Appunti integrati di Machine Learning and Data Analitics (2017/2018)

A cura di

L. B. [1]

Materiale disponibile su https://youactuary.altervista.org/

[1] Lo scopo è quello di dare un ordine gerarchico agli argomenti presenti nel materiale del prof. Eric Medvet

Indice

1 Section 1: General information **35**

2 Section 2: Introduction **36**

2.1 **Binary classification problem** 36

2.1.1 Given any new observation, we want to automatically assign the species 37

2.1.2 Sketch of a possible solution:
1. learn a model (**classifier**)
2. "use" model on new observations 37

2.2 Choosing the model 37

2.2.1 The choice of the model/tool/algorithm to be used is determined by many factors:
- Problem size (n and p)
- Availability of an output variable (y)
- Computational effort (when learning or "using")
- Explicability of the model
- 37

2.3 Comparing many models 37

2.3.1 Experimentally: does the model work well on (new) data? It does not work well because:
- the data is not informative
- the data is not representative
- the data has changed
- the data is too noisy 37

2.4 Implementation 37

2.4.1 When "solving" a problem, we usually need:
- explore/visualize data
- apply one or more learning algorithms
- assess learned models 37

3 Section 3: Fundamentals of R **38**

4 Section 4: Tree-based methods **39**

4.1 How to build a decision tree 39

4.1.1 Dividi-et-impera (recursively):
- find a cut variable and a cut value
- for left-branch, dividi-et-impera
- for right-branch, dividi-et-impera 39

4.2 How to build a decision tree: detailed
function BuildDecisionTree(X; y)
- Recursive binary splitting
- Top down (start from the "big" problem) 39

4.3 Best branch 39

4.3.1 function BestBranch(X; y) 39

4.3.2 Classification error on subset 39

4.4 Stopping criterion 39

4.4.1 function ShouldStop(y) 39

4.4.2 tree depth larger than d_{max} 39

4.5 Best branch criteria 39

4.6 Best branch criteria: binary classification 42

4.6.1 Graphics (P(**y**,c), index of **y**) : $E \leq G \leq D$ 42

4.6.2 what happens with multiclass problems? 42

4.7 Categorical independent variables 42

4.7.1 Trees can work with categorical variables
Branch node is $x_i = c$ or $x_i \in C' \subset C$ (c is a class)
Can mix categorical and numeric variables 42

4.8 Stopping criterion: role of k_{min} 42

4.8.1 Suppose $k_{min} = 1$ (never stop for **y** size) 42

4.9 Tree complexity 42

4.9.1 When the tree is "too complex":
- less readable/understandable/explicable
- maybe there was noise into the data
Tree complexity is not related (only) with k_{min}, but also with data 42

4.9.2 Tree complexity: other interpretation
maybe there was noise into the data.
The tree fits the learning data too much:
- **(overfitting)**
- does not generalize (high **variance**: model varies if learning data varies) 42

4.10 High variance 42

4.10.1 "model varies if learning data varies": what? why data varies?
- learning data is about the system/phenomenon/nature S → a collection of *observations* of S, a point of view on S
- learning is about understanding/knowing/explaining S
if I change the point of view on S, my knowledge about S **should remain the same**! 42

4.11 Spotting overfitting 43

4.11.1 Learning error curve in the plane (Model complexity, Error) decreasing, while Test error curve decrease but then increase
Test error is error on unseen data 43
4.12 k-fold cross-validation 45
4.12.1 For each fold calulcate accuracy or classification error rate or any other meaningful measure 45
4.13 Fighting overfitting 45
4.13.1 large k_{min}
when building, limit depth
when building, don't split if low overall impurity decrease
after building, prune 45
4.14 Pruning: high level idea 45
4.14.1 1. learn a full tree t_0
2. build from t_0 a sequence $T = \{t_0, t_1, ..., t_n\}$ of trees such that
- t_i is a root-subtree of t_{i-1} ($t_i \subset t_{i.1}$)
- t_i is always less complex than t_{i-1}
3. choose the $t \in T$ with minimum classification error with k-fold cross-validation 45
4.15 **Subsection 1: Regression trees** 45
4.16 Regression with trees 45
4.16.1 Trees can be used for regression, instead of classification.
decision tree vs. regression tree 45
4.16.2 Tree building: decision → regression; Stopping criterion 45
4.17 Tree building: decision → regression 46
4.17.1 function BuildDecisionTree($\mathbf{X}$; $\mathbf{y}$)
($\neq$ mean $\mathbf{y}$ in shouldStop($\mathbf{y}$) 46
4.18 Best branch 46
4.18.1 function BestBranch($\mathbf{X}$; $\mathbf{y}$)
Minimize sum of residual sum of squares (RSS) (the two $\bar{y}$ in the argmin are different) 46
4.19 Stopping criterion 46
4.19.1 function ShouldStop($\mathbf{y}$)
RSS is 0 46
4.20 Interpretation 46
4.21 Regression and overfitting 46
4.22 Trees in summary 46
4.22.1 (+) easily interpretable/explicable
(+) learning and regression/classification easily understandable
(+) can handle both numeric and categorical values
(−) not so accurate 46
4.22.2 Tree accuracy? 48
4.23 **Subection 2: Trees aggregation** 48
4.24 Weakness of the tree 48

4.24.1 Small tree:
- low complexity
- will hardly fit the "curve" part of points
- *high bias, low variance*
Big tree:
- high complexity
- may overfit the noise (on the right part of graph example)
- *low bias, high variance* 48
4.25 Big tree view: 49
4.25.1 A big tree has
- a detailed view of the learning data (high complexity)
- "trusts too much" the learning data (high variance) 49
4.26 Wisdom of the crowds
the wisdom of the crowds: a collective opinion may be better than a single expert's opinion 49
4.26.1 What if we "combine" different big tree views and ignore details on which they disagree?
- many views
- independent views
- aggregation of views 49
4.27 Wisdom of the trees 49
4.27.1 - many views: just use many trees
- independent views: **??? learning is deterministic: same data → same tree → same view**
- aggregation of views: just average prediction (regression) or take most common prediction (classification) 49
4.28 Independent views 50
4.28.1 Independent views ≡ different points of view ≡ different learning data. But we have only one learning data! 50
4.29 Independent views: idea! 50
4.29.1 Like in cross-fold, consider only a part of the data, but:
- instead of a subset
- a sample with repetitions
→ (**y** omitted for brevity)
→ learning data size is not a limitation (differently than with subset)
original learning data $\mathbf{X} = \mathbf{x}_1^T\mathbf{x}_2^T\mathbf{x}_3^T\mathbf{x}_4^T\mathbf{x}_5^T \Rightarrow \mathbf{x}_1^T\mathbf{x}_5^T\mathbf{x}_3^T\mathbf{x}_2^T\mathbf{x}_5^T = \mathbf{X}_1$ 50
4.29.2 **Bagging** of trees (bootstrap, more in general) 50
4.30 Tree bagging 50

4.30.1 When learning:
1. Repeat B times
- 1.1 Take a sample of the learning data
2. learn a tree (unpruned)

When predicting:
1. Repeat B times
- 1.1 get a prediction from ith learned tree
2. predict the average (or most common) prediction

For classification, other aggregations can be done: majority voting (most common) is the simplest 50
4.31 How many trees? 51
4.31.1 B is a parameter:
- when there is a parameter, there is the problem of finding a good value
- remember k_{min}, depth
- it has been shown (experimentally) that:
– for "large" B, bagging is better than single tree
– increasing B does not cause overfitting
– (for us: default B is ok! "large" ≈ hundreds) 51
4.32 Bagging 52
4.32.1 Graphics (Number B of trees, Test error) curve tends to stabilize and does not swing too much 52
4.33 Independent view: improvement 52
4.33.1 Despite being learned on different samples, bagging trees may be correlated, hence views are not very independent
- e.g., one variable is much more important than others for predicting (**strong predictor**).

