VelGagnNotes

liljag18

Fall 2020

1 Introduction

This is my Machine Learning notebook.

> Adapt to new circumstances and to deteect and extrapolate patterns

2 Exploring Data

2.1 Data

Data is a collection of data and attributes (a property or characteristic of an object), collection of attributes describe an object.
Attribute values

- Attribute values are numbers or symbols assigned to an attribute for a particular object
- Attributes are things like refund, height, etc (like categories)
- Same attribute can be mappeed to different attribute values
- Different attributes can bee mapped to the same set of values

Properties of Attribute Values

- Distinctness: $=\neq$
- Order: $<>$
- Differences are meaningful: $+-$
- Ratios are meaningful: $*/$

Types of Attributes

- Nominal - ID numbers, eye color, zip codes; Distinctness
- Ordinal - rankings, grades, height tall, medium, short; Distinctness and order
- Interval - calendar dates, temperatures in Celcius or Farenheit; Distinctness, order, and meaningful differences
- Ratio - temperature in Kelvin, length, time, counts; Distinctness, order, meaningful differences, and ratios are meaningful

	Categorcial Qualitative		Numeric Quantitative	
Attribute type	Nominal	Ordinal	Interval	Ratio
Description	Only distinguish	Also order objects	differences between values are meaningful	differences and ratios are meaningful
Examples	zip codes, employee ID numbers, eye color, sex	Hardness of minerals good, better, best, grades, street numbers	Calendar dates, temp in Celcius or Farenheit	Temp in Kelvin, monetary quantities, counts, age, mass, length, current
Operations	mode, entropy, contingency correlation	median, percentiles, rank correlation, run tests, sign tests	mean, standard deviation, Pearson's correlation, t and f tests	geometric mean, harmonic mean, percent variation
Transformation	Any permutation of values	An order perserving change of values, new_value = f(old_value) where f is a monotonic function	new_value = a*old_value + b where a and b are constants	new_value = a * old_value
Comments	If all employee ID numbers are reassigned would it make any difference	An attribute encompassing the notion of good, better, best can be represented equally well by the values 1,2,3	Thus the farenheit and celsius temp scales differ in terms of where their zero value is and the size of a unit (degree)	Length can be measured in meters or feet

Discrete attribute:

- Has only a finite or countably infinite set of values
- Zip codes, counts, or the set of words in a collection of documents
- Often represented as integer variables
- Binary attributes are a special case of discrete attributes

Continuous Attribute

- Has real numbers as attribute values
- Temperature, height, weight
- Practically, real values can only be measured and represented using a finite number of digits
- Continuous attributes are typically represented as flooating point variables

Asymmetric Attributes - only presence (a none-zero attribute value) is regarded as important, typicaly arise from objects that are sets.

Critiques

Incomplete: Asymmetric binary, cyclical, multivariate, partially ordered, partial membership, relationships between the data

Real data is approximate and noisy: This can complicate recognition of the proper attribute type, treating one attribute type as another may be approximately correct

Not a good guide for statistical analysis: May unnecessarily restrict operations and result (statistical analysis is often approximate), transformations are common but don't preserve scales (can transform data to a new scale with better statistical properties, many statistical analysis depend only on the distribution)

Example

ID numbers - nominal

Number of cylinders in an automobile engine - ratio (never less than 0)

Biased scale - interval (biased and no longer correct)

Key Messages for Attribute Types

- type of operations you choose should be "meaningful" for the type of data you have
- distinctness, order, meaningful intervals, and meaningful ratios are only four properties of data
- data type you can see (often numbers or strings) many not capture all properties or may suggest properties that are not there
- Analysis may depend on these other properties of the data
- Many times what is meaningful is measured by statistical significance
- But in the end, what is meaningful is measured by the domain

Important Characteristics of Data

- Dimensionality (number of attributes) - high dimensional data brings a number of challenges
- Sparsity - only presence counts
- Resolution - patterns depend on the scale
- Size - type of analysis may depend on size of data

Types of data sets

- Record (Data Matrix, Document Data, Transaction Data)

