are usually split into 50% for training, 25% for validation and 25% for testing.

Evaluation measures

Regression - The performance of a regression model \hat{f}_i can be evaluated using $R_{emp}(\hat{f}_i)$ computed on the *test samples*. This is called *expected prediction error*.

Classification - Suppose that every test sample s is a subject described by some features x_s (e.g., blood pressure, weight, height, etc.). The output y_s can belong to two classes: healthy or sick. The performance of the classifier \hat{f}_i can be evaluated using:

- *sensitivity* (or recall) $\frac{TP}{TP+FN}$
- *specificity* (or selectivity): $\frac{TN}{FP+TN}$
- *precision* : $\frac{TP}{TP+FP}$
- *accuracy*: $\frac{TP+TN}{TP+FP+FN+TN}$
- *F1 score*: $\frac{2TP}{2TP+FP+FN}$

Where TP=correctly identified, FP = incorrectly identified, TN = correctly rejected, FN = incorrectly rejected.

Evaluation measures - Classification

Confusion matrix

- It is a matrix showing whether the system mislabels one class as another one. With $K = 3$ with 10 European, 50 American and 90 Asian, one might obtain:

		Predicted classes		
		European	American	Asian
True class	European	7	3	0
	American	20	20	10
	Asian	10	30	50

- If we only have two classes (e.g. European and Asian), the matrix displays TP, TN, FP and FN:

		Predicted classes	
		European	Asian
True class	European	0 (TN)	10 (FP)
	Asian	0 (FN)	90 (TP)

Evaluation measures - Classification - Imbalanced datasets

		Predicted classes	
		European	Asian
True class	European	0 (TN)	10 (FP)
	Asian	0 (FN)	90 (TP)

- Using the previous example, we can see that the two classes are imbalanced → classifying all subjects as 'Asian' produces an accuracy score of 0.9 (max is 1)
- Balanced accuracy (like other metrics) overcomes this issue by taking into account the size of each class. In the previous example it would give a score of 0.45 (max is 1):

$$\text{balanced accuracy: } \frac{1}{2} \left(\frac{TP}{TP+FN} + \frac{TN}{TN+FP} \right)$$

- For other metrics available in scikit-learn, please refer to: https://scikit-learn.org/stable/modules/model_evaluation.html

Evaluation measures - Multi-class Classification

- Another evaluation measure used in multi-class classification (especially when classes are imbalanced) is the **weighted (categorization) accuracy** WA .
- Let $G = G_1, \dots, G_K$ be the K classes the data-set is divided into. We define it as:

$$WA = \frac{1}{N} \sum_{s=1}^N w_s I(y_s = \hat{y}_s) \quad \text{s.t. } \sum_{s=1}^N w_s = N \quad (14)$$

- Here, I is the indicator function, N the number of observations, \hat{y}_s the prediction of the model f for sample s and w_s its weight. Weights are constrained so that the maximum of WA is equal to 1. Please note that without w_s this would be exactly equal to the Accuracy.
- Weights can be used to give more importance to specific samples or classes.

Evaluation measures - Multi-class Classification

- If we decide to give the same weight to the samples belonging to the same class, we obtain:

$$\sum_{s=1}^N w_s = \sum_{t=1}^K \sum_{s \in G_t} w_s = \sum_{t=1}^K w_t |G_t| = N \quad (15)$$

- where w_t is the weight assigned to class t and $|G_t|$ is the number of elements of class G_t . We can notice that if w_t is equal to $w_t = \frac{N}{K|G_t|}$ then the constraint is fulfilled.
- By plugging it into the WA definition, we obtain:

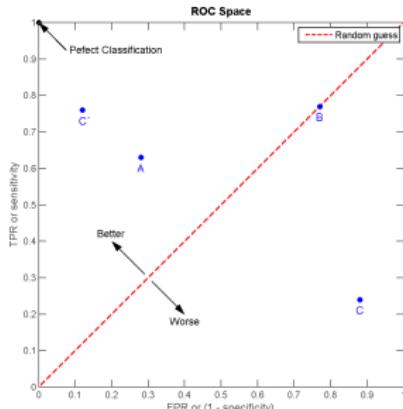
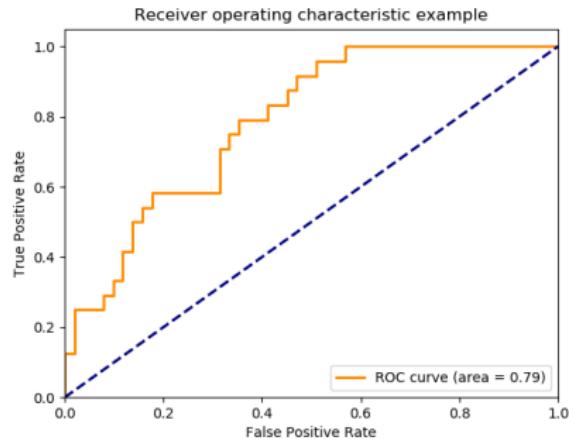
$$WA = \frac{1}{K} \sum_{t=1}^K \sum_{s \in G_t} \frac{I(y_s = \hat{y}_s)}{|G_t|} \quad (16)$$

- WA thus becomes the average across classes of the proportion of correct predictions within each class. This is exactly equal to the Balanced accuracy when $K = 2$.

Evaluation measures - ROC curves

- In a **binary** classification problem, we may predict directly the class or the probability P for each class
- Then, we can choose a threshold T on the probability to discriminate between the two classes ($P < T \rightarrow$ class 0, $P \geq T \rightarrow$ class 1)
- A receiver operating characteristic (ROC) curve shows the classification ability of a binary classifier as we vary the threshold
- It plots the True Positive Rate (TPR) or sensitivity against the False Positive Rate (FPR) or (1-specificity)
- Good model has only points in the top-left corner of the plot
- To compare two models, one could compute the **Area Under the Curve** (AUC) which varies between 0 (worst) and 1 (best)
- In case of class imbalance, Precision-Recall curves are more adapted.

Evaluation measures - ROC curves



Summary

1 Introduction - Supervised Learning

- Feature pre-processing
- Function space, Risk and Loss functions
- The Bias-Complexity tradeoff

2 Linear models for regression

- Ordinary Least Squares
- Ridge regression
- LASSO

3 Model selection and assessment

- Information criteria: AIC and BIC
- Cross-validation

4 Linear parametric models for classification

- Linear discriminant analysis (LDA)
- Quadratic discriminant analysis (QDA) and Naive Bayes
- Logistic regression

5 Non-parametric models

- K-nearest-neighbour

Summary

1 Introduction - Supervised Learning

- Feature pre-processing
- Function space, Risk and Loss functions
- The Bias-Complexity tradeoff

2 Linear models for regression

- Ordinary Least Squares
- Ridge regression
- LASSO

3 Model selection and assessment

- Information criteria: AIC and BIC
- Cross-validation

4 Linear parametric models for classification

- Linear discriminant analysis (LDA)
- Quadratic discriminant analysis (QDA) and Naive Bayes
- Logistic regression

5 Non-parametric models

- K-nearest-neighbour

Example - Linear model

In a linear model, we assume that the output random variable Y and the input random variable X have a linear relationship:

$$f(x; \theta) = \beta_0 + \sum_{j=1}^d \beta_j x_j \quad \text{with } \theta = \{\beta_j\}_{j=0,\dots,d} \quad (17)$$

By defining a dummy variable $x_0 = 1$ we can rewrite the previous Eq. as:

$$f(x; \theta) = \sum_{j=0}^d \beta_j x_j = x^T \beta \quad (18)$$

where $\beta = (\beta_0, \dots, \beta_d)^T$ and $x = (x_0, \dots, x_d)^T$ are two $(d+1) \times 1$ vectors.

Ordinary Least Squares

Let n be the number of samples in the training set, for every sample i we can write:

$$y_i = f(x_i; \theta) + \epsilon_i = x_i^T \beta + \epsilon_i \quad (19)$$

where ϵ_i is a random variable called *residual* which accounts for the prediction error.

Let \mathbf{x} be the $n \times (d + 1)$ matrix containing the x_i , \mathbf{y} the $n \times 1$ vector containing the y_i and $\boldsymbol{\epsilon}$ the $n \times 1$ vector containing the ϵ_i , we can write the previous Eq. in matrix notation:

$$\mathbf{y} = \mathbf{x}\beta + \boldsymbol{\epsilon} \quad (20)$$

In **ordinary least squares** (OLS) we look for the β coefficients that minimize the residual sum of squares (RSS), that is $R_{emp}(f)$ with a L2-loss.

Ordinary Least Squares

The goal of OLS is:

$$\begin{aligned}\beta^* &= \arg \min_{\beta} \text{RSS} = \sum_{i=1}^n (y_i - x_i^T \beta)^2 \\ &= \arg \min_{\beta} (\mathbf{y} - \mathbf{x}\beta)^T (\mathbf{y} - \mathbf{x}\beta)\end{aligned}\tag{21}$$

Differentiating with respect to β , we obtain:

$$\begin{aligned}\frac{\partial \text{RSS}}{\partial \beta} &= -2\mathbf{x}^T(\mathbf{y} - \mathbf{x}\beta) = 0 \\ \beta^* &= (\mathbf{x}^T \mathbf{x})^{-1} \mathbf{x}^T \mathbf{y}\end{aligned}\tag{22}$$

Ordinary Least Squares - Geometry

From the previous equations we can notice that we choose β^* such that the residuals $\mathbf{y} - \mathbf{x}\beta$ are orthogonal to the subspace given by the columns of \mathbf{x} , that is $\mathbf{x}^T(\mathbf{y} - \mathbf{x}\beta) = 0$.

Furthermore we can see that:

$$f(\mathbf{x}; \beta) = \mathbf{x}\beta^* = \mathbf{x}(\mathbf{x}^T \mathbf{x})^{-1} \mathbf{x}^T \mathbf{y} \quad (23)$$

Thus, the resulting estimate $\mathbf{x}\beta^*$ is the orthogonal projection of \mathbf{y} onto the columns of \mathbf{x} .

Ordinary Least Squares - Geometry

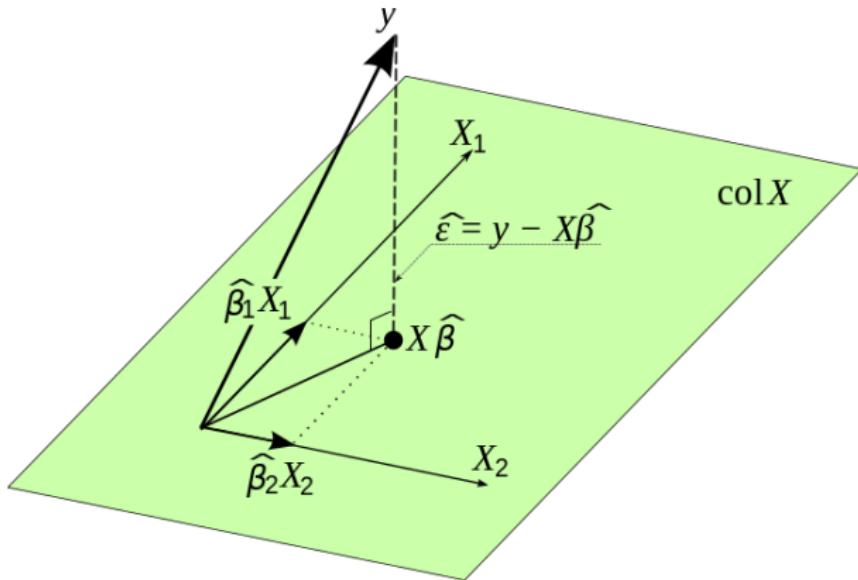


Figure 1: The OLS estimate $\mathbf{x}\hat{\beta}^*$ is the orthogonal projection of \mathbf{y} onto the columns of \mathbf{x} . Image from Wikipedia.

