Machine Learning and Pattern Recognition

Sandro Cumani

sandro.cumani@polito.it

Politecnico di Torino

Machine Learning and Pattern Recognition

Pattern Recognition:

Automatic discovery of regularities in data through the use of computer algorithms (...) to take actions such as classifying the data into different categories.[1]

Machine Learning:

A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E.[2]

[1] C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[2] T. M. Mitchell. *Machine Learning*. McGraw–Hill, 1997.

Machine Learning and Pattern Recognition

Define models that are able to capture the regularities in our data and allow performing inference about the properties we are interested in.

The models should not simply specify a set of human–defined rules, but should be able to learn from data.

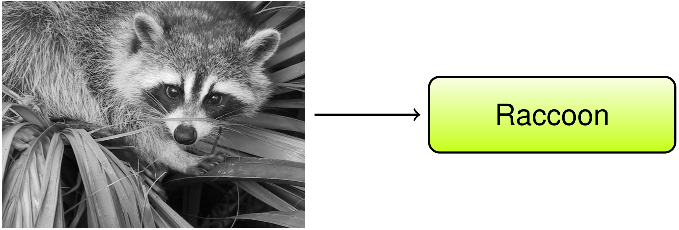
The learning stage should leverage observed data to improve the quality of the inference.

Example: Classification

Given a set of objects, assign them to discrete categories

Find a mapping from the space of input vectors representing the objects to a discrete set of labels (output values)

Possible applications include image and face recognition and speaker identification

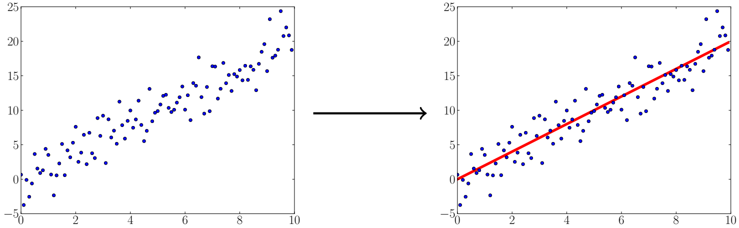


Raccoon

Example: Regression

Similar to classification, but output values are single or multiple continuous variables

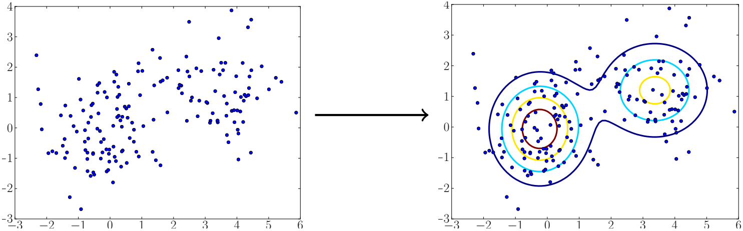
Possible applications include modeling of physical phenomena or prediction of continuous–valued functions (e.g. stock market prediction)



Example: Density Estimation

Estimation of the distribution of input vectors

Often used to explore data characteristics and to build inference models



Machine Learning and Pattern Recognition

Depending on the task, we identify two main branches

Supervised learning: Along with the *input* data the system is provided with *output* data (e.g. class labels, function values, ...). The goal is estimating a mapping between input and output data.

Common supervised tasks:

- Classification
- Regression

Machine Learning and Pattern Recognition

Depending on the task, we identify two main branches

Unsupervised learning: No feedback is provided to the system, whose goal is to identify some (useful) structure of the data.

Examples of unsupervised tasks:

- Clustering
- Density estimation
- Dimensionality reduction

Machine Learning and Pattern Recognition

Unsupervised methods are often used as pre–processing steps for supervised learning tasks (e.g. unsupervised dimensionality reduction).

Furthermore, some techniques are common to both supervised and unsupervised tasks (e.g. density estimation for generative classification models)

In this course we will focus on classification and density estimation tasks.

Pattern classification

- Assign a pattern to a class
- Classes represent characteristics of the objects that we are considering

  For example

  - What object is contained in an image
  - The language of a text
  - Tomorrow's weather
  - ...