  - Record - data that consists of a collection of records, each of which consists of a fixed set of attributes
  - Data Matrix - if data objects have the same fixed set of numeric attributes, then the data objects can be thought of as points in a multi-dimensiona space, where each dimension represents a distinct attribute; such data can be represented by an m by n matrix where there are m rows, one for each object, and n columns, one for each attribute
  - Document Data - each document becomes a 'term' vector, each term is a component (attribute) of the vector, the value of each component is the number of times the corresponding term occurs in the document
  - Transaction Data - special type of record data where each record (transaction) involves a seet of items (like in a grocery store)
- Graph (World Wide Web, Molecular Structures) - generic graph, molecule, webpages
- Ordered (Spatial Data, Temporal Data, Sequential Data, Genetic Sequence Data) - sequences of transactions, genomic sequence of data, spatio-temporal data

Data quality problems - noise and outliers, missing values, duplicate data, wrong data

Noise - for object it is an extraneous object, for attributes it refers to modification of original values
Outliers - data objects with characteristics that are considerably different than most of the other data objects in the data set (noise that interferes with data analysis or are the goal of our analysis)
Missing Values - could be because info is not collected or attributes may not be applicable to all cases; handling missing values: eliminate data objects or variables or missing values or ignore the missing value during analysis

- Missing completely at random (MCAR) - missingness of a valuee is independent of attributes, fill in values based on the attribute, analysis may be unbiased overall
- Missing at Random (MAR) - Missingness is relateed to other variables, fill in values based on other values, almost always produces a bias in the analysis
- Missing Not at Random (MNAR) - Missingness is related to unobserved measurements, informative or non-ignorable missingness
- Not possible to know the situation from the data

Duplicate Data - data set may include data objects that are duplicates or almost duplicates of one another - major issue when merging data from heterogeneous sources (same person with multiple email addresses)

Aggregation - combination two or more attributes (or objects) into a singe attribute (or object). The purpose of this is data reduction (reduce number of attributes or objects), change of scale (cities ggregatd into regions, states, countries), more stable data (aggregated data tends to have less variability.

Sampling - main technique emplyed for data reduction (often used for both preliminary investigation of the data and final data analysis). Statiticians often sample because obtaining the entire set of data of interest is too expensive or time consuming. Typically useed in data mining because processing the entire set of data of interest is too expensive or time consuming. Sample must be **representative** by having thee same properties as the original set of data

Types of Sampling

- Simple Random Sampling
  - There is an equal probability of selecting any particular item
  - Sampling without replacement: as each item is selected it is removed from the population
  - Sampling with replacement: objects are not removed from the population as they are selected for the sample, same object can be picked up more than once
- Stratified sampling - split the data into several partitions, then draw random samples from each partition

Increase in dimenstionality == data becomes increasingly sparse in the space it occupies

Dimensionailty Reduction

- Purpose: Avoid curse of dimensionality, reduce amount of time and memory required by data mining algorithms, allow data to be more easily visualized, may help to eliminate irrelevant features or reduce noise
- Techniques:
  - Principal Components Analysis (PCA) - Goal is to find a projection that captures the largest amount of variation in data
  - Singular Value Decomposition
  - Other: supervised and non-linear techniques

Feature Subset Selection

- Another way to reduce dimensionality of data

- Redundant features: duplicate much or all of the information contained in one or more other attributes
- Irrelevant features: contain no info that is useful for the data mining task at hand
- Many techniques developed, especially for classification

Feature Creation - create new attributes that can capture the important information in a data set much moree efficiently than the original attributes (Featuree extraction, Feature construction, Mapping data to a new space)

Discretization - the proceess of converting a continuous attribute into an ordinal attribute

- potentially infinite number of values are mapped into a small number of categories
- Discrestization is commonoly used in classification
- many classification algorithms work best if both the independent and dependent variables have only a few values
- we give an illustration of the usefulness of discretization using the Iris data set
- Unsupervised discretization: find breaks in the data values
- Supervised discretization: use class labels to find breaks

Binarization - maps a continuous or categorical attribute into one or mor ebinary variables, typically used for association analysis, often convert continuous attribute to a categorical attribute and then convert a categorical attribute to a set of binary attributes

Attribute Transformation - function that maps the entire set of values of a given attribute to a new set of replacement values such that each old value can be identified with one of the new values

Normalization - refers to various techniques to adjust to differences among attributes in terms of frequency of occurrence, mean, variance, range; take out unwanted, common singal.

Standardization - refers to subtracting off the means and dividing by the standard deviation

2.2 Data Mining

Data exploration to select the right tool for preprocessing or analysis, makes use of human abilities to recognize patterns.

