IS summary

Summary of the Intelligent Systems course lectures at FRI

ib8548

January 27, 2023

Contents

1 Nature inspired computing

1.1 Template of an evolutionary program

1. Generate a population of agents (objects, data structures)
2. Repeat:
   - 2.1 Compute **fitness** of the agents
   - 2.2 Select **candidates** for the reproduction (using fitness)
   - 2.3 Create new agents by **combining** the candidates
   - 2.4 **Replace** old agents with new ones
   - 2.5 Stop if satisfied

1.2 Key terms

- **Individual** — any possible solution
- **Population** — a group of all individuals
- **Search space** — all possible solutions to the problem
- **Chromosome** — a blueprint for an individual
- **Trait** — a possible aspect (features) of an individual
- **Allele** — possible settings of a trait
- **Locus** — the position of a gene on the Chromosome
- **Genome** — a collection of all chromosomes for an individual

1.3 Gene representation

- Bit vectors
- Numeric vectors
- Strings
- Permutations
- Trees (representing functions, expressions, programs)

1.4 Linear crossover

Let $\mathbf{x} = (x_1, \ldots, x_N)$ and $\mathbf{y} = (y_1, \ldots, y_N)$.
Select α in $(0, 1)$.
The result of the crossover is $\alpha\mathbf{x} + (1 - \alpha)\mathbf{y}$

1.4.1 Example

$$\alpha = 0.75$$
$$A = (5, 1, 2, 10)$$
$$B = (2, 8, 4, 5)$$

Crossover:

$$\begin{aligned}(\alpha * A_1 + (\alpha - 1) * B_1, \ldots, \alpha * A_4 + (\alpha - 1) * B_4) = \\ = (0.75 * 5 + 0.25 * 2, \ldots, 0.75 * 10 + 0.25 * 5) = \\ = (3.75 + 0.5, 0.75 + 2, 1.5 + 1, 7.5 + 1.25) = \\ = (4.25, 2.75, 2.5, 8.75)\end{aligned}$$

We can also choose a different α for each position. For example, using $\alpha = (0.5, 0.25, 0.75, 0.5)$ would result in $(3.5, 6.25, 2.5, 7.5)$.

1.5 Gray coding of binary numbers

Each number differs from the previous by **one bit**.

Binary	Gray
0000	**0000**
0001	**0001**
0010	**0011**
0011	**0010**
0100	**0110**
0101	**0101**
0110	**0100**
0111	**1100**
1001	**1101**
1010	**1111**
1011	**1110**
1100	**1010**
1101	**1011**
1110	**1011**
1111	**1000**

Constructing a n-bit Gray code recursively (example for $n = 2$):

- Create a (n-1) bit list: $\mathbf{0}, \mathbf{1}$
- Reflect: $\mathbf{1}, \mathbf{0}$
- Prefix old entries with 0: $\mathbf{00}, \mathbf{01}$
- Prefix new entries with 1: $\mathbf{11}, \mathbf{10}$
- Concatenate: $\mathbf{00}, \mathbf{01}, \mathbf{11}, \mathbf{10}$

1.6 Lamarckian mutation

- Searches for the locally best mutation

1.7 Gaussian mutation

- Selects a position in the vector of floats and mutates it by adding a Gaussian error

1.8 Selection

- Proportional
- Rank proportional
- Tournament
- Single Tournament
- Stochastic universal sampling

1.8.1 Proportional (roulette wheel) selection

Each individual gets a share on the "wheel" depending on its fitness. Fitter individuals get bigger shares.

1.8.2 Tournament selection

1. Set $\mathbf{t}$ = size of the tournament, $\mathbf{p}$ = probability of a choice
2. Randomly sample t agents from the tournament population
3. With probability p select the best agent
4. With probability $p(1-p)$ select the second best agent
5. With probability $p(1-p)^2$ select the third best agent
6. ...

The **n-th** fittest agent is therefore selected with the probability of $\mathbf{p(1-p)^{(n-1)}}$

1.9 Stochastic universal sampling (SUS)

- **Unbiased**
- Selecting N agents with combined fitness of F
- Randomly chosen first position $r \in [0, \frac{F}{N}]$
- The positions $r + i * \frac{F}{N}; i \in 0, 1, \ldots, N-1$ determine the chosen agents

1.10 Niche specialization

- Punish too similar agents

1.11 Stopping criteria

- Number of generations
- Progress
- Availability of computational resources
- ...