Ordinary Least Squares - Assumptions

Under the following assumptions, the **Gauss–Markov theorem** states that the best linear unbiased estimator (BLUE) is the OLS estimator.

Assumptions:

- \mathcal{D}_n is composed of n i.i.d. samples drawn from p_{XY}
- **Linearity:** linear relationship in the parameters β , not necessarily between y and x . This means that $y = \beta_0 + \beta_1 x^2$ is still valid ! We also assume that we have chosen the proper functional space \mathbf{F} .
- **x must have full column rank:** columns of x are linearly independent, namely $E[x^T x]$ must be finite and positive semi-definite.
- **x must be non-stochastic or at least independent of ϵ :** $E[x^T \epsilon] = 0$
- **Exogeneity:** $E[\epsilon_i | x] = 0$ for all i .
- **Homoscedasticity:** $\text{Var}[\epsilon_i | x] = \sigma^2 < \infty$ for all i
- **Uncorrelated errors:** $\text{Cov}(\epsilon_i, \epsilon_j | x_i, x_j) = 0$ for $i \neq j$
- (Optional) Normal distribution $\epsilon \sim \mathcal{N}(0, \sigma^2 \mathbf{I})$

Ordinary Least Squares - Assumptions

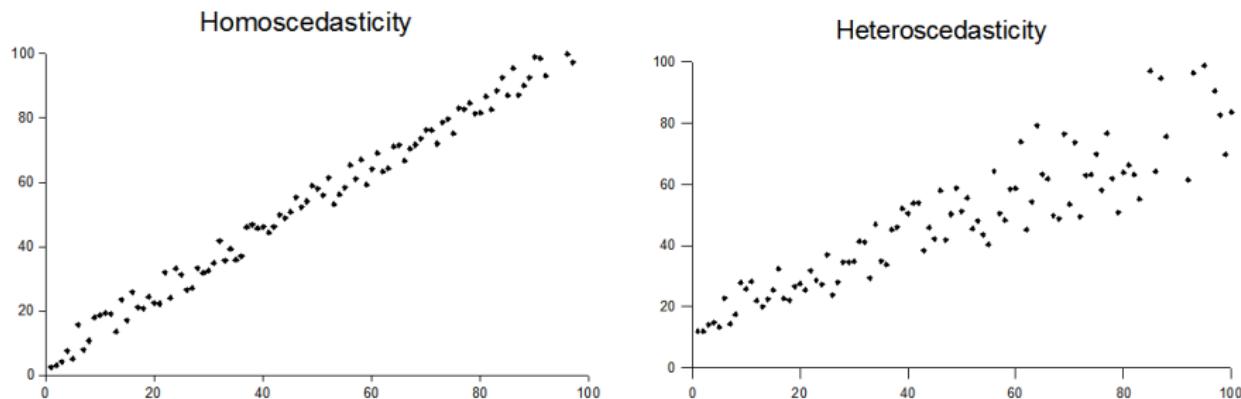


Figure 2: Difference between homoscedasticity (constant variability of the residuals) and heteroscedasticity. Image from Wikipedia.

Ordinary Least Squares

$$\begin{aligned}\beta^* &= (\mathbf{x}^T \mathbf{x})^{-1} \mathbf{x}^T \mathbf{y} \\ &= (\mathbf{x}^T \mathbf{x})^{-1} \mathbf{x}^T (\mathbf{x}\beta + \epsilon) \\ &= \beta + (\mathbf{x}^T \mathbf{x})^{-1} \mathbf{x}^T \epsilon\end{aligned}\tag{24}$$

From this we can see that OLS is unbiased $\mathbf{E}[\beta^*] = \beta$ only if \mathbf{x} is deterministic and $\mathbf{E}[\epsilon] = 0$, or at least if $\mathbf{E}[\mathbf{x}^T \epsilon] = 0$.

Furthermore, “best linear unbiased estimator” means that β^* has the smallest variance among all linear unbiased estimators (to prove in the TP...). Using $\mathbf{E}[\epsilon \epsilon^T] = \sigma^2 \mathbf{I}$:

$$\begin{aligned}\mathbf{E}[(\beta^* - \beta)(\beta^* - \beta)^T] &= \mathbf{E}[((\mathbf{x}^T \mathbf{x})^{-1} \mathbf{x}^T \epsilon)((\mathbf{x}^T \mathbf{x})^{-1} \mathbf{x}^T \epsilon)^T] \\ &= (\mathbf{x}^T \mathbf{x})^{-1} \mathbf{x}^T \mathbf{E}[\epsilon \epsilon^T] \mathbf{x} (\mathbf{x}^T \mathbf{x})^{-1} \\ &= \sigma^2 (\mathbf{x}^T \mathbf{x})^{-1}\end{aligned}\tag{25}$$

Ordinary Least Squares

It is important to notice that if the normality assumption holds (that is $\epsilon \sim \mathcal{N}(0, \sigma^2 \mathbf{I})$) it follows that

$$\beta^* \sim \mathcal{N}(\beta, \sigma^2 (\mathbf{x}^T \mathbf{x})^{-1}) \quad (26)$$

and in this case, the OLS estimator is also **efficient**, that is to say has the smallest variance among all unbiased, linear and non-linear, estimators. (Cramér–Rao inequality)

Bias-Complexity tradeoff - revisited

Let (x_0, y_0) be a test sample, $y_0 = x_0^T \beta + \epsilon_0$ the true model and $x_0^T \beta^*$ the OLS estimate. The expected prediction error, that is the mean squared error (MSE) when using a L2-norm, of OLS at (x_0, y_0) is equal to:

$$\begin{aligned}\text{MSE} &= \mathbf{E}[(y_0 - x_0^T \beta^*)^2] \\ &= \mathbf{E}[(x_0^T \beta + \epsilon_0 - x_0^T \beta^*)^2] \\ &= \mathbf{E}[\epsilon_0^2] + (x_0^T \beta)^2 + \mathbf{E}[(x_0^T \beta^*)^2] - 2(x_0^T \beta) \mathbf{E}[(x_0^T \beta^*)] \\ &\quad + \mathbf{E}[(x_0^T \beta^*)]^2 - \mathbf{E}[(x_0^T \beta^*)]^2 \\ &= \sigma^2 + (\mathbf{E}[(x_0^T \beta^*)^2] - \mathbf{E}[(x_0^T \beta^*)]^2) + (\mathbf{E}[(x_0^T \beta^*)] - (x_0^T \beta))^2 \quad (27) \\ &= \underbrace{\sigma^2}_{\text{irreducible error}} + \text{Var}(x_0^T \beta^*) + \text{Bias}^2(x_0^T \beta^*) \\ &= \sigma^2 + \sigma^2(x_0^T (\mathbf{x}^T \mathbf{x})^{-1} x_0) + 0\end{aligned}$$

Bias-Complexity tradeoff - revisited

The first term σ^2 is beyond our control. It is the inherent error in the data that we can not remove. The other two terms are exactly what we have seen before:

- *Bias* is the *approximation error*, namely the difference between the true model and the theoretical best model in \mathcal{F} .
- *Variance* is the *estimation error*, namely the difference between the theoretical best model in \mathcal{F} and the estimated model in \mathcal{F} using a limited amount of data \mathcal{D}_n .

The Bias will most likely decrease if we enrich \mathcal{F} , namely if we increase d . At the same time, if we increase d , the variance will also increase ! This is the **bias-variance tradeoff** !

Bias-variance tradeoff

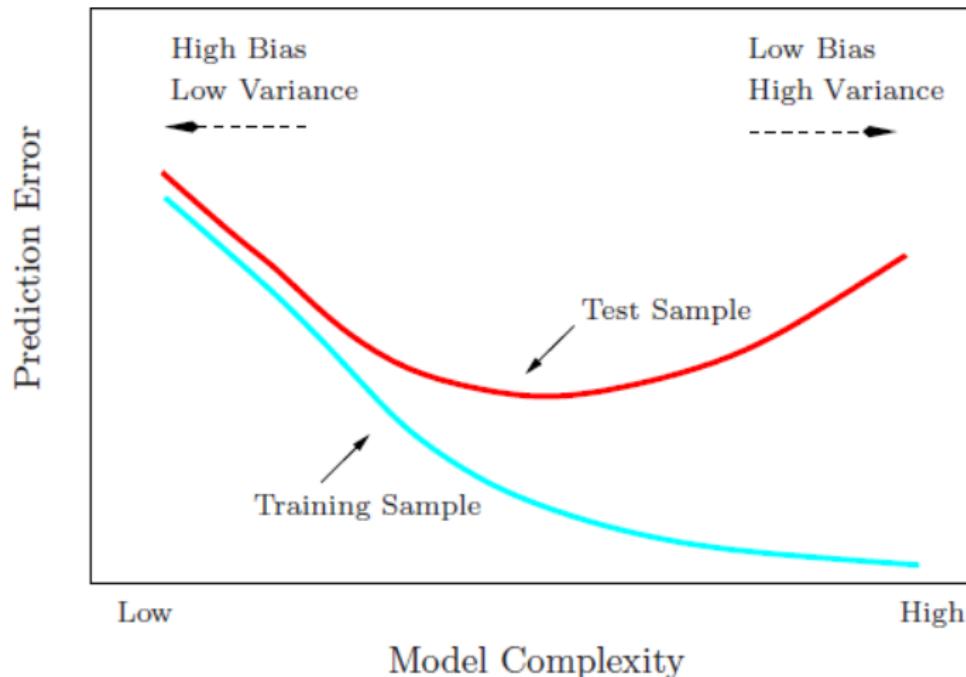
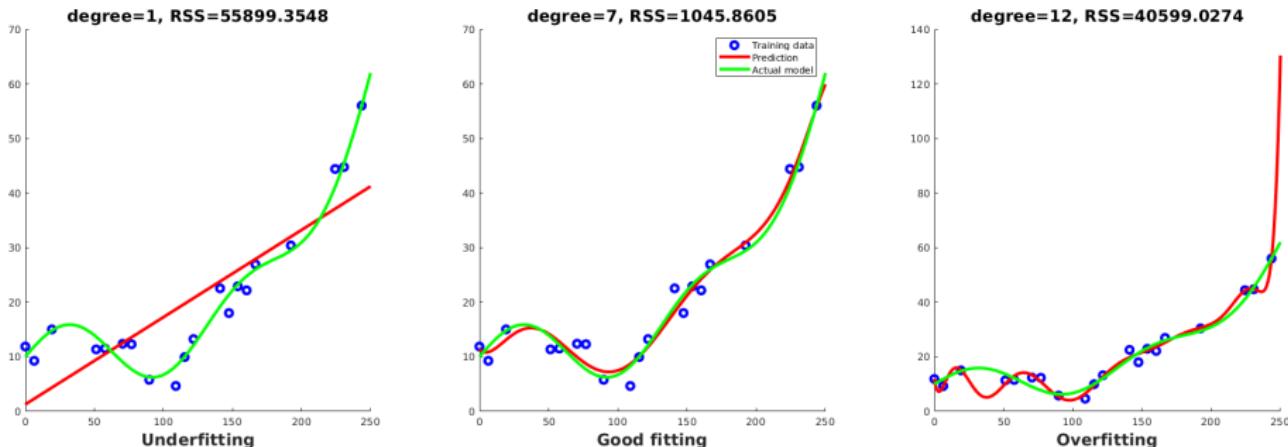