Exploratory Dataa Analysis (EDA)

- Focus on visualization
- Clustering and anomaly detection were viewed as exploratory techniques
- In data mining, clustering aand anomaly detection are major areas of interest and not thought of as just exploratory

Summary Statistics - numbers that summarize properties of the data.
The **frequency** of aan attributee value is the percentaage of time the value occurs in the dataset.
The **mode** of an attributee is the most frequent attribute value.
Notions of frequency and mode are typically used with categorical data.
For conitnuous data the notion of a **percentile** is more useful.
The **mean** is the most common measure of the location of a set of points (very sensitive to outliers though).
Median, or trimmed mean, is also commonly used.
Range is the difference between the max and min.
Variance (standard deviation) is the most common measure of the spread of a set of points, sensitive to outliers.

Visualization - conversion of data into a visual or tabular format so that characteristics of the data and the relationships among data items or attributes can be analyzed or reported.
Representation - mapping of info to a visual format, data objects their attributes and the relationships among data objects are translated into graphical elements such as points lines shapes and colors.
Arrangement - placement of visual elements within a display, can make a large difference in how easy it is to understand data.
Selection - elimination or the de-emphasis of certain objects and attributes, may involve choosing a subset of attributes or objects.

Vizualization techniques

- Histogram - show distribution of values of single variable
- 2D Histogram - show joint distribution of value of 2 attributes
- Box Plots - another way of displaying distribution of data
- Scatter Plots - attribute values determine position
- Contour Plots - useful when continuous attribute is measured on a spatial grid

- Matrix Plots - plots the data matrix (turns numbers into colors)
- Parallel Coordinates - plot the attribute values of high dimensional data, each object represented as a line, see relationship between attributes

3 Supervised Learning

Supervised Learning: learning a function from input/output pairs

3.1 Decision Trees

Classification - given a collection of records (training set) each record is characterized by a tuple (x,y) where x is the attribute set and y is the class label.

Task	Attribute set, x	Class label, y
Categorizing email messages	Features extracted from email message header and content	spam or non-spam
Identifying tumor cells	Features extracted from MRI scans	malignant or benign
Cataloging galaxies	Features extracted from telescope images	Elliptical, spiral, or irregular-shaped galaxies

Base Classifiers - Decision Tree based Methods, Rule-based Methods, Nearest-neighbor, Neural Networks, Deep Learning, Naive Bayes and Bayesian Belief Networks, Support Vector Machines.
Ensemble Classifiers - Boosting, Bagging, Random Forests.

3.1.1 Hunt's Algorithm

Decision tree is grown in recursive fashion by partitioning the training records into successively purer subsets. If all records looked at belong to the class being looked at it becomes a leaf node.

Design Issues of Decision Tree Induction

- How training records should be split - method for specifying test condition (depends on attribute types), measure for evaluating the goodness of a test condition
- When splitting procedure should stop - stop splitting if all the records belong to the same class or have identical attribute values, early termination

	Multi-way split	Binary split
Nominal Attributes	Use as many partitions as distinct values	Divides values into two subsets
Ordinal Attributes	Use as many partitions as distinct values	Divides values into two subsets, preserve order property among attribute values
Continuous	Discretization: to form an ordinal categorical attribute (static - discretize once at the beginning, dynamic - repeat at each node	Consider all possible splits and find the best cut, can be more compute intensive

3.1.2 Greedy approach to finding best split

- nodes with purer class distribution are preferred.

Equations for best split

- Gini Index - $GINI(t) = 1 - \sum_j [p(j|t)]^2$
- Entropy - $Entropy(t) = -\sum_j p(j|t) log p(j|t)$
- Misclassification error - $Error(t) = 1 - maxP(i|t)$

Finding Best Split

1. Compute impurity measure (P) before splitting
2. Compute impurity measure (M) after splitting - compute impurity measure of each child node, M is the weighted impurity of children
3. Choose the attribute test condition that produces the highest gain **Gain = P-M** or equivalently, lowest impurity measure after splitting (M)

Example of Gini for 2-class problem: If C1 = 1 and C2 = 5 we do $GINI = 1 - p^2 - (1-p)^2 = 2p(1-p) = 2(1/6)(1-1/6) = 0.278$ where p = probability of C1 which is 1/6 (get same result with probability of C2).