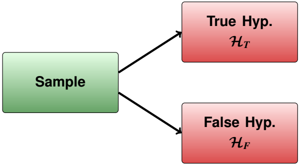
Pattern classification

- Binary classification

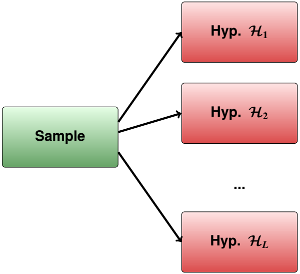
- Multiclass classification

Pattern classification

- Binary classification
  - Identity verification
  - Intrusion detection

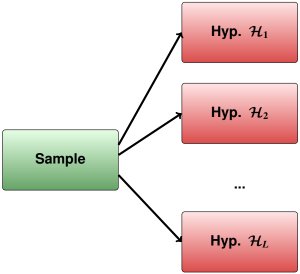


- Multiclass classification
  - Object categorization
  - Speech decoding

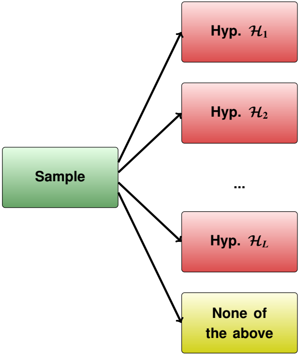


Pattern classification

- Closed–set classification

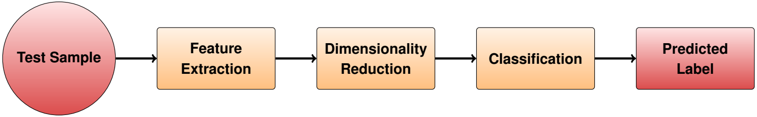


- Open–set classification



Pattern Classification

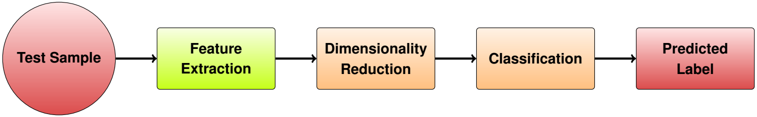
It is convenient to divide the problem in three different stages



In practice, these components often interact

- A neural network model can perform all the stages simultaneously
- Dimensionality reduction is often considered part of the feature extraction process (e.g. PCA)
- Dimensionality reduction techniques can be used to compute classification boundaries (e.g. LDA)

Feature Extraction



Test Sample

Feature Extraction

Dimensionality Reduction

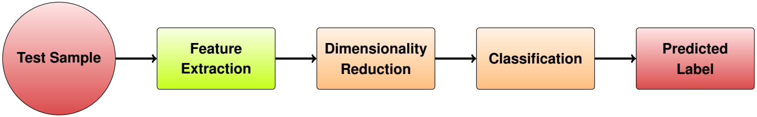
Classification

Predicted Label

We represent an object in terms of numerical attributes, usually arranged as vectors or matrices

$$\longrightarrow \begin{bmatrix} 116 & 133 & 149 & \dots & 117 & 128 & 136 \\ 86 & 107 & 126 & \dots & 116 & 132 & 144 \\ 70 & 91 & 112 & \dots & 117 & 133 & 144 \\ & \vdots & & \ddots & & \vdots & \\ 96 & 101 & 115 & \dots & 139 & 138 & 139 \\ 92 & 102 & 118 & \dots & 139 & 138 & 138 \\ 93 & 104 & 118 & \dots & 138 & 137 & 136 \end{bmatrix}$$

Feature Extraction



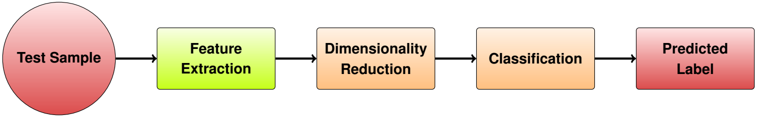
We represent an object in terms of numerical attributes, usually arranged as vectors or matrices