1.12 Parameters of genetic algorithms

- Encoding
- Length of the Strings
- Size of the pupolation (from 20–50 to a few thousand)
- Selection method
- Probability of performing a crossover (usually high — ~ 0.9)
- Probability of performing a mutation (usually low — below 0.1)
- Termination criteria (usually a number of generations or a target fitness)

1.13 Pros and cons of GA

Pros:

- **Low** time and memory **requirements** compared to searching a very large feature space
- A solution can be found without any explicit analytical work

Cons:

- Randomized and therefore not optimal
- Can get stuck on local maxima
- We have to figure out how to represent candidates with the available methods (e.g. as a bit string)

2 Predictive modelling

2.1 Learning

Acquiring new, modifiying and/or reinforcing existing:

- Knowledge
- Behaviors
- Skills
- Values
- Preferences

Statistical learning deals with finding a predictive function based on the data.

2.2 Notation

$$X = \begin{pmatrix} X_1 \\ X_2 \\ X_3 \end{pmatrix}$$
$$Y = f(X) + \varepsilon$$
$$Y_i = f(X_i) + \varepsilon$$

ε — measurement errors, its mean is 0 and it is independent from X.

2.3 Goals of learning

- Prediction
- Inference

2.4 Prediction

- If we have a good estimate for f, we can make accurate predictions for Y based on a new value of X.

2.5 Inference

- Finding out the relationship between Y and X.
- Sometimes more important than prediction (for example in medicine)

2.6 Parametric methods

- We estimate f by estimating the set of parameters

1. Come up with a model based on assumptions about the form of f, e.g. a linear model:

$$f(\mathbf{X_i} = \beta_0 + \beta_1 X_{i1} + \beta_2 X_{i2} + \cdots + \beta_p X_{ip})$$

   - More complicated and flexible models for f are often more realistic

2. Use the training data to fit the model (estimate f) — e.g. the least squares method in the case of linear models

2.7 Non-parametric methods

- They do not make assumptions about the funct. form of f. They fit a wider range of possible shapes, but require a large number of observations compared to parametric methods.

2.8 Types of learning

- Supervised
- Unsupervised
- Semi-Supervised
- Self-Supervised
- Weakly-supervised

2.8.1 Supervised learning

Both the predictors (X_i) and the response (Y_i) are given.

2.8.2 Unsupervised learning

We don't know Y — we are looking for similarities between attributes (X).

2.8.3 Semi-supervised learning

Only a small sample of labelled instances is observed, but a large set of instances is unlabelled.

- A supervised model is used to label unlabelled instances
- The most reliable predictions are then added to the training set for the next iteration of supervised learning

2.8.4 Self-supervised learning

- A mixture of supervised and unsupervised learning
- Learns from unlabelled data
- The labels are obtained **from related properties of the data**

Examples:

- in NLP: predicting hidden words in a sentence based on the remaining words
- in video processing: predicting past or future frames from the observed ones

2.8.5 Weakly-supervised data

Using imprecise or noisy sources to supervise labelling large amounts of training data, then performing supervised learning.
Example: using a smart electricity meter to estimate household occupancy

2.9 Criteria of success for machine learning

How to select the best model? Most popular criterions:

- For **regression**: mean squared error (MSE)

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - f'(x_i))^2$$

- For **classification**: classification accuracy (CA)

$$CA = \frac{1}{n}\sum_{i=1}^{n} I(y_i = y_i')$$

2.10 No free lunch theorem

If no information about $f(X)$ is provided:

- No classifier is better than some other in the general case
- No classifier is better than random in the general case

3 Bias, variance and predictive models

3.1 Bias

- It is the error that is introduced by modelling a (usually very complicated) real life problem by a much simpler problem.
- Example: linear regression assumes a linear relationship between Y and X; in reality that is unlikely to happen.
- Generally, the **more flexible/complex** a method is, the **less** bias it will have.

3.2 Variance

- Refers to how much our estimate for f would change if we had a different training data set
- Generally, the **more flexible** a method is, the **more** variance it has.

3.3 Bayes error rate

- Refers to the lowest possible error rate that could be achieved (if we knew the true probability distribution)
- On test data, no classifier can get lower error rates than this
- It can't be calculated exactly in real-life problems

3.4 Bayes optimal classifier

- For a new x_0, returns the maximally probable prediction value $P(Y = y|X = x_0)$
- In classification: $argmax_j P(Y = y_j|X = x_0)$

3.5 k-Nearest Neighbors (KNN)

- For any given X we find the closest k neighbors in the training data and examine their corresponding Y
- The **smaller** k, the **more flexible** the method will be

3.6 k-NN classifier

3.6.1 For classification

- $P(Y = j|X = x_0) = \frac{1}{K} \sum_{i \in \mathcal{N}_0} I(y_i = j)$

3.6.2 For regression

- $f(x) = \frac{1}{k} \sum_{x_i \in N_i} y_i$

3.7 Types of bias

- Reporting bias
- Automation bias
- Selection bias
- Group attribution bias
- Implicit bias