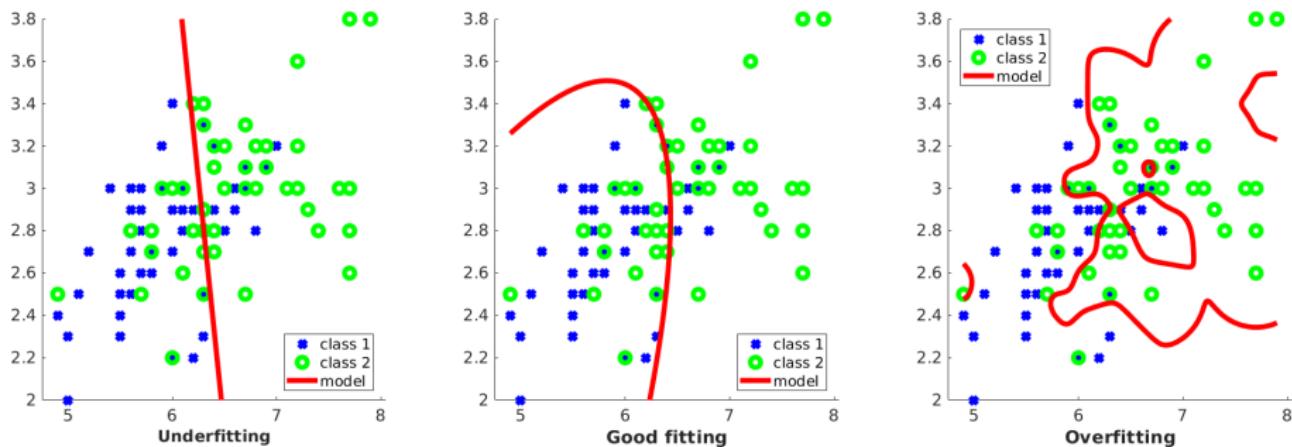
Figure 3: Test and training error as a function of the model complexity. We go from underfitting (left) to overfitting (right).

Bias-variance tradeoff - Overfitting Regression



- Here, we use a polynomial regression model:
$$y = \beta_0 + \beta_1 x + \beta_2 x^2 + \dots + \beta_q x^q$$
- Please note that this is still a linear function in the unknown parameters β !

Bias-variance tradeoff - Overfitting Classification



- Here, we have used LDA, QDA and non-linear SVM

Extending Linear Model

OLS is the best linear unbiased estimator, namely it is the estimator that has the lowest variance with a zero bias. However, the MSE is equal to a sum between *Bias* (squared) and *Variance*. Linear regression may suffer from high variance, So it may be worth sacrificing some bias to achieve a lower variance (**better prediction accuracy**).

Furthermore, we might also have a large number of regression coefficients β which can make **difficult the interpretation** of the results. We often would like to determine a smaller subset of coefficients that exhibit the strongest effect.

We will see two methods:

- Ridge regression
- LASSO

Summary

1 Introduction - Supervised Learning

- Feature pre-processing
- Function space, Risk and Loss functions
- The Bias-Complexity tradeoff

2 Linear models for regression

- Ordinary Least Squares
- Ridge regression
- LASSO

3 Model selection and assessment

- Information criteria: AIC and BIC
- Cross-validation

4 Linear parametric models for classification

- Linear discriminant analysis (LDA)
- Quadratic discriminant analysis (QDA) and Naive Bayes
- Logistic regression

5 Non-parametric models

- K-nearest-neighbour

Ridge regression

Ridge regression shrinks the regression coefficients β by imposing a penalty on their size. We force the coefficients to lie in a ball around the origin with radius \sqrt{t} . We have:

$$\beta_{ridge}^* = \arg \min_{\beta} \underbrace{\sum_{i=1}^n (y_i - x_i^T \beta)^2}_{f(\beta)} \quad \text{subject to } \underbrace{\sum_{j=1}^d \beta_j^2}_{g(\beta)} \leq t \quad (28)$$

From the first KKT equation we have: $\Delta_{\beta} [f(\beta) + \lambda(g(\beta) - t)] = 0$ which is equivalent to solving $\arg \min_{\beta} f(\beta) + \lambda g(\beta)$ for the same choice of λ .

Ridge regression

$$\beta_{ridge}^* = \arg \min_{\beta} \underbrace{\sum_{i=1}^n (y_i - x_i^T \beta)^2}_{Loss} + \lambda \underbrace{\sum_{j=1}^d \beta_j^2}_{Penalty} \quad (29)$$

Here the Lagrange multiplier $\lambda \geq 0$ can be seen as a **tuning parameter**, which controls the strength of the penalty term.

Note that:

- When $\lambda = 0$, what do we have ?

Ridge regression

$$\beta_{ridge}^* = \arg \min_{\beta} \underbrace{\sum_{i=1}^n (y_i - x_i^T \beta)^2}_{Loss} + \lambda \underbrace{\sum_{j=1}^d \beta_j^2}_{Penalty} \quad (29)$$

Here the Lagrange multiplier $\lambda \geq 0$ can be seen as a **tuning parameter**, which controls the strength of the penalty term.

Note that:

- When $\lambda = 0$, what do we have ? OLS solution

Ridge regression

$$\beta_{ridge}^* = \arg \min_{\beta} \underbrace{\sum_{i=1}^n (y_i - x_i^T \beta)^2}_{Loss} + \lambda \underbrace{\sum_{j=1}^d \beta_j^2}_{Penalty} \quad (29)$$

Here the Lagrange multiplier $\lambda \geq 0$ can be seen as a **tuning parameter**, which controls the strength of the penalty term.

Note that:

- When $\lambda = 0$, what do we have ? OLS solution
- When $\lambda = \infty$, what do we have ?

Ridge regression

$$\beta_{ridge}^* = \arg \min_{\beta} \underbrace{\sum_{i=1}^n (y_i - x_i^T \beta)^2}_{Loss} + \lambda \underbrace{\sum_{j=1}^d \beta_j^2}_{Penalty} \quad (29)$$

Here the Lagrange multiplier $\lambda \geq 0$ can be seen as a **tuning parameter**, which controls the strength of the penalty term.

Note that:

- When $\lambda = 0$, what do we have ? OLS solution
- When $\lambda = \infty$, what do we have ? $\beta_{ridge}^* = 0$

Ridge regression

$$\beta_{ridge}^* = \arg \min_{\beta} \underbrace{\sum_{i=1}^n (y_i - x_i^T \beta)^2}_{Loss} + \lambda \underbrace{\sum_{j=1}^d \beta_j^2}_{Penalty} \quad (29)$$

Here the Lagrange multiplier $\lambda \geq 0$ can be seen as a **tuning parameter**, which controls the strength of the penalty term.

Note that:

- When $\lambda = 0$, what do we have ? OLS solution
- When $\lambda = \infty$, what do we have ? $\beta_{ridge}^* = 0$
- We look for the best trade-off. When λ increases from 0 to ∞ the bias also increases whereas the variance decreases

Ridge regression

$$\beta_{ridge}^* = \arg \min_{\beta} \underbrace{\sum_{i=1}^n (y_i - x_i^T \beta)^2}_{Loss} + \lambda \underbrace{\sum_{j=1}^d \beta_j^2}_{Penalty} \quad (30)$$

Note that:

- We do not penalize the intercept β_0 otherwise, adding a constant c to each y_i , it would not result in a shift of the predictions ($x_i^T \beta$) by the same amount c .
- If we center the columns of \mathbf{x} then the OLS intercept β_0 is simply the average of y_i . Thus, we could also center both \mathbf{y} and \mathbf{x} and do not include the intercept in our linear model.
- If the predictor variables x_j (columns of \mathbf{x}) are not on the same scale (same measurement units), we should scale them to have sample variance equal to 1 (very important !)

Ridge regression

- We can rewrite the ridge regression in vector notation:

$$\beta_{ridge}^* = \arg \min_{\beta} \text{RSS} = (\mathbf{y} - \mathbf{x}\beta)^T(\mathbf{y} - \mathbf{x}\beta) + \lambda \|\beta\|_2^2 \quad (31)$$

- It is convex and hence has unique solution:

$$\begin{aligned} \frac{\partial \text{RSS}}{\partial \beta} &= -2\mathbf{x}^T(\mathbf{y} - \mathbf{x}\beta) + 2\lambda\beta = 0 \\ \beta_{ridge}^* &= (\mathbf{x}^T\mathbf{x} + \lambda I_d)^{-1}\mathbf{x}^T\mathbf{y} \end{aligned} \quad (32)$$

- Even if $(\mathbf{x}^T\mathbf{x})$ is singular (not invertible) the addition of λI_d makes it invertible ! (original motivation)

Summary

1 Introduction - Supervised Learning

- Feature pre-processing
- Function space, Risk and Loss functions
- The Bias-Complexity tradeoff

2 Linear models for regression

- Ordinary Least Squares
- Ridge regression
- LASSO

3 Model selection and assessment

- Information criteria: AIC and BIC
- Cross-validation

4 Linear parametric models for classification

- Linear discriminant analysis (LDA)
- Quadratic discriminant analysis (QDA) and Naive Bayes
- Logistic regression

5 Non-parametric models

- K-nearest-neighbour

LASSO

The LASSO (least absolute shrinkage and selection operator) estimate is defined as:

$$\begin{aligned}\beta_{lasso}^* &= \arg \min_{\beta} \sum_{i=1}^n (y_i - x_i^T \beta)^2 \quad \text{subject to } \sum_{j=1}^d |\beta_j| \leq t \\ &= \arg \min_{\beta} \underbrace{\sum_{i=1}^n (y_i - x_i^T \beta)^2}_{\text{Loss}} + \lambda \underbrace{\sum_{j=1}^d |\beta_j|}_{\text{Penalty}}\end{aligned}\tag{33}$$

The only difference between ridge regression and lasso is the penalty. The former uses a L2 penalty whereas the latter a L1 penalty.

The minimization of the L1 penalty **brings some coefficients to zero !**
The L2 penalty decreases the value of all coefficients at the same time !

LASSO

$$\beta_{lasso}^* = \arg \min_{\beta} \underbrace{\sum_{i=1}^n (y_i - x_i^T \beta)^2}_{Loss} + \lambda \underbrace{\sum_{j=1}^d |\beta_j|}_{Penalty} \quad (34)$$

$$\beta_{lasso}^* = \arg \min_{\beta} \underbrace{(\mathbf{y} - \mathbf{x}\beta)^T (\mathbf{y} - \mathbf{x}\beta)}_{Loss} + \lambda \underbrace{\|\beta\|_1}_{Penalty}$$

The LASSO is able to perform **variable selection**. As λ increases more coefficients are set to zero. We keep only the most “important” ones !

In the following courses you will see “black box” methods which can achieve impressive prediction accuracy but which may also be difficult to interpret (i.e. deep learning) !

LASSO

$$\beta_{lasso}^* = \arg \min_{\beta} \underbrace{(\mathbf{y} - \mathbf{x}\beta)^T (\mathbf{y} - \mathbf{x}\beta)}_{Loss} + \lambda \underbrace{\|\beta\|_1}_{Penalty} \quad (35)$$

Note that:

- When $\lambda = 0$, what do we have ?