$to\ be\ or\ not\ to\ be \rightarrow$

to	2	*to be*	2
be	2	*be or*	1
or	1	*or not*	1
not	1	*not to*	1

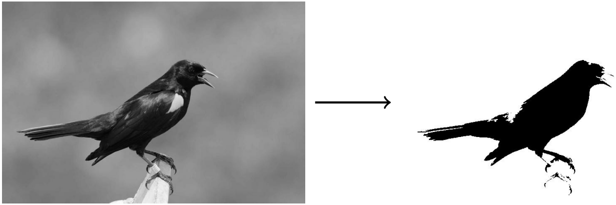
$$\longrightarrow \begin{bmatrix} 2.0 \\ 2.0 \\ 1.0 \\ 1.0 \\ 2.0 \\ 1.0 \\ 1.0 \\ 1.0 \end{bmatrix}$$

Feature Extraction

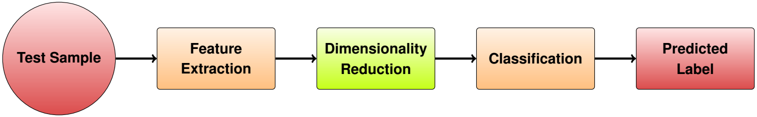


Test Sample → Feature Extraction → Dimensionality Reduction → Classification → Predicted Label

Often it involves (complex) manipulations of the object to extract useful representations



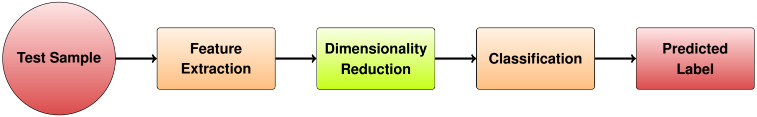
Dimensionality Reduction



The feature space is often very large and contains a large amount of unwanted and potentially harmful information.

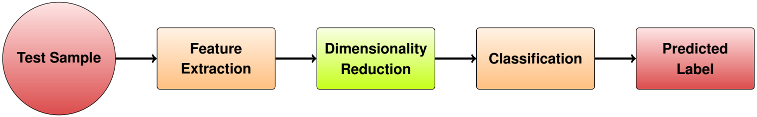
Dimensionality reduction techniques compute a mapping from the n–dimensional feature space to a m–dimensional space, with $m \ll n$.

Dimensionality Reduction



- Compress information
- Remove unwanted variability (noise)
- Data visualization
- Simplify classification (reduce curse of dimensionality, reduce overfitting)

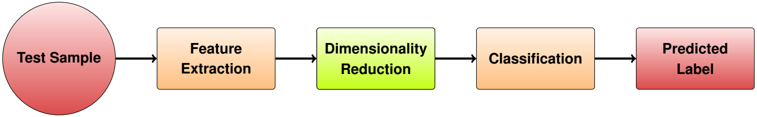
Dimensionality Reduction



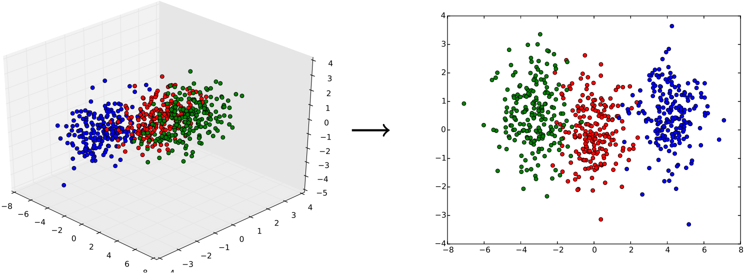
Overfitting: An over-complex model fits very accurately the observed data (very small training error), but is not able to provide accurate predictions for unseen data

Curse of dimensionality: Volumes in high-dimensional spaces grow very fast, and data becomes very sparse

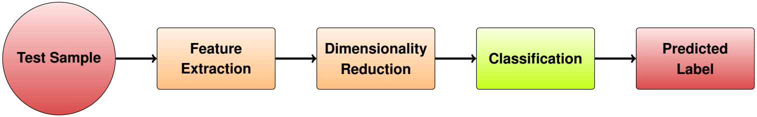
Dimensionality ReductionClassification



Test Sample → Feature Extraction → Dimensionality Reduction → Classification → Predicted Label