Idea: force point of view differentiation by "hiding" variables . . 52
4.34 Random forest 53
4.34.1 When learning:
1. Repeat B times
- 1.1 Take a sample of learning data
1.2 **consider only m on p independent variables**
1.3 learn a tree (unpruned)

When predicting:
1. Repeat B times
- 1.1 get a prediction from ith learned tree
2 predict the average (or most common) prediction

(observations and) variables are randomly chosen ...to learn a **forest** of trees 53

4.35 Random forest: parameter m 54
4.35.1 How to choose the value for m?
- $m = p \rightarrow$ bagging
it has been shown (experimentally) that
– m does not relate with overfitting
– $m = \sqrt{p}$ is good for classification
– $m = \frac{p}{3}$ is good for regression
– (for us, default m is ok!) 54
4.35.2 Experimentally shown: one of the "best" multi-purpose supervised classification methods is RF: (Manuel Fernandez) graph(classifier family, minimum Friedman rank), ma... 54
4.36 No free lunch! 56
4.36.1 (David H Wolpert)
- many restaurants, many items on menus, many possibly prices for each item: where to go to eat?
- no general answer
- but, if you are a vegan, or like pizza, then a best choice could exist 56
4.37 Observation sampling 57
4.37.1 When learning:
1. Repeat B times
- 1.1 **take a sample of the learning data**
- 1.2 consider only m on p independent variables (only for RF)
- 1.3 learn a tree (unpruned)

Each learned tree uses only a portion of the observation in the learning data:
- for each observation, $\approx \frac{B}{3}$ trees did not considered it when learned
- those observation were *unseen* for those trees, like in cross-validation (**OOB = out-of-bag**) 57
4.38 Bonus 1: OOB error 57
4.38.1 - for unseen each observation there are $\frac{B}{3}$ predictions
- can "average" prediction among trees, observation and obtain an estimate of the testing error (OOB error)
– like with cross-fold validation
– for free! 57
4.39 OOB (Out Of Bag) 57
4.39.1 Graph (number of trees, Error): test bagging, test random forest, oob bagging, oob random forest 57
4.40 Why estimating the test error? 58
4.40.1 Because the test data, in real world, is not available! 58
4.41 Bagging/RF and explicability 60
4.41.1 - Trees are easily understandable → explicability
- Hundreds of trees are not! 60

4.42 Bagging/RF and explicability: idea! 60
4.42.1 While learning:
1. for each tree, at each split
- 1.1 keep note of the split variable
- 1.2 keep note of RSS/Gini reduction
2. for each variable, sum reductions

The largest reduction, the more important the variable! 60
4.43 Bonus 2: variable importance 61
4.43.1 Instead of explicability based on tree shape: importance of variables based on RSS/Gini reduction 61
4.44 Nature of the prediction 61
4.44.1 Consider classification:
- tree → the class
- forest → the class, as resulting from a voting 61
4.45 Bonus 3: confidence/tunability 61
4.45.1 Voting outcome:
- in classification, a measure of confidence of the decision
in binary classification, voting threshold can be tuned to adjust bias towards one class (**sensitivity**) 61
4.46 Binary classification 61
4.46.1 (example) Consider the problem of classifying a person ('s data) as suffering or not suffering from a disease X.
- **positive**: an observation of "suffering" class
- **negative** an observation of "not suffering" class

In other problems, positive may mean a different thing: define it! 62
4.47 FPR, FNR 62
4.47.1 Given some labeled data and a classifier for the disease X problem, we can measure:
- the number of negative observations *wrongly* classified as positives: **False Positives** (**FP**)
- the number of positive observations wrongly classified as negatives **False Negatives** (**FN**)

To decouple FP, FN from data size:
$FPR = \frac{FP}{N} = \frac{FP}{FP+TN}$
$FNR = \frac{FN}{P} = \frac{FN}{FN+TP}$ 62
4.48 Accuracy and error rate 62
4.48.1 Accuracy = 1 - Error Rate
Error Rate= $\frac{FN+FP}{P+N}$ 62
4.49 FPR, FNR and sensitivity 63

4.49.1 Suppose FPR = 0.06, FNR = 0.04 with threshold set to 0.5 (default for RF)
- One could be interested in "limiting" the FNR...
graph (threshold t, error rate) curves FPR, FNR 63
4.50 Receiver operating characteristic (ROC) 63
4.50.1 graph (threshold t, error rate) curves FPR, FNR:
- FPR ∩ FNR → **Equal error rate** (**EER**) point in which the percentage error is the same 63
4.51 ROC and comparison 66
4.51.1 EER e AUC is used to compare two binary classificator 66
4.52 Many views and aggregation 66
4.52.1 In bagging/RF (regression):
- many views are different samples
- aggregation is average

Alternative:
- many views are subsequent residuals
- aggregation is the sum 66
4.53 Boosting 67
4.53.1 When learning:
1. Current data is learning data
2. Repeat B times
- - 2.1 learn a tree on current data
- - 2.2 current data becomes residuals of learned tree ($\hat{\mathbf{y}} - \mathbf{y}$)

When predicting:
1. Repeat B times
- 1.1 get a prediction from ith learned tree
2. sum prediction

differences w.r.t. RF? In boosting it is not usable parallel calculation 67
4.54 Boosting (regression) 68
4.54.1 function BoostTrees($\mathbf{X}$; $\mathbf{y}$)
BuildDecisionTree($\mathbf{X}$; $\mathbf{y}$; d)

- Each learned tree should be simple (maximum splits d)
- λ slows down learning

Trickier with classification. 68
4.55 Boosting parameters 68

4.55.1 - λ usually set to 0.01 or 0.001
- λ and B interact: for small λ, B should be large
large B can lead to overfitting (unlike bagging/RF (because it is trying to improve the residues)