LASSO

$$\beta_{lasso}^* = \arg \min_{\beta} \underbrace{(\mathbf{y} - \mathbf{x}\beta)^T (\mathbf{y} - \mathbf{x}\beta)}_{Loss} + \lambda \underbrace{\|\beta\|_1}_{Penalty} \quad (35)$$

Note that:

- When $\lambda = 0$, what do we have ? OLS solution

LASSO

$$\beta_{lasso}^* = \arg \min_{\beta} \underbrace{(\mathbf{y} - \mathbf{x}\beta)^T (\mathbf{y} - \mathbf{x}\beta)}_{Loss} + \lambda \underbrace{\|\beta\|_1}_{Penalty} \quad (35)$$

Note that:

- When $\lambda = 0$, what do we have ? OLS solution
- When $\lambda = \infty$, what do we have ?

LASSO

$$\beta_{lasso}^* = \arg \min_{\beta} \underbrace{(\mathbf{y} - \mathbf{x}\beta)^T (\mathbf{y} - \mathbf{x}\beta)}_{Loss} + \lambda \underbrace{\|\beta\|_1}_{Penalty} \quad (35)$$

Note that:

- When $\lambda = 0$, what do we have ? OLS solution
- When $\lambda = \infty$, what do we have ? $\beta_{lasso}^* = 0$

LASSO

$$\beta_{lasso}^* = \arg \min_{\beta} \underbrace{(\mathbf{y} - \mathbf{x}\beta)^T (\mathbf{y} - \mathbf{x}\beta)}_{Loss} + \lambda \underbrace{\|\beta\|_1}_{Penalty} \quad (35)$$

Note that:

- When $\lambda = 0$, what do we have ? OLS solution
- When $\lambda = \infty$, what do we have ? $\beta_{lasso}^* = 0$
- The difference between ridge and lasso is “in between”, that is to say how the coefficients β go to zero when varying λ .

LASSO

$$\beta_{lasso}^* = \arg \min_{\beta} \underbrace{(\mathbf{y} - \mathbf{x}\beta)^T (\mathbf{y} - \mathbf{x}\beta)}_{Loss} + \lambda \underbrace{\|\beta\|_1}_{Penalty} \quad (36)$$

Note that:

- Unlike Ridge regression, Lasso does not have a closed form solution
- Different algorithms have been proposed such as: quadratic programming ($\mathcal{O}(n2^d)$), LARS (least angle regression) ($\mathcal{O}(nd^2)$) and coordinate descent ($\mathcal{O}(nd)$)

LASSO

Note that, if $\mathbf{x}^T \mathbf{x} = I_d$, then:

$$\begin{aligned}\beta_{lasso}^* &= \arg \min_{\beta} \text{RSS} = (\mathbf{y} - \mathbf{x}\beta)^T(\mathbf{y} - \mathbf{x}\beta) + \lambda||\beta||_1 \\ \beta_{lasso}^* &= \arg \min_{\beta} -2\beta^T \mathbf{x}^T \mathbf{y} + \beta^T \beta + \lambda \sum_{j=1}^d |\beta_j| \\ \beta_{lasso}^* &= \arg \min_{\beta} -2\beta^T \beta_{OLS}^* + \beta^T \beta + \lambda \sum_{j=1}^d |\beta_j| \\ \beta_{lasso}^* &= \arg \min_{\beta} \sum_{j=1}^d \left(-2\beta_j(\beta_{OLS}^*)_j + \beta_j^2 + \lambda|\beta_j| \right)\end{aligned}\tag{37}$$

LASSO

We can work with each index j separately (we remove the index in the following for clarity purpose):

$$\beta_{lasso}^* = \arg \min_{\beta} -2\beta\beta_{OLS}^* + \beta^2 + \lambda|\beta|$$

$$\beta_{lasso}^* = \arg \min_{\beta} \begin{cases} \text{if } \beta \geq 0 & -2\beta\beta_{OLS}^* + \beta^2 + \lambda\beta \\ \text{if } \beta < 0 & -2\beta\beta_{OLS}^* + \beta^2 - \lambda\beta \end{cases}$$

$$\beta_{lasso}^* = \begin{cases} \text{if } \beta \geq 0 & \beta_{OLS}^* - \frac{\lambda}{2} \\ \text{if } \beta < 0 & \beta_{OLS}^* + \frac{\lambda}{2} \end{cases} \quad (38)$$

$$\beta_{lasso}^* = \begin{cases} \beta_{OLS}^* - \frac{\lambda}{2} & \text{if } \beta_{OLS}^* \geq \frac{\lambda}{2} \\ \beta_{OLS}^* + \frac{\lambda}{2} & \text{if } \beta_{OLS}^* < -\frac{\lambda}{2} \\ 0 & \text{otherwise} \end{cases}$$

Ridge Regression Vs LASSO

Using $\mathbf{x}^T \mathbf{x} = I_d$, we obtain:

$$\beta_{ridge}^* = \frac{\beta_{OLS}^*}{1 + \lambda}$$
$$\beta_{lasso}^* = \begin{cases} \beta_{OLS}^* - \frac{\lambda}{2} & \text{if } \beta_{OLS}^* \geq \frac{\lambda}{2} \\ \beta_{OLS}^* + \frac{\lambda}{2} & \text{if } \beta_{OLS}^* < -\frac{\lambda}{2} \\ 0 & \text{otherwise} \end{cases} \quad (39)$$

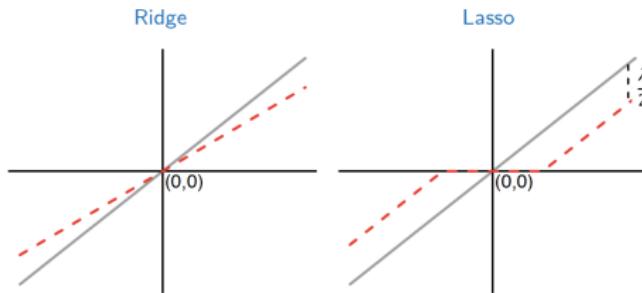


Figure 4: β^* and β_{OLS}^* are shown in the y-axis and x-axis respectively. The gray line shows β_{OLS}^* . The red lines represent the Ridge and Lasso estimators.

Ridge Regression Vs LASSO

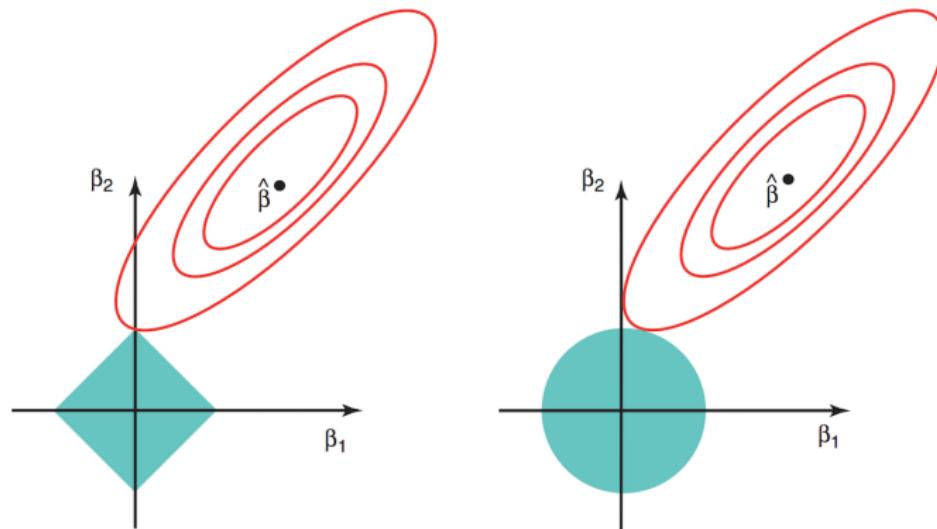


Figure 5: LASSO on the left and Ridge on the right. We suppose we have two β : β_1 and β_2 . In red are shown the elliptical contours of the OLS RSS ($\hat{\beta} = \beta_{OLS}^*$). In blue are shown the constraint regions: $|\beta_1| + |\beta_2| \leq t$ for LASSO and $\beta_1^2 + \beta_2^2 \leq t$ for Ridge.

Ridge Regression Vs LASSO

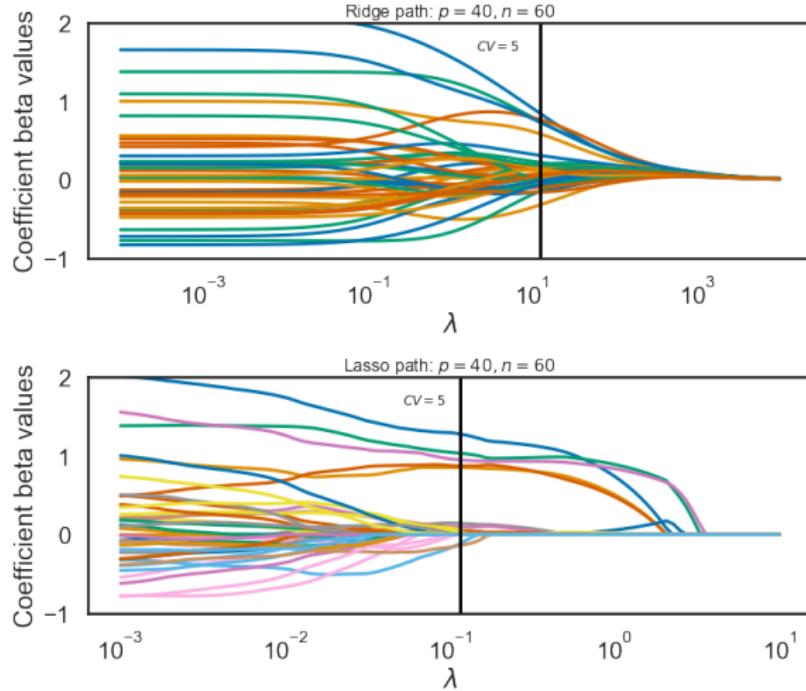


Figure 6: Trends of β when varying λ for Ridge and Lasso. In Ridge all β tends to 0 in the same way. In Lasso β go to 0 at different times.

Degrees of freedom

- The number of degrees of freedom df can be seen as the number of linearly independent parameters. However, what does it mean in Ridge and Lasso ?
- A more useful definition is: $df(\mathbf{y}^*) = \frac{1}{\sigma^2} \sum_{i=1}^N \text{Cov}(y_i^*, y_i)$ which means that the harder we fit to the data, the larger $\text{Cov}(y_i^*, y_i)$ and hence $df(\mathbf{y}^*)$ [1]
- $df(\mathbf{y}_{OLS}^*) = d$
- $df(\mathbf{y}_{ridge}^*) = \text{Tr}[\mathbf{x}(\mathbf{x}^T \mathbf{x} + \lambda I_d)^{-1} \mathbf{x}^T]$
- $df(\mathbf{y}_{lasso}^*)$ = number of predictors (non-zero coefficients in β_{lasso}^*)

Summary

1 Introduction - Supervised Learning

- Feature pre-processing
- Function space, Risk and Loss functions
- The Bias-Complexity tradeoff

2 Linear models for regression

- Ordinary Least Squares
- Ridge regression
- LASSO

3 Model selection and assessment

- Information criteria: AIC and BIC
- Cross-validation

4 Linear parametric models for classification

- Linear discriminant analysis (LDA)
- Quadratic discriminant analysis (QDA) and Naive Bayes
- Logistic regression

5 Non-parametric models

- K-nearest-neighbour

Model assessment

- If we have no hyper-parameters (like in OLS) to fix, namely we only have one model f , we do not need a Validation set. We only need a Training set to fit the model (i.e. estimate the best parameters) and a Test set to compute the prediction error (generalization error). The test set is used for **Model assessment**.