Classification

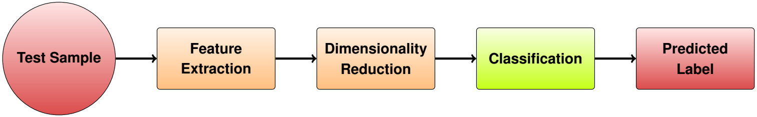


Given the (possibly reduced) feature vector representing an object, associate a label to the object based on the properties of the representation

Perform a mapping from the m–dimensional feature space to the space of labels

The mapping is also called decision function

Classification

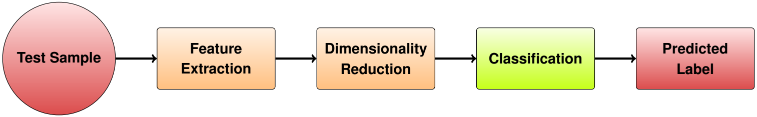


Our goal is to design suitable decision functions

- [3]Discriminant model: directly construct a function $f(\boldsymbol{x})$ mapping feature vector $\boldsymbol{x}$ directly to a label
- Discriminative non-probabilistic model: construct a function $f(\boldsymbol{x})$ mapping feature vector $\boldsymbol{x}$ to a set of "scores"

[3] C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

Classification



Our goal is to design suitable decision functions

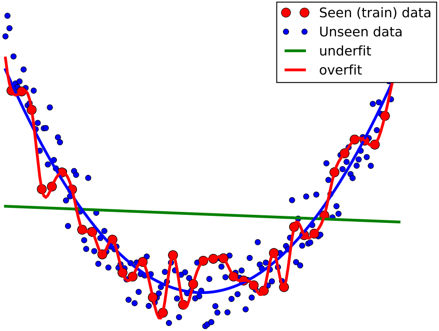
- Discriminative probabilistic model: model class posterior probabilities $P(\mathcal{C}_k|\boldsymbol{x})$ and assign labels according to posterior probabilities
- Generative probabilistic model: model the joint distribution of features and labels $P(\boldsymbol{x}, \mathcal{C}_k) = P(\boldsymbol{x}|\mathcal{C}_k)P(\mathcal{C}_k)$ and apply Bayes theorem to compute posterior probabilities $P(\mathcal{C}_k|\boldsymbol{x})$

Classification

The decision functions should provide good generalization error

We want models that are able to provide good predictions on unseen data (generalization)

Models describe data — too simple models can underfit our data, too complex models can overfit our data



Classification

Generative probabilistic model:

- The model describes the data generation process in terms of class-conditional distributions $P(\boldsymbol{x}|\mathcal{C}_k)$

- The model can incorporate application-dependent prior class information $P(\mathcal{C}_k)$

- Inference is based on Bayes theorem

$$P(\mathcal{C}_k|\boldsymbol{x}) = \frac{P(\boldsymbol{x}|\mathcal{C}_k)P(\mathcal{C}_k)}{P(\boldsymbol{x})}$$

- Class assignment is based on highest posterior probability $P(\mathcal{C}_k|\boldsymbol{x})$

- Probabilistic interpretation, optimal decisions depending only on prior information

Classification

Generative probabilistic model:

- Most demanding — it requires modeling per-class densities of our data
- Most informative — it allows us to *generate* new samples

Classification

Discriminative probabilistic model:

- Directly model class posterior probability $P(\mathcal{C}_k|\boldsymbol{x}_t)$
- Class assignment is again based on highest posterior probability
- Probabilistic interpretation
- Cannot directly incorporate application-dependent information – class prior probabilities are embedded in the model (though in many cases they can be compensated)

Classification

Discriminative non-probabilistic model:

- The output is a score for each class (or a single score for binary problems)
- The scores can be taken as a measure of strength of the hypotheses (possible class assignments) under test
- Class assignment in general is based on best (usually highest) score
- Cannot directly account for prior information (in some cases it's possible to recover probabilistic interpretations)

Classification

Discriminant function:

- Directly outputs the class label for pattern $\boldsymbol{x}$
- Does not allow measuring uncertainty for our predictions
- Cannot be adapted to take into account application costs and prior information