Find a good value for B with cross-validation 68

4.56 Bagging/RF/boosting in summary 69

4.56.1 Tree | Bagg | RF | Boost

↑ | / | / | / interpretability

↑ | ↑ | ↑ | ↑ numeric/categorical

⇓ | ↑ | ↑ | ↑ accuracy

/ | ↑ | ↑ | / test error estimate

/ | ↑ | ↑ | ↑ variable importance

/ | ↑ | ↑ | / confidence/tunability

/ | ↑ | ↑ | ⇓ fast to learn

/ | ↑ | ↑ | / (almost) non-parametric

69

5 Section 5: Support Vector Machines **70**

5.1 Binary classification 71

5.1.1 Let's draw a decision boundary! (in 2 dimension) 71

5.2 Hyperplane 71

5.2.1 We drew a line: $X_2 = mX_1 + q$ or $\beta_0 + \beta_1 X_1 + \beta_2 X_2 = 0$
in p-dimensional space $\beta_0 + \beta_1 X_1 + ... + \beta_p X_p = 0$ the line is a *separating hyperplane.* 71

5.3 Classification with a separating hyperplane 71

5.3.1 Given an osservation (2,3) it belongs to half plane on left or right

The larger the difference, the stronger the confidence (i.e. more distant from the line, more strong is the difference) 71

5.4 Learning a separating hyperplane 71

5.4.1 We want an hyperplane which perfectly separates the learning data... but there could be many ∞ of them! Which one?
Idea: the farthest from the learning observations! Idea: the farthest from the learning observations! **Maximal margin classifier** 71

5.5 Maximal margin classifier 71

5.5.1 margin M: the margin is the maximum distance between the nearest point of any classes from the separation line
support vectors are the nearest point of any classes from the separation line 71

5.6 Learning the maximal margin classifier 73

5.6.1 Find the line which:
- 1. perfectly separates learning observations
- 2. has the largest margin from support vectors

Looks like an optimization problem. 73

5.7 Support vectors 73

5.7.1 $\forall i \in \{1, 2, ..., n\}$, $y_i(\beta_0 + \beta_1 x_{i,1} + ... + \beta_p x_{i,p} = M$

They lie exactly on the margin! 73

5.8 Learning the maximal margin classifier 73

5.8.1 Looks like an optimization problem... which is not hard to be solved. 73

5.9 Maximal margin classifier issues 73

5.9.1 - What if the learning data is not perfectly separable? cannot learn!
- What if a learning observation (being a support vector) is added/removed? could learn a very different classifier ! high variance! 73

5.10 High variance of Maximal margin 74

5.10.1 Image 74

5.11 Soft margin 74

5.11.1 How to cope with these issues? Idea: be more tolerant!
- some learning observation may be within the margin
- some learning observation may be misclassified

Margin can be exceeded → **soft margin classifier** or **support vector classifier** 74

5.12 Learning with toleration: support vector classifier 75

5.12.1 $\max\limits_{\beta_0,..,\beta_p,\epsilon_1,...,\epsilon_n}$ M under constraints $\sum_{j=1}^{p} \beta_j^2 = 1$
$\forall i \in \{1, 2, ..., n\}$,
ϵ_i are positive slack variables
C is the toleration budget (C = 0 ! maximal margin classifier) (cfr 5.2.7)
$y_i(\beta_0 + \beta_1 x_{i,1} + ... + \beta_p x_{i,p} = M \underbrace{(1 - \epsilon_i)}$
$\underbrace{\forall i \in \{1, ..., n\}, \epsilon_i \geq 0}$
$\underbrace{\sum_{j=1}^{n} \epsilon_j = C}$ 75

5.13 Role of the parameter C 75

5.13.1 The larger C
- the larger the toleration
- the larger the number of learning observations which can exceed the margin (or be misclassified)
- the larger the number of support vectors
- the lower the variance 75
5.14 Summary 75
5.14.1 Maximal margin | Soft margin
↑ | ↑ fast to learn
⇓ | ↑ variance
⇓ | ↑ robustness to "trivial" observations 75
5.15 Linearity? 76
5.15.1 Some problems cannot be solved with an hyperplane! (es (x_1, x_2) of virginica and setosa points mixed together 76
5.16 Some math rewriting 77
5.16.1 Finding values for $\beta_0, ..., \beta_p$ involves computing inner products between pair of observations: $\langle x_i, x_i' \rangle = \sum_{j=1}^{p} x_{i,j} x_{i',j}$ And we can rewrite: $\beta_0 + \sum_{i=1}^{p} \beta_i x_i^* = f(x^*) = \beta_0 + \sum_{i=1}^{n} \alpha_i \langle x^*, x_i \rangle$
x^* is observation which we want to classify
For non support vectors, $\alpha_i = 0 \Rightarrow x_i$ does not impact on $f(x^*)$! 77
5.17 Non support vectors 77
5.17.1 $f(x^*) = \beta_0 + \sum_{i=1}^{n} \alpha_i \langle x^+, x_i \rangle$
- $f(x^*)$ is the distance of x^* from the decision boundary
- the (position of the) decision boundary depends only on the support vectors
- $\Rightarrow f(x^*)$ depends only on the support vectors
When predicting:$f(x^*) = \beta_0 + \sum_{\underbrace{i \in \mathcal{S}}} \alpha_i \langle x^*, x_i \rangle$
$\mathcal{S}$ set of indexes of support vectors 77
5.18 Kernel 77
5.18.1 Equation for the decision boundary can be generalized:
$f(x^*) = \beta_0 + \sum_{i=1}^{n} \alpha_i K(x^*, x_i)$
Where $K(x^*, x_i)$ is a function $K : R^p \times R^p \rightarrow R$ called **kernel** (with some other properties).

Support Vector Machines 77
5.19 Intuition for the kernel 78

5.19.1 Consider prediction:
$f(x^*) = \beta_0 + \sum_{i\in\mathcal{S}} \alpha_i K(x^+, x_i)$
- x^* is mapped from R^p to $R^{p'}$ with $p' >> p$, using a function Φ : $K(x_i, x_j)$ computes the inner product $\langle \phi(x_i), \phi(x_j)\rangle$ of mapped x_i, x_j without explicitly mapping them (**kernel trick**)
- the α_i define (indirectly) an hyperplane in $R^{p'}$
- the classification is done by means of a separating hyperplane in the new space, i.e.,$f(x^*)$ measures the distance of *mapped* x^* from the hyperplane 78

5.20 Kernels 78

5.20.1 1 - linear kernel:
$K(x^*, x_i) = \langle x_i, x^*\rangle = \sum_{j=1}^{p} x_{i,j}x_j^*$
2 - polynomial kernel: (d is the degree)
$K(x^*, x_i) = \Big(1 + \sum_{j=1}^{p} x_{i,j}x_j^*\Big)^d$
3 - radial basis function kernel (or radial, or RBF, or Gaussian):
$K(x^*, x_i) = \exp\Big(-\gamma \sum_{j=1}^{p}(x_{i,j} - x_j^*)^2\Big)$ (very used, default in R) 78