Recipe:

- Randomize and, if necessary, standardize/normalize the data \mathcal{D}_n
- Split \mathcal{D}_n into, for instance, 70% for Training set and 30% for Test set
- Choose a class of models \mathcal{F} without hyper-parameters, like OLS
- Fit the model $f \in \mathcal{F}$ to the Training set (i.e. estimate the best parameters β^*)
- Evaluate the generalization error of the model f using the Test set

Model selection and assessment

- If we do have hyper-parameters, like the λ in Ridge and LASSO, we also need a Validation set for **Model selection**. Each value of λ corresponds to a different model f_i .
 - Randomize and, if necessary, standardize/normalize the data \mathcal{D}_n
 - Split \mathcal{D}_n into, for instance, 70% for Training set, 20% for Validation set and 10% for Test set
 - Choose a class of models \mathcal{F} with one hyper-parameter λ , like Ridge or LASSO
 - Choose a set of values L for the hyper-parameter $\{\lambda_l\}_{l=1,\dots,L}$
 - **Loop** ($l = 1$ to L)
 - Fit the model f_{λ_l} to the Training set (i.e. estimate the best parameters β^* using λ_l)
 - Evaluate the performance of the model f_{λ_l} using the Validation set
 - Select as best model f^* the one, among $\{f_{\lambda_l}\}$, with the best performance on the Validation set (i.e. lowest prediction error)
 - Evaluate the generalization error of the model f^* using the Test set

Model selection

- If the data-set \mathcal{D}_n is very rich (i.e. N is big), we can randomly divide it into three subsets (Training, Validation and Test). But what if \mathcal{D}_n is small ?
- We need to find a way to estimate the best model \hat{f} with few data.
In the case of Ridge and Lasso this means finding the best hyper-parameter λ
- Two possible ways are:
 - Information criteria: AIC and BIC
 - Cross-validation

Summary

1 Introduction - Supervised Learning

- Feature pre-processing
- Function space, Risk and Loss functions
- The Bias-Complexity tradeoff

2 Linear models for regression

- Ordinary Least Squares
- Ridge regression
- LASSO

3 Model selection and assessment

- Information criteria: AIC and BIC
- Cross-validation

4 Linear parametric models for classification

- Linear discriminant analysis (LDA)
- Quadratic discriminant analysis (QDA) and Naive Bayes
- Logistic regression

5 Non-parametric models

- K-nearest-neighbour

Information criteria: AIC

- We can use an information criterion (or more than one) to select the “best” model \hat{f}
- The model \hat{f} should not be too much complex (high df) in order not to over-fit the data but at the same time it should not be too simple (low df) in order not to under-fit the data !
- The **Akaike information criterion (AIC)** finds the best model \hat{f} among a collection of models $\{f_i\}$ by balancing the goodness of fit with the complexity of the model: $AIC_{f_i} = 2k_{f_i} - 2 \ln(\hat{L}_{f_i})$
 - k_{f_i} is the number of effective parameters of the model f_i (i.e. degrees of freedom)
 - \hat{L}_{f_i} is the maximum likelihood estimate of the model f_i
- A further refinement to correct for small data-sets is:
$$AICc_{f_i} = AIC + \frac{2k(k+1)}{N-k-1}$$
 where N is the number of samples
- The best model \hat{f} is the one with the lowest **AIC** or **AICc**

Information criteria: BIC

- Another criterion for model selection is the **Bayesian information criterion (BIC)**: $BIC_{f_i} = \ln(N)k_{f_i} - 2\ln(\hat{L}_{f_i})$
- The only difference between the two criteria is the penalty for the number of effective parameters ($2k_{f_i}$ Vs $\ln(N)k_{f_i}$) → BIC penalizes model complexity more heavily !
- Main differences:
 - BIC will select the “true optimal model” (if it is present in the list of candidates) with probability 1 when $N \rightarrow \infty$. The probability for AIC/AICc can be less than 1.
 - If the “true optimal model” is not present in the list of candidates and/or if N is small, AIC and AICc may select a better model than BIC

Information criteria - OLS

- If all residuals are i.i.d. normally distributed (with zero mean) $\epsilon_i \sim \mathcal{N}(0, \sigma^2)$ then the log-likelihood is equal to
$$\ln(L_{f_i}) = \ln \left(\prod_{i=1}^N \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(\frac{-\epsilon_i^2}{2\sigma^2}\right) \right) = -\frac{N}{2} \ln(2\pi) - \frac{N}{2} \ln(\sigma^2) - \frac{RSS}{2\sigma^2}$$
- The maximum likelihood estimate for σ^2 is equal to: $\sigma^2 = \frac{RSS}{N}$
- Thus, $\ln(\hat{L}_{f_i}) = -\frac{N}{2} \ln(RSS) + C$ where C is a constant which depends on the data (so it's the same for all models f_i)
- Thus we obtain:
 - $AIC_{f_i} = 2k_{f_i} + N \ln(RSS)$
 - $BIC_{f_i} = \ln(N)k_{f_i} + N \ln(RSS)$

Information criteria

Pros

- Easy to interpret
- Fast to compute

Cons

- Need proper estimation of degrees of freedom (may be difficult like in Lasso)
- Asymptotic results (true only when $N \rightarrow \infty$)
- Assume that data are actually generated by the model f_i
- Tend to have bad results when $k > N$

Summary

1 Introduction - Supervised Learning

- Feature pre-processing
- Function space, Risk and Loss functions
- The Bias-Complexity tradeoff

2 Linear models for regression

- Ordinary Least Squares
- Ridge regression
- LASSO

3 Model selection and assessment

- Information criteria: AIC and BIC
- Cross-validation

4 Linear parametric models for classification

- Linear discriminant analysis (LDA)
- Quadratic discriminant analysis (QDA) and Naive Bayes
- Logistic regression

5 Non-parametric models

- K-nearest-neighbour

Prediction error

- Given a data-set \mathcal{D}_n divided into Training set \mathcal{T} and Test set $((X_T, Y_T))$, what we actually measure is the **empirical generalization error** on the Test set given \mathcal{T} , which is defined as
$$\text{Err}_{\mathcal{T}} = E[L(Y_T, \hat{f}(X_T))|\mathcal{T}]$$
- For statistical analysis, it would be better suited the **expected empirical generalization error** defined as
$$\text{Err} = E[L(Y, \hat{f}(X))] = E[\text{Err}_{\mathcal{T}}]$$
 where we would average over all possible Training sets \mathcal{T} . This means that we should have many data-sets \mathcal{D}_n sampled from the same theoretical distribution, divided into Training and Test sets.

Prediction error

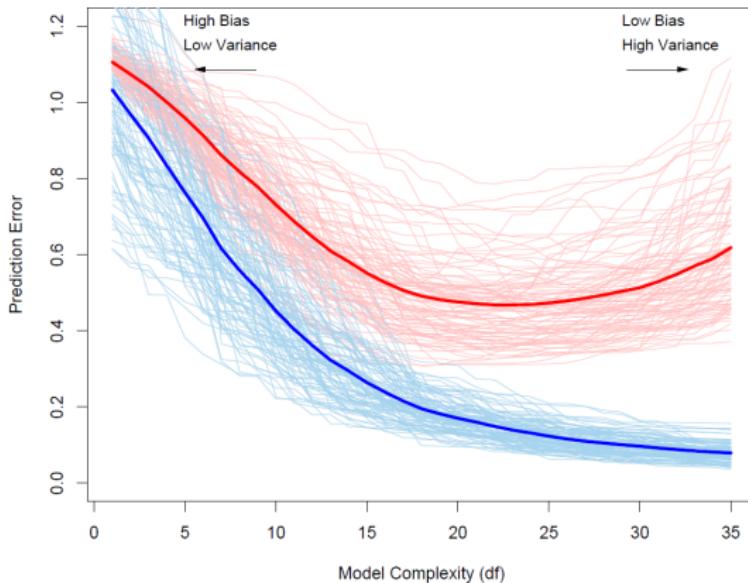


Figure 7: Behavior of training and test error as the model complexity is varied. Light curves refer to the errors given a training set. Solid curves show the averages error. Image taken from [1]

Cross-validation (CV)

- Unfortunately, we usually have only one data-set set $\mathcal{D}_n \rightarrow$ the solution of cross-validation is to divide the data-set into Training and Test sets not only once but several times, combining then (i.e. averaging) the test performances. This should reduce the variability of the generalization error.

Cross-validation (CV)

- Unfortunately, we usually have only one data-set set $\mathcal{D}_n \rightarrow$ the solution of cross-validation is to divide the data-set into Training and Test sets not only once but several times, combining then (i.e. averaging) the test performances. This should reduce the variability of the generalization error.
- **Exhaustive cross-validation:** It uses all possible splits into Training and Test sets of the original data $\mathcal{D}_n \rightarrow$ usually computationally infeasible even for small n

Cross-validation (CV)

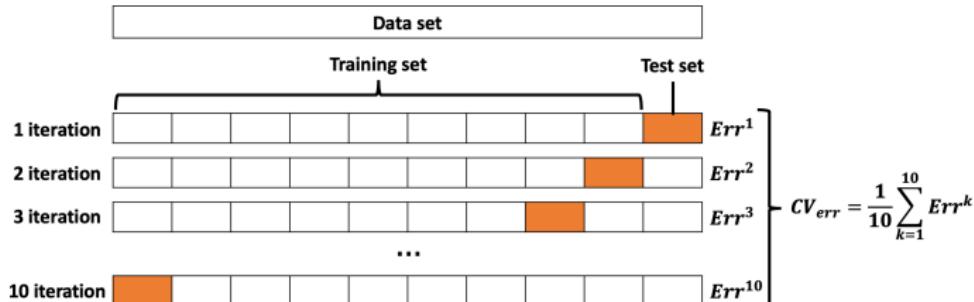
- Unfortunately, we usually have only one data-set set $\mathcal{D}_n \rightarrow$ the solution of cross-validation is to divide the data-set into Training and Test sets not only once but several times, combining then (i.e. averaging) the test performances. This should reduce the variability of the generalization error.
- **Exhaustive cross-validation:** It uses all possible splits into Training and Test sets of the original data $\mathcal{D}_n \rightarrow$ usually computationally infeasible even for small n
- **K-fold cross-validation:** \mathcal{D}_n is randomly partitioned into K equal sized parts or “folds”. We then train on all but the k-th part which is used as Test set. We then iterate over all K parts and average the test error over all folds. All observations are used for both training and testing. It is computationally feasible.