3.7.1 Reporting bias

- When the frequency of events does not accurately reflect their real-world frequency
- Because people tend to focus on documenting unusual or especially memorable circumstances

Example: The majority of book reviews are extreme opinions (very positive or very negative), because people were more inclined to giving a review if they had a strong opinion of the book

3.7.2 Automation bias

- The tendency to prefer automated systems over non-automated ones

Example: Engineers develop a new model for identifying product defects to relieve the burden off inspectors, but its results are found out to be worse than those of human inspectors

3.7.3 Selection bias

- Occurs if the dataset's examples are chosen irrepresentatively

3.7.4 Group attribution bias

- Generalizing properties of individuals to the entire group they belong to
  - In-group bias: you **also belong** to the group
    Example: Presuming that people who study at FRI are better at computer science than students of other computer science faculties
  - Out-group bias: you **don't belong** to the group
    Example: Making the presumption that people who did not attend a computer science academy do not have sufficient expertise in the field

3.7.5 Implicit bias

- Occurs when assumptions are made based on an individual's mental models and personal experiences that do not apply more generally

4 Feature selection

4.1 Types of feature selection methods

- **Filter methods:** independent of the learning algorithm, selecting the most discriminative features based on the character of data (information gain, ReliefF)
- **Wrapper methods:** using the intended learning algorithm to evaluate the features (e.g. progressively adding features while performance increases)
- **Embedded methods:** selecting features during the process of learning

4.2 Heuristic measures for attribute evaluation

- Impurity based — assuming conditional independence between the attributes
  - Information theory based (information gain, gain ratio, distance measure, J-measure)
  - Probability based (Gini index, DKM, classification error)
  - MDL
  - G, $\aleph^2$
  - Mean squared error (MSE), mean absolute error (MAE)
- Context-sensitive measures — affinity graph based
  - Relief, Contextual Merit
  - Random forests or boosting based evaluation

4.2.1 Information gain

$$I(\tau) = -\sum_{i=1}^{c} p(\tau_i) \log_2 p(\tau_i)$$

$$I(\tau|A) = -\sum_{j=1}^{v_A} p(v_j) \sum_{i=1}^{c} p(\tau_i|v_j) \log_2 p(\tau_i|v_j)$$

$$IG(A) = I(\tau) - I(\tau|A)$$

Measures the purity of labels before and after the split, with each attribute evaluated independently from the others.

4.2.2 Relief algorithms

- Criterion: evaluate attributes according to their power of separation between near instances
- Values of a good atribute should:
  - **Distinguish** between near instances from **different** class
  - Have **similar** values to near instances from the **same** class
- These algorithms take no assumptions about conditional independence, but are reliable also in problems with strong conditional dependencies
- Extension: **ReliefF**
  - Multi-class problems
  - Incomplete and noisy data
  - Robust

4.3 Ridge regression

- Ordinary Least Squares (OLS) minimizes:

$$RSS = \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij} \right)$$

- Ridge regression uses a slightly different equation — it minimizes the following expression:

$$RSS + \lambda \sum_{j=1}^{p} \beta_j^2$$

- This has the effect of "shrinking" large values of βs towards zero (we added a penalty term to the OLS equation)
- The penalty term **introduces bias** but can substantially **reduce variance** — there is a bias/variance tradeoff

4.4 The LASSO method

- Because the penalty term in ridge regression will never force any of the coefficients to be zero, using ridge regression wil cause the final model to include all variables, which makes it hard to interpret
- LASSO fixes this problem by using a different penalty term:

$$RSS + \lambda \sum_{j=1}^{p} |\beta_j|$$

- Using LASSO therefore enables us to produce a model that has high predictive power, but is simple to interpret

4.5 Wrapper approach

1. Start with an empty set of features $S = \emptyset$
2. Repeat until all features are added to S:
   - 2.1 Add all unused features one by one to S
   - 2.2 Train a prediction model with each set S
   - 2.3 Evaluate each prediction model
   - 2.4 Keep the best added feature in S
3. Return the best set of features encountered

4.6 Confusion matrix

A \ P	C	$\neg C$	
C	**TP**	**FN**	**P**
$\neg C$	**FP**	**TN**	**N**
	P'	**N'**	All

4.7 Model evaluation metrics

- Regression: MSE, MAE
- Classification: accuracy, sensitivity, specificity, AUC, precision, recall

4.7.1 Classification accuracy

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{All}}$$

The ratio of test set tuples that are correctly classified.