5.21 Intuition behind radial kernel 81

5.21.1 $K(x^*, x_i) = \exp\Big(-\gamma \sum_{j=1}^{p}(x_{i,j} - x_j^*)^2\Big) = \exp(-\gamma||x_i, x^*||^2)$
- the coordinates in the new space are related to the distances of x^* from the support vectors (the closer, the higher the $K(\cdot)$, $K(\cdot) \in]0; 1]$)
- γ determines how fast the coordinate goes to 0, i.e., a support vector becomes irrelevant for classifying x^* 81

5.22 Very raw visual intuition 81

5.22.1 example in which increase dimension of space to increase distance between points new points (x*,y*,z*) $\in \Big(d(x), d(y), d(y)\Big)$ 81

5.23 Multiclass ($\geq$2) classification with SVM 81

5.23.1 one-vs.-one classification
one-vs.-all classification

and many other proposals... 81

5.24 One-vs.-one SVM 81

5.24.1 When learning:
1. for each pair $(\mathcal{C}_1, \mathcal{C}_2)$ of classes, learn a binary SVM

When predicting:
1. for each learned SVM, predict class $\hat{y}$
2. choose the most frequently predicted class

$\binom{k}{2} = \frac{k(k-1}{2}$ binary classifier 81

5.25 One-vs-all SVM 81

5.25.1 When learning:
1. - for each class C_i , learn a binary SVM ($\mathcal{C}_i$ vs. all $\mathcal{C}_j$, with $j \neq i$, $\mathcal{C}_i$ coded as y=+1)

When predicting:
1. for each learned SVM, get $f(x^*)$
2. choose the class with the largest $f(x^*)$

K classifiers . . . 81

6 Section 6: Text mining 83

6.1 Text mining . . . 84

6.1.1 Definition
Text mining is about extracting high level information from textual data.

A joint effort of Machine Learning and Natural Language Processing (NLP). . . . 84

6.2 Example 1: sentiment on brands . . . 84

6.2.1 Is people on Twitter talking good or bad about brand X? . . . 84

6.3 Example 2: topics in letters . . . 84

6.3.1 In this corpus of letters to/from the front in WW1, which are the topics covered? . . . 84

6.4 Common tasks . . . 84

6.4.1 - text categorization
- text clustering
- entity extraction
- sentiment analysis
- 84

6.5 Example 3: relevance of citations . . . 85

6.5.1 Find a way to quantify relevance of a citation from a scientific paper A to a scientific paper B? . . . 85

6.6 Step 0 . . . 85

6.6.1 Define the nature of the solution:
- input
- output
- learning data (if any)
Define a way to asses a solution . . . 85

6.7 Natural language and ambiguity . . . 85

6.7.1 Text is (usually) natural language
Natural means humans naturally express" → ambiguity!

Expect results" to be worse than in normal ML! . . . 85

6.8 **Subsection 1: Sentiment analysis (and text categorization)** . . 85

6.9 Problem formalization . . . 85

6.9.1 Input: a piece of text (*document*)
- Output?
– numeric value in [-1 1] (positivity)
–a categorical value in {Pos; Neg}
– categorical value in {Pos; Neutral; Neg}

Regression, multiclass classification, or binary classification (possibly with confidence).

We can use the techniques we already know (e.g., RF)! 85

6.10 Which are the features? 86

6.10.1 example: review of a restaurant: “Good coffee. Great for families. Always had good service. We go early so pretty empty. Flexible with menus. Wish they would remove service charge.” → Need to transform a document into an numerical vector! 86

6.11 Text to features 86

6.11.1 - Not only for sentiment analysis
- Many options
- Options can be combined 86

6.12 Bag of words 86

6.12.1 One dimension (feature, dependent variable) for each word
Value of $x_{i,j}$ is the number of occurrences of j-th word in the i-th document

t_1 = the cat is on the table
$x_{1;the} = 2$
$x_{1;cat} = 1$
... . 86

6.13 Bag of which words? 87

6.13.1 - One dimension (feature, dependent variable) for each word

- Which words? How big is x_i? p =?
- common solution: the most k frequent words in the corpus
"Interesting" words not frequent enough in the corpus may be lost 87

6.14 Stop words 87

6.14.1 Some words may be very frequent, but useless for specific task (e.g., sentiment analysis)
- a, an, the, are, . . . (**stop words**)
- Just remove them!

Stop words are language dependent! 87

6.15 Stemming 87

6.15.1 - There are variants for many words:
– drink, drinks, drinking
–happy, happier
- Even more in other languages:
– mangio, mangia, mangi, mangiai,...
- **Stemming:** reduce word to its word stem (the morphological root)
– drinking → drink
– argued → argu

Stemming are language dependent! 87
6.16 A typical workow 88
6.16.1 Preprocessing ($d \to d'$):
- 1. remove punctuation
- 2. to lowercase
- 3. remove stop words
- 4. stemming
Learning
- 1. preprocess each d in corpus
- 2. find most frequent k words in preprocessed corpus
- 3.compute $\mathbf{X}$
- 4. learn a classifier
Predicting
- preprocess input d
- predict based on preprocessed d 88
6.17 Limitations and caveat: punctuation 89
6.17.1 remove punctuation

It has been show that often punctuation matter (e.g., Twitter sentiment analysis):
I just saw Alice.
I just saw Alice!!!!!
I just saw Alice!!! :-)))) 89
6.18 Words that matter 89
6.18.1 Word count may be too coarse to capture desired information:
- documents with very different lengths
- irrelevant terms with general high frequencies

Use frequency or more complex variants 89
6.19 Goal, context, hypothesis 89

6.19.1 -Twitter profiling: predict age and gender of user from his/her tweets.
- people of different ages differently use case
- people of different age/gender differently use punctuation

A step in the work ow corresponds to an (implicit) hypothesis:
- remove stop words → stop words frequencies is not useful for predicting X 89
6.20 Tf -idf 90
6.20.1 $x_{i,j} = x_{d,t} = tf(t,d)idf(t,D)$
- $tf(t;d) = f_{t,d}$ term frequency
– the more important the term t in document d, the larger
$idf(t,D) = \frac{|D|}{|\{d \in f_{t,d} > 0\}|}$ inverse document frequency
–the more common t in the corpus, the lower 90
6.21 Bag of words and ordering 91
6.21.1 Sentiment analysis of restaurant reviews:
- t_1 = The beer was good and the pub was not too noisy.
- t_2 = The beer was not good and the pub was too noisy.

- fundamental problem: ordering is lost
- even more fundamental: natural language can be hard to algorithmically understand (irony, sarcasm,...)