Cross-validation (CV)

- Unfortunately, we usually have only one data-set set $\mathcal{D}_n \rightarrow$ the solution of cross-validation is to divide the data-set into Training and Test sets not only once but several times, combining then (i.e. averaging) the test performances. This should reduce the variability of the generalization error.
- **Exhaustive cross-validation:** It uses all possible splits into Training and Test sets of the original data $\mathcal{D}_n \rightarrow$ usually computationally infeasible even for small n
- **K-fold cross-validation:** \mathcal{D}_n is randomly partitioned into K equal sized parts or “folds”. We then train on all but the k-th part which is used as Test set. We then iterate over all K parts and average the test error over all folds. All observations are used for both training and testing. It is computationally feasible.
- In classification, we usually use **stratified K-fold cross-validation** where each fold contains roughly the same proportion of observations from the different classes

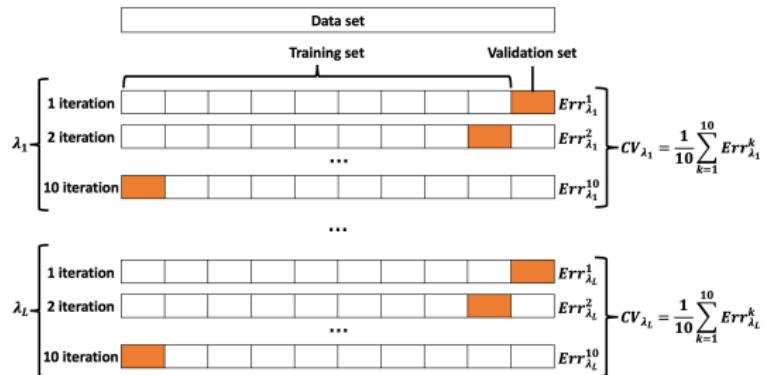
K-fold cross validation - Model Assessment

- Split the training N samples into K disjoint sets $\{S_k\}_{k=1,\dots,K}$ of size N/K
- Loop ($k = 1$ to K):**
 - Fit the model f^k using all samples $\notin S_k$, that is estimate the optimal parameters of the model without the k -th part
 - Record the error on the k -th (test) fold $\text{Err}^k = \sum_{i \in S_k} L(y_i, f^k(x_i))$
- Compute the average error over all K folds: $\text{CV}_{\text{err}} = \frac{1}{N} \sum_{k=1}^K \text{Err}^k$



K-fold cross validation - Model Validation

- Given a set of tuning hyper-parameters $\{\lambda_l\}_{l=1,\dots,L}$, split the training N samples into K disjoint sets $\{S_k\}_{k=1,\dots,K}$ of size N/K
- Loop ($l = 1$ to L):**
 - Loop ($k = 1$ to K):**
 - Fit the model $f_{\lambda_l}^k$ using all samples $\notin S_k$, that is estimate the optimal parameters of the model without the k -th part and using λ_l
 - Record the error on the validation set $\text{Err}_{\lambda_l}^k = \sum_{i \in S_k} L(y_i, f_{\lambda_l}^k(x_i))$
 - Compute the average error over all K folds: $CV_{\lambda_l} = \frac{1}{N} \sum_{k=1}^K \text{Err}_{\lambda_l}^k$



K-fold cross validation - Model Validation

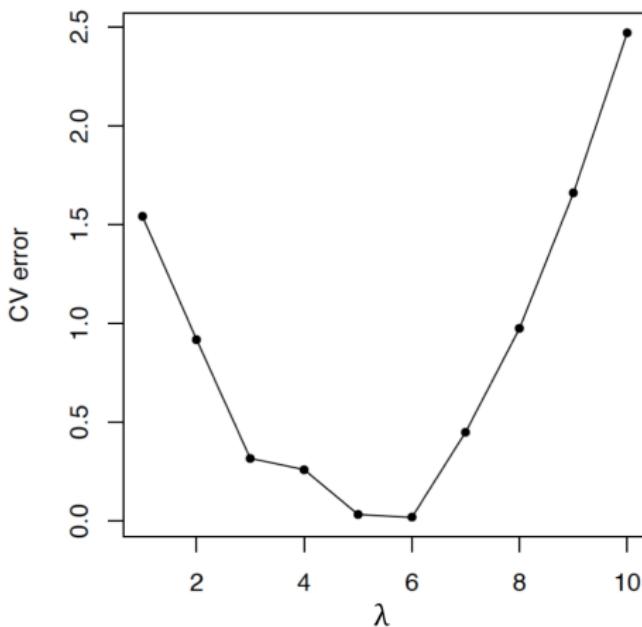


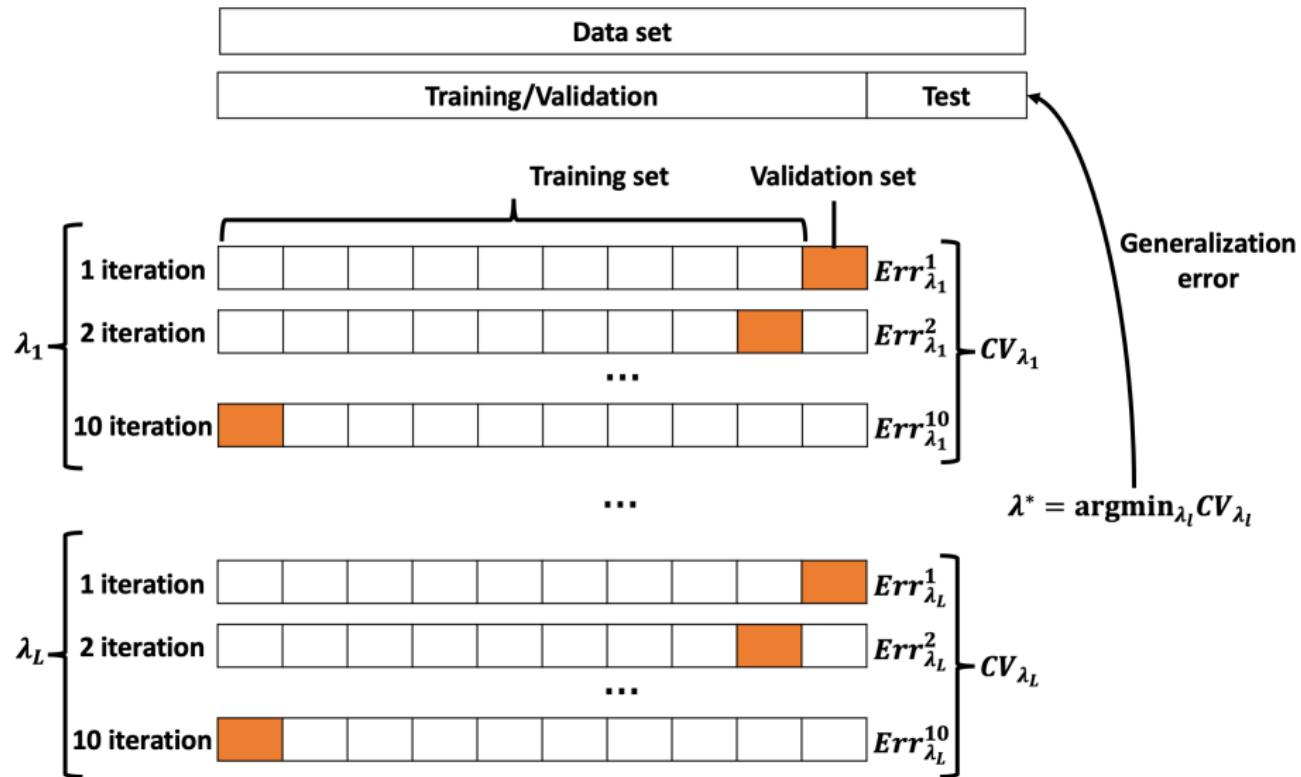
Figure 8: Curve of the CV error which is a function of λ . Image taken from [1]

We then choose the best λ as: $\lambda^* = \arg \min_{\lambda_l} CV_{\lambda_l}$

K-fold cross validation - Model Validation and Assessment

- Divide \mathcal{D}_n into two parts, one for training/validation and one for test
- Given a set of tuning hyper-parameters $\{\lambda_l\}_{l=1,\dots,L}$, split the training/validation samples into K disjoint sets $\{S_k\}_{k=1,\dots,K}$
- **Loop** ($l = 1$ to L):
 - **Loop** ($k = 1$ to K):
 - Fit the model $f_{\lambda_l}^k$ using all samples $\notin S_k$, that is estimate the optimal parameters of the model without the k -th part and using λ_l
 - Record the error on the validation set $\text{Err}_{\lambda_l}^k = \sum_{i \in S_k} L(y_i, f_{\lambda_l}^k(x_i))$
 - Compute the average error over all K folds: $CV_{\lambda_l} = \frac{1}{N} \sum_{k=1}^K \text{Err}_{\lambda_l}^k$
- Choose as best model f_{λ^*} the one with the smallest error CV_{λ_l}
- Recompute the parameters of the best model over all $K - 1$ sets
- Compute the generalization error on the test set
$$\text{Err}_{\lambda^*} = \sum_{i \in \text{Test}} L(y_i, f_{\lambda^*}(x_i))$$

K-fold cross validation - Model Validation and Assessment



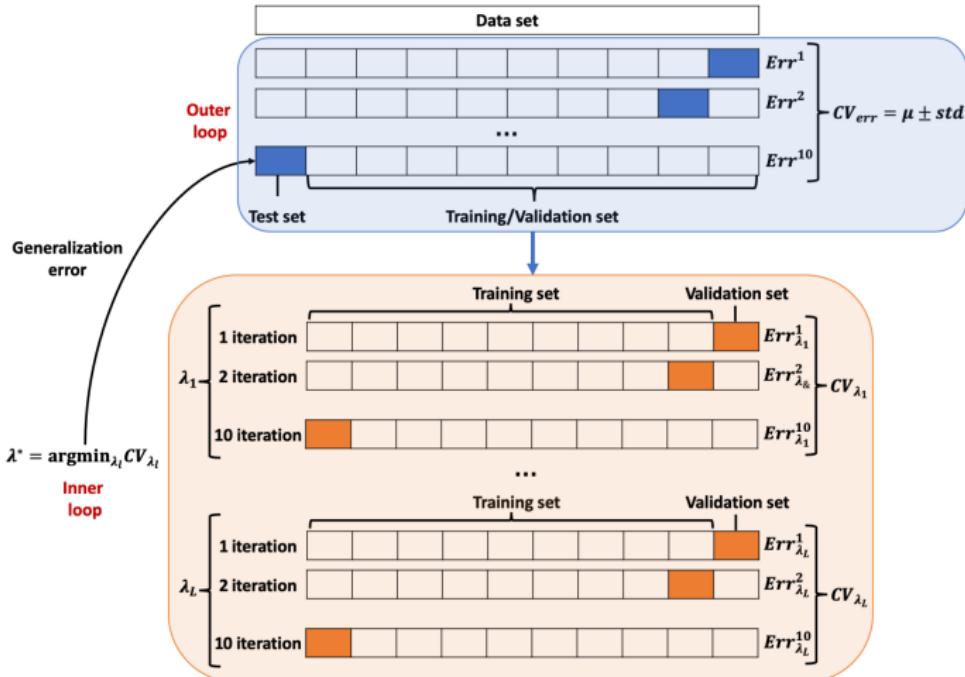
K-fold cross validation

- Be careful, once estimated the best tuning parameter λ^* , we fit our model to the **entire training/validation set** ! We can then use the estimated best parameters β^* to compute the generalization error in the test set
- The choice of K is important
 - $K = 2$: high bias since we will train on only half of the data
 - $K = N$ (leave-one-out): high variance (big overlap between the training sets of each fold)
 - $K = 5, 10$: seems to be a good trade-off between bias and variance (no theory behind)
- Using the previous technique, there was only one test set → **nested cross-validation** (more computationally demanding)

k^*t -fold nested cross-validation

- Split all the available n samples into K disjoint sets S_i of size n/K
- Here each model corresponds to a different value of the hyper-parameter λ
 - **Outer loop** ($i=1$ to K):
 - For every i , use S_i as test set and all the other $K - 1$ sets as training + validation set
 - Divide the training + validation set into t disjoint sets S_j of equal size
 - **Inner loop** ($j=1$ to t):
 - For every j , train the model in $t - 1$ sets (training set) and evaluate it (validation set) in the left out j -th set
 - Choose the model with the best average performance over all t runs
 - Recompute the parameters of the best model over all $K - 1$ sets
 - Compute the error in the left-out S_i set
 - Compute the average and standard deviation of the CV error as the mean \pm std of the errors over all K runs.

k^*t -fold nested cross-validation



Implementation using scikit-learn: https://scikit-learn.org/stable/auto_examples/model_selection/plot_nested_cross_validation_iris.html

Summary

1 Introduction - Supervised Learning

- Feature pre-processing
- Function space, Risk and Loss functions
- The Bias-Complexity tradeoff

2 Linear models for regression

- Ordinary Least Squares
- Ridge regression
- LASSO

3 Model selection and assessment

- Information criteria: AIC and BIC
- Cross-validation

4 Linear parametric models for classification

- Linear discriminant analysis (LDA)
- Quadratic discriminant analysis (QDA) and Naive Bayes
- Logistic regression

5 Non-parametric models

- K-nearest-neighbour

Summary

1 Introduction - Supervised Learning

- Feature pre-processing
- Function space, Risk and Loss functions
- The Bias-Complexity tradeoff

2 Linear models for regression

- Ordinary Least Squares
- Ridge regression
- LASSO

3 Model selection and assessment

- Information criteria: AIC and BIC
- Cross-validation

4 Linear parametric models for classification

- **Linear discriminant analysis (LDA)**
- Quadratic discriminant analysis (QDA) and Naive Bayes
- Logistic regression

5 Non-parametric models

- K-nearest-neighbour

Linear discriminant analysis (LDA)

Remember that in a classification task, with a 0-1 loss, the optimal procedure is to estimate the most probable class of a test sample x_j (i.e. maximize the posterior probability):

$$f^*(x_j) = \arg \max_{f \in \mathcal{Y}} P(Y = f(x_j) | X = x_j) = \arg \max_{C_k} P(X = x_j | Y = C_k) \pi_{C_k} \quad (40)$$

where we have used the Bayes' theorem ($P(A|B) \propto P(B|A)P(A)$).