4.7.2 Sensitivity and specificity

Sensitivity: True Positive recognition rate

$$\text{Sensitivity} = \frac{\text{TP}}{\text{P}}$$

Specificity: True Negative recognition rate

$$\text{Sensitivity} = \frac{\text{TN}}{\text{N}}$$

4.7.3 Precision and recall

Precision: *exactness* — what percentage of positively classified tuples are actually positive

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

Recall: *completeness* — what percentage of positive tuples the classifier labelled as positive

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

There is an inverse relationship between precision and recall, the perfect score is 1.0

4.7.4 F-measures

F-measure (F_1 or F-score): a harmonic mean of precision and recall

$$F = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

$\mathbf{F}_\beta$: a weighted measure of precision and recall (assigns β times as much weight to recall as to precision)

$$F_\beta = \frac{(1 + \beta^2) \times \text{Precision} \times \text{Recall}}{\beta^2 \times \text{Precision} + \text{Recall}}$$

4.7.5 ROC curve

- Shows both TP rate and FP rate simultaneously
- To summarize overall performance, we also use *area under the ROC curve* (AOC)
- The larger the AUC, the better the classifier

4.8 Unsupervised feature selection

- Criterion: preserve similarity between instances
- SPEC: spectral feature selection

5 Ensemble methods

- Learn a large number of basic (simple) classifiers
- Merge their predictions

The most successful methods:

- Bagging
- Boosting
- Random forest

5.1 Bagging

- **B**ootstrap **agg**regat**ing**
- Solves the problem of decision trees suffering from high variance
- Based on:
  - Averaging — reduces variance
  - Bootstrapping — plenty of training datasets
- Procedure:
  1. Generate B different bootstrapped training datasets
  2. Train the statistical learning method on each of the B training datasets and obtain the prediction

5.1.1 Bootstrapping

- Resampling of the observed dataset (each dataset is of equal size to the observed dataset)
- Draw instances from a dataset **with replacement** (tuples may repeat)

5.1.2 Bagging for regression trees

1. Construct B regression trees using B bootstrapped training datasets
2. Average the resulting predictions

Averaging these trees reduces variance, thus we end up lowering both variance and bias

5.1.3 Bagging for classification trees

1. Construct B decision trees using B bootstrapped training datasets
2. For prediction, there are two approaches:
    - **Majority vote:** record the class that each bootstrapped data set predicts and provide an overall prediction to the most commonly occurring one
    - Average the probabilities and then predict to the class with the highest priority

5.2 Random forests

- A very efficient statistical learning method
- Builds on the idea of bagging, but provides an improvement by de-correlating the trees

1. Build a number of decision trees on bootstrapped training sample
2. When building these trees, each time a split in a tree is considered, a random sample of m predictors is chosen as split candidates from the full set of p predictors

Properties:

- Low classification/regression error
- No overfitting
- Robust
- Relatively fast
- Instances that are not selected with bootstrap replication are used for the evaluation of the tree (**OOB — Out-Of-Bag evaluation**)

5.3 Out-of-bag evaluation

- On average, $\sim$ 37% of the learning set is not used to train each of the basic classifiers

5.4 Boosting

- Grows the tree sequentially: each added tree uses information about errors of previous trees

5.5 Weighting of the trees

- Not all trees are equally important
- Weight the trees according to the data
- Assume linear combination of base coefficients

5.6 MARS — Multivariate Adaptive Regression Splines

- Generalization of stepwise linear regression
- Modification of trees to improve regression performance
- Able to capture additive structures
- Not tree-based
- Set C represents a candidate set of linear splines with "knees" at each data point X_i
- Models are built with elements from C or their products

$$C = \{(X_j - t)_+, (t - X_j)_+\}_{t\ni} x_{1j}, x_{2j}, \ldots, x_{Nj}, j = 1, 2, \ldots, p$$

- Model form:

$$f(X) = \beta_0 + \sum_{m=1}^{M} \beta_m h_m(X)$$

Procedure:

1. Given a choice for the h_m, the coefficients β are chosen by the standard linear regression
2. Start with $h_0(X) = 1$; all functions in C are candidate functions
3. At each stage, consider as a new basis function pair all products of a function h_m in the model set M, with one of the reflected pairs in C

$$\beta_{M+1} h_l(X) * (X_j - t)_+ + \beta_{M+2} h_l(X) * (t - X_j)_+, h_l \in M$$

4. We add to the model terms of the form:

$$h_m(X) * (t - X_j)_+ \qquad h_m(X) * (X_j - t)_+$$

6 Kernel methods

6.1 Support vector machines

- Idea: finding a hyperplane that gives the biggest separation between the classes
- In practice, it is usually not possible to find a hyperplane that perfectly separates two classes
- We try to find the best separation