Solutions:
-ngrams
- text parsing (NLP) 91
6.22 Parsing: POS tagging 91
6.22.1 Assign a role to part of speech (**POS**) (is the process of assigning a part of speech or lexical class marker to each word in a corpus) 93
6.23 n-grams 93
6.23.1 Instead of counting words, count short (up to n) sequences of words: $x_{1;dog,eat,cat}$ instead of $x_{1;dog}$
- size of data (p) grows dramatically (and is sparser)
- useful in general for manipulating sequences

n-gram is a contiguous sequence of n elements of a given text or speech sample; objects can be phonemes, syllables, letters, words 93
6.24 Generality of a sentiment classifier 93

6.24.1 How many sentiment classifier should exist? Words that matter in sentiment should be predefined
- predefined list of opinion words (positive, negative), i.e., features are those words
- but context often matter
– predictable is good for a car and bad for a movie
– features for sentiment analysis in Twitter are likely different than features from sentiment analysis of a early '900 writer's corrispondence

There are many pre-trained tool, often with more complex outcome than positive/negative. 94
6.25 Out-of-the-box sentiment analysis 94
6.25.1 Should I use a pre-trained tool or build my own?

It depends:
- is sentiment analysis just a piece of a more complex ML system?
- which is my budget?
- is learning data easily available? 94
6.26 Lab: Text categorization: sport vs. politics 94
6.26.1 Build a binary classifier for tweets: sport vs. politics
- 1. decide input, output
- 2. decide solution assessment
- 3. decide (if any) how to obtain learning data
- 4. decide work and ML technique 94
6.27 **Subsection 2: Topic modeling (very very very briey)** 94
6.28 Topics 94
6.28.1 Given a corpus of documents, what do they talk about?
- talks about → *topic* 94
6.29 Probabilistic model 95
6.29.1 Assume stochastic document building process:
- there exist k topics
- a topic is a distribution over words
- a topic is assigned to the document according to a known probability (a document may exhibit multiple topics)
- a word in a document is drawn according to topic and document-topic assignment

Words order does not matter!

image rom Princeton 95
6.30 Probabilistic graphical model 97

6.30.1 - nodes are random variables
- edges are dependencies
- shaded nodes are observed
- boxes are repeated variables 97
6.31 Latent Dirichlet allocation (LDA) 97
6.31.1 A way for inferring distributions/assignments from observed values! (Posterior inference)

Given K (parameter),
- for each topic, compute words distribution
- for each document, compute topic "distribution"

- **Latent** refers to the unknown random variables
- **Dirichlet** is the distribution assumed for topics and words
- **Allocation** of words to topics and topics to documents 97
6.32 LDA internals 99
6.32.1 (Just a coarse overview)
While inferring posterior, try to (both):
- associate each document with as few topics as possible
- associate each document with as few words as possible

Conflicting goals, which results in finding (and putting in the same topics) words which often co-occur 99
6.33 For each document of the corpus, a vector in $[0;1]^K$ where i-th value is "how much the document exhibits i-th topic"
- - reasonable values for the number of topics K is some tens (10-50)
- For each topic, a vector $[0,1]^V$ where the i-th value is "how much the i-th word (on V words) is associated with the topic"
- - how to visualize/understand a topic? Select its most likely words . . 99
6.34 Visualize topics 100
6.34.1 image from princeton 100
6.35 LDA as a building block 100
6.35.1 - corpus visualization
- document similarities
- ...
- document → R^K 100
6.36 LDA: document→ R^k 100
6.36.1 - How to apply to new data?
assume everything is known (i.e., already computed on the corpus)
- just infer the posterior of topic assignment for the new document100
6.36.2 Lab: Sport vs. politics with topics 100

7 Section 7: Recommender systems 101
7.1 Recommender systems 102

7.1.1 Scenario: a service where users consume items
- predict the users' rating to unconsumed items
- great interest from industry
- - e-commerce
- - online social networks
- - entertainment on demand
- users usually pay for consuming 102
7.2 Example: movie recommendation 102
7.2.1 A| B | C| D| Movie
5 | 5 | 0 | 0 | Hugs and kisses
5 | ? | ? | 0 | Sweetness day
? | 4 | 0 | ? | The true love
0 | 0 | 5 | 4 | Crazy Max
0 | 0 | 5 | ? | The final judgement

- some users rated some movies
- predict the rating of unrated movies
(it is a regression problem (for each individual) 102
7.3 Content-based representation 102
7.4 Notation 104
7.4.1 - X_1 represents "romance"
- X_2 represents "action"
- $X_0 = 1$ represents bias
A| B | C| D| X_1| X_2 | Movie
5 | 5 | 0 | 0 | 0.95 | 0.01 | Hugs and kisses (+ romance than action)
5 | ? | ? | 0 | 1 | 0 | Sweetness day
? | 4 | 0 | ? | 0.99 | 0 |The true love
0 | 0 | 5 | 4 | 0 | 1 | Crazy Max
0 | 0 | 5 | ? | 0.2 | 0.99 |The final judgement

- $r_{i,j} \in \{0, 1\}$ is 1 iff user j rated movie i
- $y_{i,j}$ is rating given by user j to movie i (iff $r_{i,j} = 1$)
- $\mathbf{x}_i$ is the feature vector of movie i

"predict the rating of unrated movies" corresponds to solving
to **solving n_u (number of users) regression problems**
- learn $f_{\text{Alice}}(x), f_{\text{Bob}}(x), ...$

example for Alice: It's a regression problem (my Y is 5 5 ? 0 0)
→ predict Alice's preferences 104
7.5 Recommendation as linear regression 104

7.5.1 Assume a linear dependency between rating and features:
$y_{i,j} = \theta_{0,j}x_{i,0} + \theta_{1,j}x_{i,1} + \theta_{2,j}x_{i,2} + ... = \theta_j^T x_i$ (movie i, user j)
$\theta_j \in R^p$ is the set of parameters of user j
- θ_j represents preferences of user j

"solving n_u (number of users) regression problems" corresponds to **for each user** j, **learn** θ_j
it seems reasonable to use the linear regression model 104

7.6 Learning θ_j 106

7.6.1 we want to learn the user's preferences that minimize the error on the evaluation of the films which he has already evaluated
- for a single user (at the moment) - minimize sum of squared errors
- consider only rated movies ($r_{i,j} = 0$ for unrated)
- we also want to minimize the square sum of preferences ("regulariazion") so as not to evaluate extreme features
- If lambda is zero then we don't do regularization in coefficient theta and so is the risk of overfitting. If lambda is too large than we can occur to large bias!!!!
$\min_{\theta_j} \frac{1}{2}\sum_{i=1}^{n} r_{i,j}\left(\theta_j^T x_i - y_{i,j}\right)^2 + \frac{\lambda}{2}\sum_{k=1}^{p}\theta_{k,j}^2$ 106

7.6.2 for all users $\min_{\theta_1,\ldots,\theta_{n_u}} \frac{1}{2}\sum_{j=1}^{n_u}\sum_{i=1}^{n} r_{i,j}\left(\theta_j^T x_i - y_{i,j}\right)^2 + \frac{\lambda}{2}\theta_j^T\theta_j$
Can be solved with any optimization algorithm, e.g., gradient descent 107

7.7 The movie features? (change pov) 107

7.7.1 Which features to characterize movies? (or songs, products, people,...)
I Who assign feature values to movies? Is it costly?
- scale?
Assume we "want" p features:
A |B |C |D ||X_1|...|X_p| Movie
1 | 2 | 2 | 4 ||? | ?|...|?| film1
0 | 2 | 1 | 5 ||? | ?|...|?| film2
107