We will focus (mainly) on probabilistic models

Solving the inference problem

Create a model $\mathcal{M}$ describing relationships between features and labels

- Parametric models: The model depends on a set of parameters $\boldsymbol{\theta}$ whose size does not depend on the available data (we will focus on these models)
- Non-parametric models: The model complexity grows with the sample size

Create a (labeled) training set $\mathcal{D} = \{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_N, y_N)\}$

- The set contains the observed samples that will be used for the *learning* stage

Solving the inference problem — the frequentist approach

Learn the model parameters

We will mainly follow a frequentist approach

Since $\boldsymbol{\theta}$ is unknown, we compute an estimate $\boldsymbol{\theta}^*$ of its value from the training set:

$$\mathcal{D}, \mathcal{M} \rightarrow \boldsymbol{\theta}^*$$

Usually $\boldsymbol{\theta}^*$ is obtained through the optimization of a suitable objective function

$\boldsymbol{\theta}^*$ encodes all the information extracted from the dataset $\mathcal{D}$

Solving the inference problem — the frequentist approach

Inference: predict the class label $\boldsymbol{y}_t$ for a (previously unseen) test sample $\boldsymbol{x}_t$

- Generative model: compute the class-conditional probabilities for the sample $P(\boldsymbol{x}_t|\mathcal{C}_k, \mathcal{D}, \mathcal{M})$

$$P(\boldsymbol{x}_t|\mathcal{C}_k, \mathcal{D}, \mathcal{M}) \approx P(\boldsymbol{x}_t|\mathcal{C}_k, \boldsymbol{\theta}^*, \mathcal{M})$$

- Discriminative model: compute posterior class probabilities for the sample $P(\mathcal{C}_k|\boldsymbol{x}_t, \mathcal{D}, \mathcal{M})$

$$P(\mathcal{C}_k|\boldsymbol{x}_t, \mathcal{D}, \mathcal{M}) \approx P(\mathcal{C}_k|\boldsymbol{x}_t, \boldsymbol{\theta}^*, \mathcal{M})$$

Solving the inference problem — the Bayesian approach

The frequentist approach is not the only possibility

Alternative methods exsist, and may be also more effective in a variety of scenarios

Bayesian approach: consider the parameters as unknown quantities $\longrightarrow$ model θ as random variable(s)

Learning: update knowledge using probability rules

Solving the inference problem — the Bayesian approach

Define a prior distribution for $\boldsymbol{\theta}$

The prior distribution encodes our knowledge on the problem, before we actually see the data

Learn: update the prior distribution as to take into account the training set data

$$P(\boldsymbol{\theta}|\mathcal{M}), \mathcal{D} \rightarrow P(\boldsymbol{\theta}|\mathcal{D}, \mathcal{M})$$

The posterior distribution density $P(\boldsymbol{\theta}|\mathcal{D}, \mathcal{M})$ encodes our knowledge on the model parameters, after we have seen the data

Inference: marginalize over the model parameters — for example, for generative models:

$$P(\boldsymbol{x}_t|\mathcal{C}_k, \mathcal{D}, \mathcal{M}) = \int P(\boldsymbol{x}_t, \boldsymbol{\theta}|\mathcal{C}_k, \mathcal{M}) P(\boldsymbol{\theta}|\mathcal{D}, \mathcal{M}) d\boldsymbol{\theta})$$

Solving the inference problem — the Bayesian approach

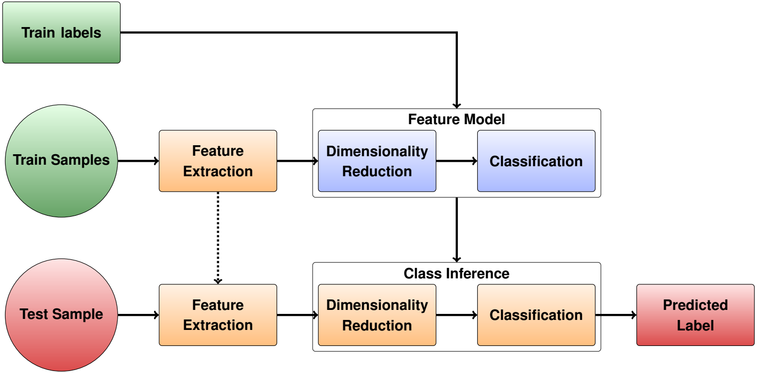
The Bayesian approach models the uncertainty over the model parameter estimates