$$GINI(t) = 1 - \sum_j [p(j|t)]^2$$

C1	**0**
C2	**6**

P(C1) = 0/6 = 0 **P(C2) = 6/6 = 1**

Gini = 1 – P(C1)² – P(C2)² = 1 – 0 – 1 = 0

C1	**1**
C2	**5**

P(C1) = 1/6 **P(C2) = 5/6**

Gini = 1 – (1/6)² – (5/6)² = 0.278

C1	**2**
C2	**4**

P(C1) = 2/6 **P(C2) = 4/6**

Gini = 1 – (2/6)² – (4/6)² = 0.444

Gini for when node p is split into k partitions: $GINI_{split} = \sum_{i=1}^{k} \frac{n_i}{n} GINI(i)$ where n_i = number of records at child i, n = number of records at parent node p.

Now we have:

	Parent	N1	N2
C1	7	5	2
C2	5	1	4

We calculate (from the GINI values we got earlier) that the weighted GINI of N1 and N2 (where the number of records for each child is 6 and the records of the parent node for both is 12) is $6/12 * 0.278 + 6/12 * 0.444 = 0.361$. The parent Gini (where C1 = 7 and C2 = 5) is 0.486 which gives us a GAIN of parentGINI - weightedChildGINI = 0.486 - 0.361 = 0.125.

Multi-way split GINI example:

	Family	Sports	Luxury
C1	1	8	1
C2	3	0	7

The Gini is calculated for Family $(2(1/4)(1 - (1/4)) = 3/8)$, Sports $(2(1)(1-1) = 0)$, and Luxury $(2(1/7)(1 - (1/7)) = 7/32)$ and from that a weighted GINI is found which is $(4/20) * (3/8) + 0 + (8/20) * (7/32) = 0.1625$.

We want smallest GINI index

3.1.3 Measure of Impurity: Entropy

Example: C1 = 1, C2 = 5 So Entropy = $E = -(1/6)log_2(1/6) - (5/6)log_2(5/6) = 0.65$. C1 = 2, C2 = 4 so Entropy = $E = -(2/6)log_2(2/6) - (4/6)log_2(4/6) = 0.92$

MAXIMIZE GAIN!!!!
Node impurity measures tend to prefer splits that result in large number of partitions, each being small but pure.

$$GAIN_{split} = Entropy(p) - (\sum_{i=1}^{k} \frac{n_i}{n} Entropy(i)$$

$$GainRATIO_{split} = \frac{GAIN_{split}}{SplitINFO} = \frac{GAIN_{split}}{-\sum_{i=1}^{k} \frac{n_i}{n} log_2 \frac{n_i}{n}}$$

Error of single node computations:
If C1 = 1 and C2 = 5 then $Error = 1 - max(1/6, 5/6) = 1 - 5/6 = 1/6$

Decision Tree based Classification

- Advantages
  - Inexpensive to construct
  - Extremely fast at classifying unknown records
  - Easy to interpret for small-sized trees
  - Robust to noise (especially when methods to avoid overfitting are employed)
  - Can easily handle redundant or irrelevant attributes (unless the attributes are interacting)
- Disadvantages
  - Space of possible decision trees is exponentially large, greedy approaches are often unable to find the best tree
  - Does not take into account interactions between attributes
  - Each decision boundary involves only a single attribute

3.2 Overfitting

Classification Errors

- Training errors (apparent errors) - errors committed on the training set
- Test errors - errors committed on the test set

- Generalization errors - expected error of a model over random selection of records from same distribution

Too many parameters can cause overfitting.

Underfitting: when model is too simple both training and test errors are large.

Overfitting: when model is too complex, training error is small but test error is large. Increasing the size of training data reduces the difference between trainind and testing errors at a given size of model. Reasons for overfitting is limited training size and high model complexity (multiple comparison procedure). Results in decision trees that are more complex than necessary. Training error does not provide a good estimate of how well the tree will perform on previously unseen records. Need ways for estimating generalization errors.

Multiple Comparison Procedure - get 50 analysts, each analyst makes 10 random guesses, choose the analyst that makes the most number of correct predictions.

Model Selection - performed during model building, purpose is to ensure that model is not overly complex (to avoid overfitting).