6.1.1 Non-linear support vector classifier

$$Y_i = \beta_0 + \beta 1 b_1(X_i) + \beta_2 b_2(X_i) + \cdots + \beta_p b_p(X_i) + \varepsilon_i$$

6.1.2 Example

- Suppose we use polynomials as bases

$$X_1,\ X_2,\ X_1^2,\ X_2^2,\ X_1X_2,\ X_1^3,\ X_1X_2^2,\ \ldots$$

- We go from p-dimensional space to $M > p$-dimensional space and fit the SVM classifier in the enlarged space
- For the bases $(X_1,\ X_2,\ X_1^2,\ X_2^2,\ X_1X_2)$, this gives a non-linear classifier in the original space

$$\beta_1 X_1 + \beta_2 X_2 + \beta_3 X_1^2 + \beta_4 X_2^2 + \beta_5 X_1 X_2 = 0$$

6.1.3 Common kernel functions

Instead of choosing the basis, we use a kernel function. Common kernel functions include:

- Linear
- Polynomials
- Radial basis
- Sigmoid

6.1.4 SVM for more than one class

- OVA — One versus All
  - Fit K different 2-class SVM classifiers, each class versus the rest
  - Classify new x to the class for which the value of the kernel function is the largest.

- OVO — One versus One
  - Fit all $\binom{K}{2}$ pairwise classifiers
  - Classify new x ti the class that wins the most pairwise competitions
- If K is not too large, use OVO

7 Neural networks

7.1 Neuron

- Computational units
- Passing messages (information) in the network
- Typically organized into layers

7.2 Activation functions

- Step function: $f(x) = \begin{cases} 1, & x > 0 \\ 0, & x \leq 0 \end{cases}$
- Sigmoid (logistic) function: $f(x) = \frac{1}{1+e^{-x}}$
- ReLU (rectified linear unit): $f(x) = \max(0, x)$
- Softplus / approximation of ReLU with continuous derivation: $f(x) = \ln(1 + e^x)$
- ELU (Exponential Linear Unit): $ELU(x) = \begin{cases} c * (e^x - 1) & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$

7.3 Multi-layeres NNs

1. The **inputs** to the network correspond to the attributes measured for each training tuple
2. Inputs are fed simultaneously into the units making up the **input layer**
3. They are then **weighted** and fed simultaneously to a **hidden layer**
4. The number of hidden layers is arbitrary; if more than 1 hidden layer is used, the network is called a **deep neural network**
5. The weighted outputs of the last hidden layer are inputs to units making up the **output layer**, which emits the network's prediction
6. The network is **feed-forward**: none of the weights cycle back to an input unit or to an output unit of a previous layer
7. If we have backwards connections, the network is called a **recurrent neural network**
8. From a stytistical point of view, networks perform **non-linear regression**: given enough hidden units and enough training samples, they can closely approximate any function

7.4 Backpropagation learning algorithm for NN

- Backpropagation: a neural network learning algorithm
- Started by psychologists and neurobiologists to develop and test computational analogues of neurons
- During the learning phase, the network learns by adjusting the weights so as to be able to predict the correct class label of the input tuples
- Also referred to as **connectionist learning** due to the connections between units

7.5 Softmax

- Normalizes the output scores to be a probability distribution (values between 0 and 1, the sum is 1)

$$y_i = \frac{e^{z_i}}{\sum_{j \in \text{group}} e^{z_j}}$$

$$\frac{\partial y_i}{\partial z_i} = y_i(1 - y_i)$$

7.6 Criterion function

- Together with softmax we frequently use cross entropy as the cost function C

$$C = -\sum_j t_j \log y_j$$

$$\frac{\partial C}{\partial z_i} = \sum_j \frac{\partial C}{\partial y_j} \frac{\partial y_j}{\partial z_i} = y_i - t_i$$

7.7 Backpropagation

- Iteratively process a set of training tuples and compare the network's prediction with the actual known target value
- For each training tuple, the weights are modified to **minimize the mean squared error** between the network's prediction and the actual target value
- Modifications are made in the "**backwards**" direction: from the output layer, through each hidden layer down to the first hidden layer (hence "backpropagation")
- Steps:
    1. Initialize weights to small random numbers, associated with biases

2. Propagate the inputs forward (by applying activation function)
3. Backpropagate the error (by updating weights and biases)
4. Terminating condition: when the error is very small, etc.