7.8 Collaborative filtering 107

7.8.1 In content-based:
- we know movie features $x_1, x_2, ..., x_n$ and ratings $\mathbf{y}_1, \mathbf{y}_2, ..., \mathbf{y}_{n_u}$
- we learn users' preferences $\theta_1, \theta_2, ..., \theta_{n_u}$
Assume we know users' preferences:
- we learn movie feature
Users (implicitly) **collaborate** to characterize content 108

7.9 Learning features from preferences 108

7.9.1 - one movie
$\min\limits_{x_i} \frac{1}{2}\sum_{j=1}^{n_u} r_{i,j}\left(\theta_j^T x_i - y_{i,j}\right)^2 + \frac{\lambda}{2} x_j^T x_j$
- all movies
$\min\limits_{x_1,\dots,x_n} \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n_u} r_{i,j}\left(\theta_j^T x_i - y_{i,j}\right)^2 + \frac{\lambda}{2}\sum_{i=1}^{n} x_j^T x_j$ 108

7.10 Users' preferences 109

7.10.1 "Assume we want p features" corresponds to "users' preferences are p dimensional"
- how to collect users' preferences?
- how many?
- relation with "linear dependency assumption"?

It may be preferable to learn features and preferences together! 109

7.11 Learning features and preferences 109

7.11.1 $\min\limits_{x_1,\dots,x_n,\theta_1,\dots,\theta_{n_u}} \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n_u} r_{i,j}\left(\theta_j^T x_i - y_{i,j}\right)^2 + \frac{\lambda}{2}\sum_{i=1}^{n} x_j^T x_j + \frac{\lambda}{2}\sum_{j=1}^{n_u} \theta_j^T \theta_j$
- it's a big optimization problem. Users work together to characterize the content.
- here we regularize both for preferences and for items 109

7.12 New user? 109

7.12.1 $\min\limits_{x_1,\dots,x_n,\theta_1,\dots,\theta_{n_u}} \frac{1}{2}\underbrace{\sum_{i=1}^{n}\sum_{j=1}^{n_u} r_{i,j}\left(\theta_j^T x_i - y_{i,j}\right)^2}_{0} + \frac{\lambda}{2}\sum_{i=1}^{n} x_j^T x_j + \frac{\lambda}{2}\sum_{j=1}^{n_u} \theta_j^T \theta_j$
- sum of squared errors is always zero for the new user
- the only goal is to minimize sum of preferences"
what happens without regularization?The values will remain those of initialization
- which results in no preferences ($\forall k, \theta_{k,j} = 0$), and hence equal predicted ratings for all movies 109

7.13 Cold start problem 109

7.13.1 When something "new" arrives and no data is available
- new user
- new movie
One possible solution: use mean values $\mathbf{y}_{j'} = \frac{1}{n_u}\sum_{j=1}^{n_u} \mathbf{y}_j$ 109

7.14 Recommender system assessment 109

7.14.1 Many options:
- as a regression problem, RMSE (root mean square error (to understand how good it is. I take the y, I take the y cap estimate, I look at the RMSE)
- as a classification problem, accuracy
(if someone tells me what is right or wrong I can make it become a recommendation problem.) - - does the tool recommend the most preferred item to the user?
- as a classification problem, **accuracy@K**
– does the tool recommend the most preferred item to the user among the top k recommendations?(I take the most recommended films, with a prefixed number of elements k. Then I ask the program: "tell me the 5 numbers you would recommend"). it is accuracy on the first k recommended elements.
- as an information retrieval problem, precision and recall
- -does the tool recommend relevant items?

Prec = $\frac{(\text{relevant}\wedge\text{recomm.})}{recomm.}$ Rec.= $\frac{(\text{relevant}\wedge\text{recomm.})}{relevant}$
precision: number of documents that are simultaneously returned (recommended) and relevant.
In practice, how to measure them? I measure them as I would measure them in a recommendation problem. Assume that if a user has consumed a content, the recommendation is made. . . 110

7.15 Beyond accuracy 111

7.15.1 - diversity (desirable diversity (do not always recommend the same films))
- serendipity: positive surprise
- revenue? number of click/user/usages?

More in general, UI plays a crucial role! 111

8 Section 8: Evolutionary computation 112

8.1 What is Machine Learning? 113

8.1.1 Definition **Machine Learning** is the science of getting computer to learn without being explicitly programmed. 113

8.2 Up to now 113

8.2.1 "learn without being explicitly" ! refine some predededefine more general solution scheme
- RF for regression → find a good forest
- SVM for binary classification → finnd a good hyperplane
We have some (quite precise) idea (*the hypothesis*).. What if we do not? 113

8.3 No hypothesis 113

8.3.1 We just have a way to assess a candidate solution
No hypothesis
Computer: be free, learn a (good) solution! (= program yourself!

How? A significant case:
problem: life
user:God?
computer: nature
learning method: natural evolution 113
8.4 Evolutionary process 114
8.4.1 A general and basic scheme:
- population of individuals compete for limited resources
- the population is dynamic: individuals die and are born
- fittest individual survive and reproduce more than the others
- offspring inherit some characters from parents (they are similar but not identical)

On/by/for computers? **Evolutionary computation** (EC) . . . 114
8.5 EC: a bit of history 115
8.5.1 1930s first ideas
1960s ideas development using first computers
1970s exploration
1980s exploitation
1990s unification
2000s+ mature expansion 115
8.6 Communities 115
8.6.1 At least three communities:
- biologists: simulate/understand real evolution
- computer scientists/engineers: build interesting artifacts
- artificial-life researchers: build/study artificial worlds

Result:
- some duplications
- different vocabularies
- strong habits
the same ideas proposed by people from different scientific communities studies are evaluated differently

Kenneth A De Jong. Evolutionary computation: a unified approach. MIT press, 2006(https://cs.gmu.edu/ kdejong/) from here on, the slides are inspired by this book: in fact, the book was intended to unify all these approaches in one 115
8.7 What can be taught/learned? 116

8.7.1 EC: evolutionary computation EA: evolutionary algorithm

Here:
- general scheme
-terminology
- some significant variants
-general usage guidelines

Not here:
- (variant) details
- detailed motivation ("theory")
- specific tools 116

8.8 General scheme 117

8.8.1 - a population of individuals compete for limited resources
- the population is dynamic: individuals die and are born
- fittest individual survive and reproduce more than the others (there are subjects favored in evolution)
- offspring inherit some characters from parents (they are similar but not identical)

Some questions (to introduce argument):
- what is an individual?
- what is a population? what are resources?
- how individuals compete?
- how fitness is measured?
- how do individual reproduce? 117

8.9 Individual 118

8.9.1 What is an individual? It is a possible solution to the problem I am dealing with.

A candidate solution for the considered problem: (**phenotype**=external representation)
- a program in a given programming language
- a set of numerical parameters
- a picture
- ...