LDA proposes to model all conditional distributions

$P(X = x_j | Y = C_k) = P_k(x)$ as Gaussian distributions $P_k(x) \sim \mathcal{N}(\mu_k, \Sigma)$.

Classes have **different means** μ_k but the **same covariance matrix** Σ .

Recall that: $\mathcal{N}(\mu_k, \Sigma) \propto \frac{1}{|\Sigma|^{1/2}} \exp(-\frac{1}{2}(x - \mu_k)^T \Sigma^{-1} (x - \mu_k))$

Linear discriminant analysis (LDA)

We can notice that:

$$\begin{aligned} f^*(x_j) &= \arg \max_{C_k} P_{C_k}(x_j) \pi_{C_k} = \arg \min_{C_k} -2 \log[P_{C_k}(x_j) \pi_{C_k}] \\ &\propto \arg \min_{C_k} (x - \mu_k)^T \Sigma^{-1} (x - \mu_k) + \log(|\Sigma|) - 2 \log(\pi_{C_k}) \\ &= \arg \min_{C_k} -2x^T \Sigma^{-1} \mu_k + \mu_k^T \Sigma^{-1} \mu_k - 2 \log(\pi_{C_k}) \tag{41} \\ &= \arg \min_{C_k} \underbrace{\mu_k^T \Sigma^{-1} \mu_k}_{a_k} - 2 \log(\pi_{C_k}) + x^T \underbrace{-2 \Sigma^{-1} \mu_k}_{b_k} \end{aligned}$$

This is a linear discriminant function !

Linear discriminant analysis (LDA)

In order to estimate μ_k and Σ , we can maximize the log-likelihood of the training set (x_i, y_i) :

$$l(\mu_1, \dots, \mu_K) = - \sum_{k=1}^K \sum_{i:y_i=k} \frac{1}{2} (x_i - \mu_k)^T \Sigma^{-1} (x_i - \mu_k) - \frac{1}{2} n \log(|\Sigma|) \quad (42)$$

Let $n_k = \#\{i : y_i = k\}$ be the number of observations in class k . The log-likelihood is maximized by:

$$\mu_k^* = \frac{1}{n_k} \sum_{i:y_i=k} x_i \quad \Sigma^* = \frac{1}{n} \sum_{k=1}^K \sum_{i:y_i=k} (x_i - \mu_k^*)(x_i - \mu_k^*)^T \quad (43)$$

The a priori probabilities π_{C_k} are estimated as the proportion of samples of class k : $\pi_{C_k}^* = \#\{i : y_i = k\}/n$

Linear discriminant analysis (LDA)

Thus, the best classifier under the assumption that $P_k(x) \sim \mathcal{N}(\mu_k, \Sigma)$ is given by:

$$f^*(x_j) \arg \min_{C_k} -2x_j^T (\Sigma^*)^{-1} \mu_k^* + (\mu_k^*)^T (\Sigma^*)^{-1} \mu_k^* - 2 \log(\pi_{C_k}^*) \quad (44)$$

for every sample x_j . We can also notice that if we have two classes (0 and 1) their *linear* (in x_j) decision boundary is given by their log-ratio:

$$\begin{aligned} \log \frac{P(Y = C_0 | X = x_j)}{P(Y = C_1 | X = x_j)} &= \log \frac{\pi_0}{\pi_1} - \frac{1}{2} (\mu_0 + \mu_1)^T \Sigma^{-1} (\mu_0 - \mu_1) + \\ &\quad + x_j^T \Sigma^{-1} (\mu_0 - \mu_1) = a + x^T b = 0 \end{aligned} \quad (45)$$

Linear discriminant analysis (LDA)

Pros

- Easy to interpret
- Fast to compute
- Optimal if Gaussian, homoscedasticity and linear assumptions are met

Cons

- Covariance matrix inversion might be ill-posed if $N \ll p$ (need of regularization)
- Sensitive to outliers (observations far from the boundary have all the same influence)
- Predictive power can decrease if variables are correlated

Summary

1 Introduction - Supervised Learning

- Feature pre-processing
- Function space, Risk and Loss functions
- The Bias-Complexity tradeoff

2 Linear models for regression

- Ordinary Least Squares
- Ridge regression
- LASSO

3 Model selection and assessment

- Information criteria: AIC and BIC
- Cross-validation

4 Linear parametric models for classification

- Linear discriminant analysis (LDA)
- Quadratic discriminant analysis (QDA) and Naive Bayes
- Logistic regression

5 Non-parametric models

- K-nearest-neighbour

QDA and Naive Bayes

QDA

- In LDA we assume that classes have the same covariance matrix Σ .
In QDA, each class has its own covariance matrix Σ_k .
- More flexible but also more parameters to compute

Naive Bayes

- In Naive Bayes, we also assume that all features are independent (given a certain class) which means that each class has its own *diagonal* covariance matrix Σ_k .
- Faster and easier to compute than QDA
- Be careful: features should not be correlated !

In both cases, the discriminant functions are **quadratic** !

QDA and Naive Bayes

$$\begin{aligned} f^*(x_j) &= \arg \min_{C_k} (x - \mu_k)^T \Sigma_k^{-1} (x - \mu_k) + \log(|\Sigma_k|) - 2 \log(\pi_{C_k}) \\ &= \arg \min_{C_k} x^T \Sigma_k^{-1} x - 2x^T \Sigma_k^{-1} \mu_k + \mu_k^T \Sigma_k^{-1} \mu_k - 2 \log(\pi_{C_k}) \quad (46) \\ &= \arg \min_{C_k} x^T \underbrace{\frac{A_k}{\Sigma_k^{-1}} x}_{-\frac{b_k}{2\Sigma_k^{-1}\mu_k}} + \underbrace{\frac{c_k}{\mu_k^T \Sigma_k^{-1} \mu_k - 2 \log(\pi_{C_k})}} \end{aligned}$$

This is a quadratic discriminant function !

Summary

1 Introduction - Supervised Learning

- Feature pre-processing
- Function space, Risk and Loss functions
- The Bias-Complexity tradeoff

2 Linear models for regression

- Ordinary Least Squares
- Ridge regression
- LASSO

3 Model selection and assessment

- Information criteria: AIC and BIC
- Cross-validation

4 Linear parametric models for classification

- Linear discriminant analysis (LDA)
- Quadratic discriminant analysis (QDA) and Naive Bayes
- Logistic regression

5 Non-parametric models

- K-nearest-neighbour

Logistic regression

- Can we use a linear regression model $y = x^T \beta$ to predict categorical variables (i.e. for classification in $\mathcal{Y} = \{C_1, \dots, C_K\}$) ?

Logistic regression

- Can we use a linear regression model $y = x^T \beta$ to predict categorical variables (i.e. for classification in $\mathcal{Y} = \{C_1, \dots, C_K\}$) ?
- No, since the predicted outcome y might assume all intermediate values and not necessarily $\mathcal{Y} = \{C_1, \dots, C_K\}$

Logistic regression

- Can we use a linear regression model $y = x^T \beta$ to predict categorical variables (i.e. for classification in $\mathcal{Y} = \{C_1, \dots, C_K\}$) ?
- No, since the predicted outcome y might assume all intermediate values and not necessarily $\mathcal{Y} = \{C_1, \dots, C_K\}$
- Instead of predicting directly y , we could predict the probability of y being equal to a certain class knowing x (i.e. $P(Y = C_k | X = x)$). Is linear regression suited for predicting conditional probabilities ?

Logistic regression

- Can we use a linear regression model $y = x^T \beta$ to predict categorical variables (i.e. for classification in $\mathcal{Y} = \{C_1, \dots, C_K\}$) ?
- No, since the predicted outcome y might assume all intermediate values and not necessarily $\mathcal{Y} = \{C_1, \dots, C_K\}$
- Instead of predicting directly y , we could predict the probability of y being equal to a certain class knowing x (i.e. $P(Y = C_k | X = x)$). Is linear regression suited for predicting conditional probabilities ?
- Again no, predicted values are usually unbounded → we have to force the outcome to be in the unit interval $[0, 1]$

Logistic regression

- Can we use a linear regression model $y = x^T \beta$ to predict categorical variables (i.e. for classification in $\mathcal{Y} = \{C_1, \dots, C_K\}$) ?
- No, since the predicted outcome y might assume all intermediate values and not necessarily $\mathcal{Y} = \{C_1, \dots, C_K\}$
- Instead of predicting directly y , we could predict the probability of y being equal to a certain class knowing x (i.e. $P(Y = C_k | X = x)$). Is linear regression suited for predicting conditional probabilities ?
- Again no, predicted values are usually unbounded → we have to force the outcome to be in the unit interval $[0, 1]$
- Furthermore, we want that $\sum_k P(Y = C_k | X = x) = 1$

Logistic regression

- Logistic regression uses the logistic function $l(x) = \frac{e^x}{1+e^x} = \frac{1}{1+e^{-x}}$ to bound the output between $[0, 1]$. Since we have K possible classes, we model each conditional probability of the classes as:

$$\begin{aligned} P(Y = C_k | X = x; \theta) &= \frac{\exp(x^T \beta_k)}{1 + \sum_{l=1}^{K-1} \exp(x^T \beta_l)} \quad k = 1, \dots, K-1 \\ P(Y = C_K | X = x; \theta) &= \frac{1}{1 + \sum_{l=1}^{K-1} \exp(x^T \beta_l)} \end{aligned} \tag{47}$$

- where we have chosen the last class K as reference one in order to have a sum equal to 1. This choice is arbitrary in that the estimates are equivariant wrt this choice
- θ is the entire set of parameters $\{\beta_k\}_{k=1}^{K-1}$

Logistic regression

- **Binomial** or binary logistic regression: y can only have two values ($K = 2$). It can be seen as a series of iid Bernoulli trials.
- **Multinomial** logistic regression: y can have three or more values not ordered ($K > 2$)