7.8 Error backpropagation

- Performing gradient descent on the whole network
- Training will procees from the last layer to the first
- introduce variables over the neura network

$$\vec{\theta} = \{w_{ij}, w_{jk}, w_{kl}\}$$

  - Distinguish the input and output of each node

7.9 Autoencoders

- Designed to reproduce their input, especially for images
- The key point is to reproduce the input from a learned encoding
- The loss function is the reproduction error

7.9.1 Structure

- **Encoder:** compresses input into a latent-space of usually smaller dimension. $h = f(x)$
- **Decoder:** reconstructs input from the latent space. $r = g(f(x))$ with r as close to x as possible

7.9.2 Denoising autoencoders

- Basic autoencoders train to minimize the loss between x and the reconstruction, $g(f(x))$
- Denoising autoencoders train to minimize the loss between x and $g(f(x + w))$, where w is random noise
- Same possible architectures, different training data

7.9.3 Properties of autoencoders

- **Data-specific:** autoencoders are only able to compress data similar to what they have been trained on
- **Lossy:** the decompressed outputs will be degraded compared to the original inputs
- **Learned automatically from examples:** It is easy to train specialized instances of the algorithm that will perform well on a specific type of input

7.10 Generative Adversarial Networks (GANs)

- Learn a generative model
- Are trained in an adversarial setting
- Use deep neural networks

7.11 Adversarial training

- **Generator:** generates fake samples and tries to fool the discriminator
- **Discriminator:** tries to distinguish between real and fake samples
- We train them against each other
- By repeating the training process, the generator and discriminator improve

8 Inference and explanation of prediction models

8.1 Visualization

- Helps us understand the data we are working with
- Due to limitations of human visual perception, we often see what we want to see, therefore it is useful to use visualizations which expose "the unknown"

8.2 Explanation of predictions

- A number of successful prediction algorithms exist (SVM, boosting, random forests, neural networks), but they are a **black box** to the user
- There are many fields where users are concerned about the transparency of the models (medicine, law, consultancy, ...)

8.3 Domain level explanation

- Trying to explain the "true causes and effects"
- Usually unreachable
- Some aspects are covered with attribute evaluation, detection of redundancies, ...
- Targeted indirectly through models

8.4 Model-based explanations

- They make the prediction process of a particular problem transparent
- The explanation correctness is independent of the correctness of the prediction
- Better models enable (in principle) better explanations at the domain level
- We are mostly interested only in the explanation at the model level

8.5 Types of explanation techniques

- Model-specific
  - Especially used for deep neural networks
- Model-agnostic
  - Can be used for any predictor
  - Based on perturbation of the inputs

8.6 Perturbation-based explanations

- Importance of a feature or a group of features in a specific mode can be estimated by simulating lack of knowledge about the values of the feature(s)

8.7 Instance-level explanations

- They explain prediction for each instance separately
  - This is what **practitioners** applying the models are interested in
- Model-based

8.8 Model-level explanations

- They show the overall picture of a problem the model conveys
  - This is what **knowledge extractors** are interested in
- Model-based

8.9 The EXPLAIN method

- "Hide" one attribute at a time
- Estimate the contribution of the attribute:

$$p(y_k|x) - p_{S\setminus\{i\}}(y_k|x)$$

- Assume an instance $(\mathbf{x}, y)$, where components of $\mathbf{x}$ are values of attribute A_i
- For a new instance $\mathbf{x}$, we want to know what role each attribute's value plays in the prediction model f
- For that purpose,
  - We compute $f(\mathbf{x}\setminus A_i)$, the model's prediction for $\mathbf{x}$ without the knowledge of the event $A_i = a_k$ (marginal prediction)
  - We compare $f(\mathbf{x})$ and $f(\mathbf{x}\setminus A_i)$ to assess the importance of $A_i = a_k$
  - The larger the difference, the more important the role of $A_i = a_k$ in the model
- $f(\mathbf{x})$ and $f(\mathbf{x}\setminus A_i)$ are the source of explanations

8.10 Evaluation of prediction differences

1. Difference of probabilities

$$\text{probDiff}_i(y|\mathbf{x}) = p(y|\mathbf{x}) - p(y|\mathbf{x}\backslash A_i)$$

2. Information gain

$$\text{infGain}_i(y|\mathbf{x}) = \log_2 p(y|\mathbf{x}) - \log_2 p(y|\mathbf{x}\backslash A_i)$$

3. Weight of evidence (also log odds ratio)

$$odds(z) = \frac{p(z)}{1 - p(z)}$$

$$WE_i(y|\mathbf{x}) = \log_2 odds(y|\mathbf{x}) - \log_2 odds(y|\mathbf{x}\backslash A_i)$$