Internally it could be represented differently

Internally represented as: (**genotype**)
- itself (program, set, picture,...)
- some well defined data structure:
- -a fixed/variable-length string of bits
- -an abstract syntax tree
-...

There must be some way to deterministically map a genotype with a phenotype 118
8.10 Individual: why genotype/phenotype? 119
8.10.1 To resemble nature
To ease manipulation
- - how two programs should reproduce?
- - how two images should reproduce?
- To allow reuse, hence enabling actual usage of EC
- -I someone found a good way of making bits strings reproduce
- user "just" need to decide how to transform (**genotype-phenotype mapping)**)a bits string to his/her solution form (e.g., numerical parameters) 119
8.11 Population and competition for resources 119
8.11.1 Mainstream:
- a population is a set of individuals with a fixed (max) size
"limited resources" is a place in the population

The population is dynamic:
when a new individual is born, some individual must leave the population (die): which one? 119
8.12 Population dynamics 120

8.12.1 How/when individuals are replaced? (**generational model**) or replacement strategy (is the definition of the way in which the population is dynamic)

Underlying (and common) assumptions:
- individuals life is instantaneous
- - given the genotype, the phenotype (if any) and the fitness are immediately known (appena nascono, si considera fissata la loro vita . Non può subire altre evoluzioni genetiche per esempio. Da quel momento può solo peggiorare.)
- When a new individual is born, some individual must leave the population (die): which one? 120

8.13 Generational model: general scheme 120

8.13.1 It has 3 parameters:
a population of m parents
a population of n offspring (built from parents; how? later)
a boolean flag (overlapping vs. non-overlapping)
(Recall: population size is fixed)

At every moment:
- the population is composed of m individuals who are all potentially parents
- after a step n children are generated.
- a Boolean flag (overlapping or not) But the population is fixed, that is = m 120

8.14 Overlapping generational model 121

8.14.1 At each time tick(of time):
1. build n offspring from the m parent
2. obtain an n + m population by merging parents and offspring
3. select m individuals to survive 121

8.15 Non-overlapping generational model 121

8.15.1 At each time tick:
1. build n offspring from the m parents
(assume $n \geq m$) parents die all, only m among the n children are selected and survive
All parents die! 121

8.16 Common cases 122

8.16.1 (typical combinations):
- $n = m$, overlapping
- $n = m$, non-overlapping
- $n = 0.8m$, overlapping
- $n = 1$, overlapping (steady state) (at every step a single individual is generated)

Problem:
- different degrees of dynamicity in the single time tick, makes different variants comparison difficult
Solution:
- measure time flowing as number of births referred to population size m
- a **generation** occurs each m births (time is measured in generations, every m birth spends a generation)

Which is the impact of parameters on population? LARGE IMPACT, usually approach are different, moreover is not easy to compare how fast evolution go on. Which is a good idea to compare different velocity? F.e. comparing the number of add / remove individuals 122
8.17 Selection criteria 123
8.17.1 It is one of the most important components of evolutionary computation.
How to
- select individuals to survive?
- select parents to reproduce?

Many options:
- 1. uniform (neutral) selection: with uniform probability peach an individual who will survive or reproduce
- 2. fitness-proportional selection: probabilistic extraction in proportion to fitness has been completed
- 3. rank-proportional selection
- 4. truncation selection
- 5. tournament selection
- 123
8.18 (2.& 3.) Fitness/rank-proportional 124
8.19 (1.& 4.) Uniform and truncation 125
8.19.1 Uniform:
1. pick randomly an individual (with uniform probability)

Truncation:
1. pick the best individual (**elitism**) (chooses exactly the best individual). It is Deterministic 125

8.20 (5.) Tournament selection 125

8.20.1 Given a parameter n_{size} (size of the tournament):
1. randomly (with uniform probability) pick n_{size} individuals
2. from them, choose the one with the best fitness
es. Randomly choose 5 (n_{size}) individuals with repetition (can be rescued who I have already fished) (among these 5 I choose the one with better fitness) 125

8.21 Selection criteria differences 125

8.21.1 Is criterion A better than criterion B? just measure

Criteria differ in how strongly they tend to prefer fit vs. unfit individuals:
- (1.) uniform selection: no preferences
- (4.) truncation selection: strong preference of fit individuals
- (5.) tournament: $n_{size} \rightarrow 1$: no preference, $n_{size} \rightarrow$m: strong preference
n_{size} determines the selectivity of the preference

Main impact of selection criteria is our preference in favoring fitter individual. In case of uniform selection we don't have preferences, in truncation selection we have strong preference to fittest individual, tournament is a criterion in which we can choose the preferences with the parameter nsize, so you can tune the preference. 125

8.22 Selecting fit/unfit individuals 126

8.22.1 Strong preference (or selective/evolutionary pressure):
- population tends to converge to fittest individuals
- evolution concentrates in improving most promising solutions (**exploitation**)
- risk of "falling" in local optimum

Weak preference (or selective/evolutionary pressure):
- population includes also unfit individuals
- evolution investigates many different (maybe not promising) solutions (**exploration**)
- risk of not finding a good solution

Exploration/exploitation trade-off is hard to rule! 126

8.23 Selectors: common cases 127

8.23.1 Reproduction: tournament of n_{size}. E.g. $m = n_{pop} = 500$, $n_{size} = 5$
Note than tournament can be applied also to non numerican fitness; we have just to decide which is the best individual.
- Survival: truncation
- Reproduction: fitness proportional
- Survival: truncation 127

8.24 Reproduction 128

8.24.1 Build n offspring from the m parents. How?
General scheme:
- given one or more parents, an offspring is generated by applying a unary or binary genetic operator on parent genotypes:
- - unary (**mutation**): $f : \mathcal{G} \to \mathcal{G}$
- - binary (recombination or (**crossover**): $f : \mathcal{G} \to \mathcal{G}$
- given n and a set of wighted operators, generate offspring with operators according to their weights (deterministically or stochastically)

I create a child applying a genetic operator that is a function that in the case of 1 parent (work in the domain: space of all genotypes, codomain: genotype space) binary genetic operator also called crossover -> from 2 individuals generates one. Instead mutation: from 1 parent it generates 1 128

8.25 Choice of operators 128

8.25.1 Operators:
- crossover for generating 80% of offspring
- mutation for generating 20% of offspring

Deterministically:
1. for 0:8n times - 1.1 select 2 parents (with reproduction selection criterion)
- 1.2 apply crossover to genotypes
2. for 0:2n times
- 2.1 select 1 parent (with reproduction selection criterion)
- 2.2 apply mutation to genotype

Stochastically:
1. for n times
- 1.1 randomly choose between mutation/crossover with 20/80 probability
- 1.2 select 1 or 2 parents (with reproduction selection criterion) accordingly
- 1.3 apply operator to genotype(s) 128