In the following, we will discuss in detail the two-class $\{0, 1\}$ case where we can write that $P(Y = 1|X = x; \beta) = p(x; \beta)$ and $P(Y = 0|X = x; \beta) = 1 - p(x; \beta)$. It follows that:

$$p(x; \beta) = \frac{\exp(x^T \beta)}{1 + \exp(x^T \beta)} \quad (48)$$

We can also compute the log-odds, namely the log of the likelihood of the first class with respect to the second one (resemblance with LDA...):

$$\log \frac{p(x; \beta)}{1 - p(x; \beta)} = x^T \beta \quad (49)$$

Logistic regression

$$\log \frac{p(x; \beta)}{1 - p(x; \beta)} = x^T \beta \quad (50)$$

- The **linear** decision boundary is thus $x^T \beta = 0$
- We should predict $Y = 1$ when $p(x; \beta) > (1 - p(x; \beta))$, which means when $x^T \beta \geq 0$, and $Y = 0$ when $p(x; \beta) < (1 - p(x; \beta))$, that is when $x^T \beta < 0$
- Remember that x includes the constant term 1 to accommodate the intercept β_0

Logistic regression

- Logistic regression predicts conditional probabilities (not only classes) → we can fit it using maximum likelihood
- Given N training observations (i.e. images), each described by a set of features x_i and a known binary class y_i , they can be seen as a sequence of iid Bernoulli trials.
- Remember that the Bernoulli distribution is $P(Y = y) = p^y(1 - p)^{1-y}$ where $y = \{0, 1\}$ which means that $P(Y = y) = p$ if $y = 1$, $P(Y = y) = 1 - p$ if $y = 0$ and 0 otherwise.
- The (conditional) likelihood of all N observations is:

$$L(\beta) = \prod_{i=1}^N P(Y = y_i | X = x_i; \beta) = \prod_{i=1}^N p(x_i; \beta)^{y_i} (1 - p(x_i; \beta))^{1-y_i} \quad (51)$$

Logistic regression

- It is more convenient to work with the log-likelihood $l(\beta) = \ln(L(\beta))$:

$$\begin{aligned} l(\beta) &= \sum_{i=1}^N y_i \ln(p(x_i; \beta)) + (1 - y_i) \ln(1 - p(x_i; \beta)) \\ &= \sum_{i=1}^N y_i \ln(P(Y = 1|X = x; \beta)) + (1 - y_i) \ln(P(Y = 0|X = x; \beta)) \\ &= \sum_{i=1}^N y_i \ln\left(\frac{P(Y = 1|X = x; \beta)}{P(Y = 0|X = x; \beta)}\right) + \ln(P(Y = 0|X = x; \beta)) \\ &= \sum_{i=1}^N y_i x_i^T \beta - \ln(1 + e^{(x_i^T \beta)}) = -\ln\left(e^{(-y_i x_i^T \beta)}(1 + e^{(x_i^T \beta)})\right) \end{aligned} \tag{52}$$

- No closed form solution → gradient and Hessian computation

Logistic regression

- We could use a gradient ascent $\beta^{new} = \beta^{old} + \frac{\partial l(\beta^{old})}{\partial \beta}$:

$$\begin{aligned}\frac{\partial l(\beta)}{\partial \beta} &= \sum_{i=1}^N x_i \left(y_i - \frac{\exp(x_i^T \beta)}{1 + \exp(x_i^T \beta)} \right) = \sum_{i=1}^N x_i (y_i - p(x_i; \beta)) \\ &= \sum_{i=1}^N x_i (y_i - P(Y = 1 | X = x_i; \beta))\end{aligned}\tag{53}$$

- the term within the parenthesis is the prediction error, namely the difference between the observed y_i and its predicted probability of being equal to 1. This is then multiplied by x_i , which accounts for the magnitude of the model $x_i^T \beta$ in making this prediction

Logistic regression

- Otherwise we could use the Newton's method:

$$\beta^{new} = \beta^{old} - \left(\frac{\partial l^2(\beta^{old})}{\partial \beta \beta^T} \right)^{-1} \frac{\partial l(\beta^{old})}{\partial \beta}$$

$$\frac{\partial l^2(\beta)}{\partial \beta \beta^T} = \sum_{i=1}^N x_i x_i^T p(x_i; \beta)(1 - p(x_i; \beta)) \quad (54)$$

- in matrix notation

$$\beta^{new} = \beta^{old} + (X^T W X)^{-1} X^T (\mathbf{y} - \mathbf{p}) \quad (55)$$

- where \mathbf{y} denotes the vector of y_i , \mathbf{p} is a vector containing the $p(x_i; \beta^{old})$, X is a matrix with the x_i and W is a diagonal matrix containing the $p(x_i; \beta^{old})(1 - p(x_i; \beta^{old}))$

Logistic regression

- Please note that if $y \in \{-1; +1\}$, the loss is different but the gradient is the same ! We can choose as loss:

$$\begin{aligned} L(\beta) &= \prod_{i=1}^N p(x_i; \beta)^{\frac{y_i+1}{2}} (1 - p(x_i; \beta))^{\frac{1-y_i}{2}} \\ l(\beta) &= \sum_{i=1}^N \frac{y_i + 1}{2} \ln(p(x_i; \beta)) + \frac{1 - y_i}{2} \ln(1 - p(x_i; \beta)) = \\ &= \sum_{i=1}^N -\left(\frac{1 - y_i}{2}\right) x_i^T \beta - \ln(1 + \exp(-x_i^T \beta)) \\ \frac{\partial l(\beta)}{\partial \beta} &= \sum_{i=1}^N x_i \left(\frac{1 + y_i}{2} - p(x_i; \beta)\right) \end{aligned} \tag{56}$$

Logistic regression

- An interesting choice of loss when $y \in \{-1; +1\}$ is
 $L(\beta) = \prod_{i=1}^N \frac{1}{1+\exp(-y_i x_i^T \beta)}$ which means maximizing

$$\begin{cases} p(x_i; \beta) = \frac{1}{1+\exp(-x_i^T \beta)}, & \text{if } y_i = 1 \\ 1 - p(x_i; \beta) = \frac{1}{1+\exp(x_i^T \beta)}, & \text{if } y_i = -1 \end{cases} \quad (57)$$

- This comes from the fact that

$$1 - \frac{1}{1 + \exp(-x_i^T \beta)} = \frac{\exp(-x_i^T \beta)}{1 + \exp(-x_i^T \beta)} \frac{\exp(x_i^T \beta)}{\exp(x_i^T \beta)} = \frac{1}{1 + \exp(x_i^T \beta)} \quad (58)$$

- The gradient is as before $\frac{\partial l(\beta)}{\partial \beta} = \frac{-\partial \ln(1+\exp(-y_i x_i^T \beta))}{\partial \beta} = \frac{y_i x_i \exp(-y_i x_i^T \beta)}{1+\exp(-y_i x_i^T \beta)}$:

$$\begin{cases} x_i \left(\frac{\exp(-x_i^T \beta)}{1+\exp(-x_i^T \beta)} \right) = x_i(1 - p(x_i; \beta)), & \text{if } y_i = 1 \\ -x_i \left(\frac{\exp(x_i^T \beta)}{1+\exp(x_i^T \beta)} \right) = -x_i p(x_i; \beta), & \text{if } y_i = -1 \end{cases} \quad (59)$$

Logistic regression

- Similarly to OLS, we could also add a regularization on the parameters β like $\|\beta\|_2^2$ (i.e. Ridge, avoid overfitting) or $\|\beta\|_1$ (i.e. LASSO, variable selection)
- Logistic regression is similar to LDA (decision boundary as log-ratio is a linear function in x) but it makes *less assumptions* since conditional distributions are not assumed to follow a Gaussian distribution → if data are not Gaussian logistic regression is more robust to outliers

Summary

1 Introduction - Supervised Learning

- Feature pre-processing
- Function space, Risk and Loss functions
- The Bias-Complexity tradeoff

2 Linear models for regression

- Ordinary Least Squares
- Ridge regression
- LASSO

3 Model selection and assessment

- Information criteria: AIC and BIC
- Cross-validation

4 Linear parametric models for classification

- Linear discriminant analysis (LDA)
- Quadratic discriminant analysis (QDA) and Naive Bayes
- Logistic regression

5 Non-parametric models

- K-nearest-neighbour

Summary

1 Introduction - Supervised Learning

- Feature pre-processing
- Function space, Risk and Loss functions
- The Bias-Complexity tradeoff

2 Linear models for regression

- Ordinary Least Squares
- Ridge regression
- LASSO

3 Model selection and assessment

- Information criteria: AIC and BIC
- Cross-validation

4 Linear parametric models for classification

- Linear discriminant analysis (LDA)
- Quadratic discriminant analysis (QDA) and Naive Bayes
- Logistic regression

5 Non-parametric models

- K-nearest-neighbour

K-nearest-neighbour

In the discrete case with a L_2 norm, we select as best f^* the one that minimizes:

$$f^*(x_j) = \arg \min_f E_{Y|X}[L(Y, f(x_j))|X = x_j] = E_{Y|X}[Y|X = x_j] \quad (60)$$

The k-nearest-neighbour method approximates this solution by looking within a neighbourhood of every point x_j . More precisely :

$$f_k^*(x_j) = \text{Ave}(y_i|x_i \in N_k(x_j)) = \frac{1}{k} \sum_{x_i \in N_k(x_j)} y_i \quad (61)$$

where Ave is the average, $N_k(x_j)$ is the neighbourhood of x_j defined by the k closest points x_i in the Training set \mathcal{T} . Closeness implies a metric (Euclidean, cosine, etc.) !

K-nearest-neighbour

If \mathcal{T} is large, it seems that we could always approximate $f(x_j)$ with a k-nearest-neighbour method, since we should always be able to find a fairly large neighbourhood of observations close to any x_j and average them.

Even in high dimension ? **Curse of dimensionality** ! If with $d = 1$ we need $N = 100$ points to accurately sample the space, we would need $N = 100^{10}$ points to preserve the same sampling density with $d = 10$!

K-nearest-neighbour - Bias-variance tradeoff

As before, let (x_0, y_0) be a test sample, $y_0 = x_0^T \beta + \epsilon_0$ the true model and $f_k^*(x_0)$ the k-nearest-neighbour estimate. The expected prediction error, that is the mean squared error (MSE) when using a L2-norm, of k-nearest-neighbour estimate at (x_0, y_0) is equal to:

$$\begin{aligned} \text{MSE} &= \mathbf{E}[(y_0 - f_k^*(x_0))^2] \\ &= \underbrace{\sigma^2}_{\text{irreducible error}} + \text{Var}_{\mathcal{T}}(f_k^*(x_0)) + \text{Bias}^2(f_k^*(x_0)) \\ &= \sigma^2 + \frac{\sigma^2}{k} + [f(x_0) - \mathbf{E}_{\mathcal{T}}[f_k^*(x_0)]]^2 \end{aligned} \tag{62}$$

When k increases the variance decreases but the bias is likely to increase
 $\rightarrow k$ can be chosen using CV !

K-nearest-neighbour

Pros

- Decision boundaries are non-convex in general
- It can be used with any metric
- It is naturally multi-class

Cons

- Curse of dimensionality
- Computational time can be long (need to compute many distances)

References

- ① Hastie, Tibshirani and Friedman (2009). The Element of Statistical Learning. Springer.
- ② Christopher M. Bishop (2006). Pattern Recognition and Machine Learning. Springer.
- ③ Shai Shalev-Shwartz and Shai Ben-David (2014). Understanding Machine Learning: From Theory to Algorithms. Cambridge University Press
- ④ Very good course:
<http://www.stat.cmu.edu/~ryantibs/datamining/>