Need to estimate generalization error

- Using Validation set
  - divide traing data into two sets: Training (used for model building) and validation set (used for estimating generalization error, not the same as test set)
  - Drawback: less data available for training
- Incorporating Model Complexity
  - Occam's Razor
  - Given two models of similar generalization errors, one should prefer the simpler model over the more complex model
  - A complex model has a greater chance of being fitted accidentally
  - Therefore one should include model complexity when evaluating a model
- Estimating Statistical Bounds
  - Pessimistic Error Estimate of decision tree T with k leaf nodes $err_{gen}(T) = err(T) + \omega x \frac{k}{N_{train}}$
  - err(t): error rate on all training records

  - ω: trade-off hyper-parameter (similar to α) - relative cost of adding a leaf node
  - k: number of leaf nodes
  - N_{train}: total number of training records
- Resubstitution Estimate
  - Using training error as an optimistic estimate of generalization error
  - Referred to as optimistic error estimate

Minimum Description Length (MDL)

- Cost(Model, Data) = Cost(Data—Model) + α x Cost(Model) - cost is the number of bits needed for encoding, search for the least costly model
- Cost(Data—Model) encodes the misclassification errors
- Cost(Model) uses node encoding (number of children) plus splitting condition encoding

Pre-Pruning (Early Stopping Rule

- Stop the algorithm before it becomes a fully grown tree
- Typical stopping conditions for a node: stop if all instances belong to the samee class or if all the attribute values are the same
- More restrictive conditions:
  - Stop if number of instances is less than some user-specified thershold
  - Stop if class distribution of instances are independent of the available features
  - Stop if expanding the current node does not improve impurity measures
  - Stop if estimated generlization error falls below certain threshold

Post-pruning

- Grow decision tree to its entirety
- Subtree replacement
  - Trim the nodes of the decision tree in a bottom-up fashion
  - If generalization error improves after trimming, replace sub-tree by a leaf node
  - Class label of leaf node is determined from majority class of instances in the sub-tree
- Subtree raising - replace subtree with most frequently used branch

Variations on Cross-validation

- Repeated cross-validation
  - Perform cross-validation a number of times
  - Gives an estimate of the variance of the generalization error
- Stratified cross-validation
  - Guarantee the same percentage of class labels in training and test
  - Important when classes are imbalanced and the sample is small
- Use nested cross-validation approach for model selection and evaluation

3.3 Model Evaluation

Purpose: to estimate performance of classifier on previously unseen data (test set)
Holdout: Reserve k% for training and (100-k)% for testing, random subsampling:repeated holdout
Cross Validation: partition data into k disjoint subsets, k-fold: train on k-1 partitions, test on remaining one, leave-one-out: k=n

Metrics for performance evaluation: focus on the predictive capability of a model rather than how fast it takes to classify or build models, Confusion matrix.
Accuracy = $\frac{a+d}{a+b+c+d}$ - a and d are diagonal from one another in confusion matrix.

Methods for performance evaluation: Performance of a model may depend on other factors besides the learning algorithm like class distribution, cost of misclassification, size of training and test sets.

Methods of Estimation:

- Holdout - reserver 2/3 for training and 1/3 for testing
- Random subsampling - repeated holdout
- Cross validation - partition data into k disjoint subsets
- Stratified sampling - oversampling vs undersampling
- Bootstrap - sampling with replacement

Methods for model comparison: use confidence interval for accuracy and be wary of which data model is better based on accuracy (sample size matters).

3.4 Nearest Neighbor

Basic idea: if it walks like a duck, quacks like a duck, then it's proabably a duck.

Requires three things:

- The set of labeled records
- Distance metric to compute distance between records
- The value of k, the number of nearest neighbors to retrieve

To classify an unkown record:

- Compute distance to other training records
- Identify k nearest neighbors
- Use class labels of nearest neighbors to determine the class label of unknown record (like by taking a majority vote)

Use Euclidean distance to compute proximity between two points, cosine if it is documents. Weight factor $w = \frac{1}{d^2}$.
Choosing value of k: too small then sensitive to noise points, too big and neighborhood may include points from other classes.

Preprocessing required: attributes may have to be scaled to prevent distance measures from being dominated by one of the attributes, attributes are often standardized to have 0 means and a standard deviation of 1.

Nearest neighbor classifiers are LOCAL classifiers, they can produce decision boundaries of arbitrary shapes.

How to handle missing values in training and test sets: proximity computations normally require the presence of all attributes, some approaches use the subset of attributes present in two instances -> this may not produce good results since it effectively uses different proximity measures for each pair of instances thus proximities are not compaarable.