8.26 Mutation for bits string genotypes 129

8.26.1 Most classical option: probabilistic bit flip mutation
1. copy parent genotype g_p as child genotype g_c
2. for each bit in the in gc, flip it ($0 \rightarrow 1$ or $1 \rightarrow 0$) with p probability
Commonly, $p = 0.01$
change g_p to g_c through the random replacement of a string aid bit element

$g_p = 001010011101010101100100101$
$g_c = 001010\underline{1}1110101010110\underline{1}100101$ 129
8.27 Crossover for (bits) string genotypes 129
8.27.1 Many options:
- one-point crossover
- two-points crossover
- n-points crossover
- uniform crossover
- 129
8.28 One-, two-, n-points crossover 130
8.28.1 Assume parents with equal genotype size:
1. choose randomly one (two, n) cut points in the genotype (indexes i such that $i < |g_{p_1}| = |g_{p_2}|$
2. child bits before the cut point comes from parent 1, child bits after the cut point comes from parent 2
In general, jth bit comes from parent 1 iff closest larger cut point is even, from 2, otherwise. 130
8.29 Uniform crossover 131
8.29.1 One-point:
$g_{p_1} = 00101001110101010|1100100101$
$g_{p_2} =\underline{11101010101001010|0101110111}$
g$_c$ = $00101001110101010\underline{0101110111}$

$Two - points:$
g$_{p_1}$ = $0010100|1110101010|1100100101$
$g_{p_2} =\underline{1110101|0101001010|0101110111}$
g$_c$ = $0010100\underline{0101001010}1100100101$ 131
8.30 Uniform crossover 131
8.30.1 A cut point is placed at each index with $p = 0.5$ probability . . 131
8.31 Crossover with variable length (bits) string genotype 131
8.32 Mutation (trees) 134
8.32.1 1. choose a random subtree
2. replace with a randomly generated subtree 134
8.33 Crossover (trees) 134

8.33.1 first representation: 1) genotype as byte string 2) other representation: genotype is a given structure (as a random forest tree)

1. choose a random subtree in parent 1
2. choose a random subtree in parent 2
3. swap subtrees (child is copy of parent)

Usually, constraints on depth 134
8.34 Role of operators 135
8.34.1 Mutation (x)or crossover?
- mutation → exploitation
-crossover → exploration 135
8.35 Population initialization 135
8.35.1 Totally random
More specific approaches, dependent on genotype form 135
8.36 Fitness 135
8.36.1 the choice of fitness is the most important thing for the model (and for the success of the algorithm)

Fitness of an individual = ability to solve the problem of interest
- errors on several fitness cases by execution/simulation/application

Common cases:
- one numerical index
- more than one numerical indexes
- ...

Closely related with selectors 135
8.37 Many indexes: multiobjective 136
8.37.1 $f(i) = \langle f_1(i), ..., f_n(i)\rangle$

How to compare individuals i_1, i_2?
- linearization
- - $f(i) = \alpha_1 f_1(i) + ... + \alpha_n f_n(i)$ (Assign coefficient values: values that are higher, the greater the importance of the corresponding object)
- lexicographical order: compare
$f_1(i_1)?{>}f_1(i_2)$;
$f_2(i_1)?{>}f_2(i_2)$;...
- Pareto dominance:
- 136
8.38 Pareto dominance 136

8.38.1 this is one of the possible ways to compare objects with different characteristics

i_1 dominates i_2 iff: $\forall j, f_j(i_1) \geq f_j(i_2) \wedge \exists k, f_k(i_1) > f_k(i_2)$, it is no worse for every object and is at least larger for an object
- 1st Pareto front: undominated solutions
- 2nd Pareto front: undominated solutions, while not considering 1st front (forgetting the first frontier) 136

8.39 In practice 138

8.39.1 - Is my EA working?
- I When to stop evolution?
- I How to choose value for parameter X?
visualization tool (plot the fitness vs Generation):
chart where you put the highest value of the fitenss for individuals,
Shows the speed of convergence.
From an idea of the numbers of the phenomenon.
Oss. fitness measure chosen by me, so it may not be suitable. . 138

8.40 Issues 139

8.40.1 - Diversity
- Variational inheritance: children inherit some features from but not all parents
- Expressiveness: is the chosen representation able to represent a good solution
- 139

8.41 Diversity 139

8.41.1 Is the population diverse enough?
"No" → too much exploitation → local minimum
"yes" → in principle, no drawbacks
- how to measure diversity?
- how to enforce/promote diversity?
Diversity= related to diversity in population. In general high diversity is nt a problem. The point is that is not easy to measure diversity!!! And this measurement has effect to the solution. Also if we have a good measurement, there is a common problem of too low diversity 139

8.42 Variational inheritance 139

8.42.1 Are children similar but not identical to parents?
- "Too much similar" → too much exploitation → local minimum, no/slow evolution
"Too much different" → no exploitation,just coarse exploration (random walk)
Children should be similar but not equal to parents. If they are the same we have only exploitation (a way of no evolution) On the other extreme children are very dissimilar from parents, we randomly walk. There are no measurements but only some properties; locality, redundancy, degenerary, uniformity
- How to measure? (locality, redundancy, degeneracy, uniformity,...
How to tackle? Operators, mapping, both? 139
8.43 Expressiveness 139
8.43.1 Is the representation (phenotype) expressive enough?
- "Low expressiveness" → good/optimal solution might not be representable, or might not be reachable
- "Large expressiveness" → "Large expressiveness" → very long or infiniti convergence time

trade-off between low and high representation because that affect the convergence 139
8.44 Fitness landscape 139
8.44.1 How are genotype and fitness spaces related?
- What does a small step on one correspond to on the other? . . 139
8.45 Genetic Algorithms (GA) 139
8.45.1 First (of 3) family of algorithms

- Genotype = phenotype = bits string
- $m = n \approx 1000$, no overlapping
- Fitness-proportional selection, or multiobjective (Pareto-based) selection

- Most widely used/studied - Genotypes often encodes numerical parameters 139
8.46 Genetic Programming (GP) 141
8.46.1 (2 of 3) family of algorithms

Focus: individuals are programs
- Genotype = phenotype = tree (tree-based GP) or list of instructions (linear GP)
- $m = n \approx 1000$, overlapping
- Tournament selection
- Syntactic/semantic validity? 141
8.47 Grammatical Evolution (GE) 141

8.47.1 (3 of 3) family of algorithms

A form of GP based on GA, given a context-free grammar $\mathcal{G}$
- Genotype = bits string, phenotype = string $\in \mathcal{L}(\mathcal{G})$
We need a genotypical space with a map function that tells us if the strings are defined according to their grammatics
GA works with bit strings (works with operators, ...)
GE works on GA and obtains solutions as long as there is grammar on the grammar
- steady state ($m \approx 500, n = 1, overlaping$)
- Tournament selection 141

8.48 GE (standard) genotype-phenotype mapping 142

8.48.1 NO LOCALITY: small modification of genotypes lead to a large modification of phenotype
So there are other generalized methods 142

Bibliografia **143**