Typically, effective when training data is scarce

However, both training and inference are significantly more complex

Due to the additional complexity, we won't see any Bayesian classification model

Machine learning for pattern classification



Machine learning for pattern classification

We also need to be able to assess the quality of our predictions

We are interested in performance on unseen data

Since we cannot know the system performance on unseen data, we simulate its behavior on *known* data

We use a (labeled) evaluation or test set

Machine learning for pattern classification

The test set should mimic real use cases

- It should contain data similar to that of our use case
- It should not contain data that was used to train the model
- Ideally, it should not be used to make any decision on the system

In practice, test sets are usually employed to compare the performance of different systems

Although unavoidable for practical reasons, with time this may lead to biased conclusions

Machine learning for pattern classification

Models and / or training procedures often contain hyperparameters, for example

- Number of dimensions of reduced features
- Step size in gradient-descent based optimization
- Regularization coefficients

The optimal value of these hyperparameters usually cannot be estimated using the same criterion employed for the model parameters

In other cases, we may have different competing models, and we would like to select the one that provides the best predictions

Machine learning for pattern classification

We need a way to perform model selection

However, we cannot use test data to select good hyperparameters values, as this would bias our evaluation

We make use of an additional training set, called validation set.

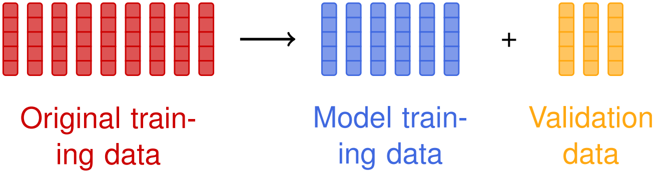
The validation set is used to asses the influence of model hyperparameters on prediction accuracy in order to select good hyperparameter values.

Machine learning for pattern classification

We can build a validation set by extracting some data from the training set

Part of the training set will be used for estimating our models

The remaining part (validation set) will simulate the evaluation data, and will be used to infer hyperparameters and for model selection



Machine learning for pattern classification

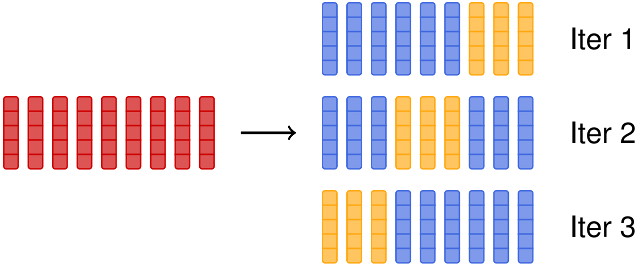
When training data is scarce, cross-validation is usually employed

K-fold cross-validation:

- Subdivide the training data in K folds
- Repeatedly use $K-1$ folds as training data, the remaining fold as validation set
- Combine the validation results on the K folds and select optimal values for hyperparameters
- Retrain the model using the selected hyperparameters but using all data

Machine learning for pattern classification

K-Fold: Repeatedly use $K-1$ folds as training data, the remaining fold as validation set



Machine learning for pattern classification

Small values of K require less computational resources (fewer models to train), but each model is trained with less data

Large values of K require training many models, but each model will be much more similar to the one trained over all data

Limiting case: leave-one-out approach

- Similar to K-fold, but each evaluation partition contains a single sample

Machine learning for pattern classification

In some experimental protocols, validation sets are extracted from the evaluation set

- Higher similarity with evaluation data
- We have to guarantee that the data does not overlap
- Validation data becomes, in all aspects, training material
- In real use cases, evaluation data may not be available, so the protocol may not reflect actual usage of the models. This may lead to biased results

For our projects, validation data can be extracted from the training set. Evaluation data should not be used to estimate anything