8.10.1 Implementation

- $p(y|\mathbf{x})$: classify $\mathbf{x}$ with the model
- $p(y|\mathbf{x}\backslash A_i)$: simulate lack of knowledge of A_i in the model
  - Replace with special NA value: good for some, but mostly bad
  - Average prediction across perturbations of A_i

$$p(y|\mathbf{x}\backslash A_i) = \sum_a p(A_i = a_s) p(y|\mathbf{x} \leftarrow A_i = a_s)$$

  - Use discretization for numeric attributes
  - Use Laplace correction for probability estimation

8.11 The IME method

- Interactions-based Method for Explanation
- "Hide' any subset of attributes at a time (2^a subsets!)
- The **source of explanations** is the difference in prediction using a subset of features Q and an empty set of features $\emptyset$

$$\Delta Q = h(x_Q) - h(x_\emptyset)$$

- The feature gets some credit for standalone contributions and for contributions in interactions

8.12 The LIME explanation method

- Optimize a trade-off between local fidelity of explanation and its interpretability
- LIME samples around the explanation instance x to draw samples z weighted by the distance $\pi(x, z)$
- Samples z are used to train an interpretable model g (linear model)
- Faster than IME

8.13 The SHAP method

- SHapley Additive exPlanation
- Unification of several explanation methods, including IME and LIME
- Faster than IME, but still uses a linear model with all its strengths and weaknesses

9 Natural language processing

9.1 Uses

- Speech recognition and synthesis
- Automatic reply Engineers
- Machine translation
- Text summarization
- Question answering
- Language generations
- Interface to databases
- Intelligent search and information extraction
- Sentiment detection
- Semantic analysis
- Named entity recognition and linking
- Categorization, classification of documents, messages, etc.
- Many tools and language resources
- Prevalence of deep neural network approaches
- Cross-lingual approaches

9.2 Linguistic analysis

- **Prosody** — the patterns of stress and intonation in a language
- **Phonology** — systems of sounds and relationships among the speech souds that constitute the fundamental components of a language
- **Morphology** — the admissin+ble arrangements of sounds in words; how to form words, prefixes and suffixes...
- **Syntax** — the arrangement of words and phrases to reate well-formed sentences in a language
- **Semantics** — the meaning of a word, phrase, sentence or text
- **Pragmatics** — language in use and the contexts in which it is used, taking turns in conversation, text organization, presupposition and implicature
- Knowing the world: knowledge of the physical word, humans, society, intentions in communications...

9.3 The classic approach

0. Text preprocessing
1. Syntactic analysis
2. Semantic interpretation
3. Use of world knowledge

9.4 Lemmatization and stemming

- **Lemmatization:** the process of grouping together the different forms of a word into a single item
- The stemmer operates on a single word without knowledge of its context
- Lemmatization difficulty is language dependent
- Usage of rules and dictionaries

9.5 Part of speech (POS) tagging

- Assigning the correct part of speech (noun, verb, ...) to words
- Helps recognize phrases and names
- Uses rues and machine learning models

9.6 Named entity recognition (NER)

- Recognizing named entities, such as names of organizations (e.g. NATO), people (e.g. Jens Stoltenberg), countries ...
- Named entity linking (NEL) links same words with different meaning (e.g. jaguar, Paris) to an unique identifier (wikification)

9.7 Syntax analysis

- **The 1. phase of text understanding**
- POS tagging
- Determining the words' roles in the sentence (subject, object, predicate)
- The result is mostly presented in a form of a parse tree
- Syntax, morphology and some semantics are needed

9.8 n-gram tagging

- Contect of $n-1$ preceding words
- Corpus-based learning
- Markov models, HMM, learning with EM

$$t_i = \arg\max_j P(t^{(j)}|t_{i-1}) * P(w_i|t^{(j)})$$

9.9 Grammars

- Many tools, e.g. NLTK
- Ambiguity, several parsing trees

9.10 Interpretation

- **The 2. phase of text understanding**
- Knowledge of word meaning and their language use
- Result: conceptual graphs, frames, logical program
- Check semantics

9.11 Use of world knowledge

- **The 3. phase of text understanding**

9.12 Document retrieval

- Historical: keywords
- Now: whole text search
- Organize a database, indexing, search algorithms
- Input: a query

9.13 Document indexing

- Collect all words from all documents **(use lemmatization)**
- Inverted file
- For each word keep:
  - The number of appearing documents
  - Overall number of appearances
  - For each document:
    - Number of appearances
    - Location

9.14 Ranking-based search

- Web search
- Less frequent terms are more informative
- NL input — stop words, lemmatization
- Vector-based representation of documents and queries (bag-of-words and dense embeddings)

9.15 Vectors and documents

- A word occurs in several documents
- Both words and documents are vectors
- A term-document matrix of dimensions $|V| \times |D|$
- A sparse matrix
- Word embedding

9.16 Document similarity

- Assumes orthogonal dimensions
- Dot product of vectors

$$\cos(\Theta) = \frac{A\dot{B}}{|A||B|}$$

9.17 Importance of words

- Frequencies of words in a particular document (and overall)
- Inverse document frequency (IDF):

$$\text{idf}_b = \log\left(\frac{N}{n_b}\right)$$

  - N — number of documents in collection
  - n_b — number of documents with word b

9.18 Weighting dimensions

- The weight of word b in document d

$$w_{b,d} = \text{tf}_{b,d} \times \text{idf}_{b,d}$$

  - $\text{tf}_{b,d}$ — frequency of term b in document d
- Called **TF_IDF weighting**

9.19 Performance measures for search

- Statistical measures
- Subjective measures
- Precision, recall
- A contingency table analysis of precision and recall

9.19.1 Precision and recall

- N — number of documents in the collection
- n — number of inpportant documents for a given query q
- Search returns m documents, including a relevant ones
- **Precision:** $P = \frac{a}{m}$ — the proportion of relevant documents in the obtained ones
- **Recall:** $R = \frac{a}{n}$ — the proportion of obtained relevant documents

9.19.2 Other ranking measures

- **Precision@k** — the proportion of relevant documents in the first k obtained ones
- **Recall@k** — the proportion of relevant documents in the k obtained among all relevant documents
- F_1@k
- Mean reciprocal rank:

$$MRR = \frac{1}{Q} \sum_{i=1}^{Q} \frac{1}{rank_i}$$

  - Over Q queries
  - Considers only the rank of the best answer

9.20 PageRank

- p — a web page
- $O(p)$ — pages pointed to by p
- $I(p) = \{i_1, i_2, \ldots, i_n\}$ — pages pointing to p
- d = damping factor between 0 and 1 (default 0.85 or 0.9)
- $\pi(p)$ — page quality, depends on the quality of pages pointing to it

$$\pi(p) = (1-d) + d\frac{\pi(i_1)}{|O(i_1)|} + \cdots + d\frac{\pi(i_n)}{|O(i_n)|}$$

9.21 Dense vector embeddings

- Advantages compared to sparse embeddings:
  - Less dimensions, less space
  - Easier input for ML methods
  - Potential generalization and noise reduction
  - Potentially captures synonims
- The most popular approaches:
  - Matrix-based transformations to reduce dimensionality (SVD and LSA)
  - Neural embeddings (word2vec, Glove)
  - Contextual neural embeddings

9.22 The word2vec method

- Instead of **counting** how often each word w occurs near "apricot", train a classifier on a binary **prediction** task (is w likely to show up near "apricot"?)
- Take the obtained (learned) classifier weights as the **word embeddings** (groups of similar words)

9.23 Contextual embeddings

- Also take context into account
- ELMo, BERT

9.23.1 ELMo

- Looks at the entire sentence before assigning each word in it an embedding
- Predicts **the next word** in a sequence
- Includes subword units

9.23.2 BERT

- Predicts **masked words** in a sentence
- Also predicts the order of sentences

9.24 Text summarization

- Evaluation:
  - ROUGE scores
  - BERTScore
  - with QA: question generation and searching for answers in the summary
- Deep neural networks
- Short and long texts

9.25 Sentiment analysis (SA)

- A computational study of opinions, sentiments, emotions and attitude expressed in texts towards an entity

10 Reinforcement learning

10.1 Procedural learning

- Learning how to act to accomplish goals (given an environment that contains rewards, learn a policy for acting)
- You have to try things in order to learn

10.2 Agent

- Temporally situated
- Continuous learning and planning
- Tries to affect the stochastic and uncertain environment

10.3 Key features of reinforcement learning

- The learner is not told which actions to take
- Trial and error
- The need to **explore** and **exploit**

10.4 Components of RL

Policy $\subset$ Reward $\subset$ Value $\subset$ Model of environment

- **Policy:** what to do?
  - Defines the actions and choices
  - Represented with rules, tables, NNs, ...
- **Reward:** what is good?
  - Feedback from the environment
- **Value:** an internal representation of what is good, it *predicts* the reward
  - The agent's expectation of what can be expected in a given state (long-term)
  - Implicitly contains evaluations of next states
  - Has to be learned
- **Model:** what follows what
  - An internal representation of the environment
  - Enables the agent to evaluate values and actions without performing them
  - Optional

10.5 Types of RL

- Search-based: evolution directly on a policy
- Model-based: build a model of the environment
- Model-free: learn a policy without any models

10.5.1 Types of model-free RL

- Actor-critic learning
- Q-learning