Handeling irrelevant and redundant attributes

- Irrelevant attributes add noise to the proximity measure
- Redundant attributes bias the proximity measure towards certain attributes
- Can use variable selection or dimensionality reduction to address irrelevant and redundant attributes

Improving KNN Efficiency

- Avoid having to compute distance to all objects in the training set - Multi-dimensional access methods (k-d trees), Fast approximate similarity search, Locality Sensitive Hashing (LSH)
- Condensing - determine a smaller set of objects that give the same performance
- Editing - remove objects to improve efficiency

3.5 Bayesian Classifiers

Conditional probability: $P(Y|X) = \frac{P(X,Y)}{P(X)}$ and $P(X|Y) = \frac{P(X,Y)}{P(Y)}$
Bayes theorem: $P(Y|X) = \frac{P(X,Y)P(Y)}{P(X)}$

Replace continuous value with binary value.

Probability density estimation

- Assume attribute follows a normal distribution
- Use data to estimate parameters of distribution (eg. mean and sd)
- Once probability distribution is known, use it to estimate the conditional probability $P(X_i|Y)$

Normal distribution: $P(X_i|Y_j) = \frac{1}{\sqrt{2\pi\sigma_{ij}^2}} e^{-\frac{(X_i-\mu_{ij})^2}{2\sigma_{ij}^2}}$ - μ = mean and σ = sd and σ^2 = variance.

If one of the conditional probabilites is zero then the entire expression becomes zero.

Naive Bayes summary

- Robust to isolated noise points
- Handle missing values by ignoring the instance during probability estimate calculations
- Robust to irrelevant attributes
- Redundant and correlated attributes will violate class conditional assumption - use other techniques such as Bayesian Belief Networks (BNN)

Bayesian Belief Networks

- Provides graphical representation of probabilistic relationships among a set of random variables

- A direct acyclic graph (dag) - node corresponds to a variable, arc corresponds to dependence relationship between a pair of variables
- A probability table associating each node to its immediate parent

Conditional Independence: A node in a Bayesian network is conditionally independent of all of its nondescendants if its parents are known.

3.6 Support Vector Machines

Linear Machines

- Learn to separate two classes via a linear decision function
- Defined by a weight vector w and a bias b $w^T x + b = 0$
- SVMs construct an optimal linear decision function in a high-dimensional feature space nonlinearly related to the input space
- Assume classes are linearly separable
- Maximize margin (ϱ) by minimizing the norm of the weight vector

Nonseparable Patterns

- In interesting problems, no linear function can perfectly classify all examples
- Define a margin to be soft if $\exists i : y_i(w^T x_i + b) < 1$
- To allow soft margins we introduce slack variables into the constraints yielding $y_i(w^T x_i + b) \geq 1 - \xi_i$
- Error occurs iff $\xi_i \geq 1$
- Number of training errors bounded by $\sum_{i=1}^{|} \xi_i$
- Thus we change the optimization problem to be that of minimizing $\frac{1}{2} \parallel w \parallel^2 + C \sum_{i=1}^{|} \xi_i$

Kernel Machines

- Problem with using ϕ directly - very high (possibly infinite dimensionality and therefore computational complexity
- what if there were a Kernel function $K(x_i, x_j)$ such that $K(x_i, x_j) = \phi(x_i) \cdot \phi(x_j)$
- Commonly used Kernels: Polynomial Kernels, Radial Basis Function Kernels, Hyperbolic Tangent Kernels

One-Class SVM/ SVM Multi-class Classification

- SVM's only have binary output
- To get an M-class classifier train M binary classifiers, each separating one class from the rest
- Real-valued output of an SV¡ (distance to decision boundary) can be interpreted as confidence value
- Choose the class for wich teh confidence is the largest $argmax w^{j^T} x + b^j$, j=1...M

Basic ideas: Find linear decision boundary that maximizes the margin, use slack variables to deal with noise, use kernel function (mapping to higher dimensional space) to get non-linear decision boundary.

3.7 Ensemble Methods

Construct a set of classifiers from the training data, predict class label of previously unseen records by aggregating predictions made by multiple classifiers.

1. Create multiple data sets
2. Build multiple classifiers
3. Combine classifiers

Bagging

- Sampling with replacement
- Build classifier on each bootstrap sample
- Each sample has probability $(1 - 1/n)^n$ of being selected

Boosting

- An iterative procedure to adaptively change distribution of training data by focusing more on previously misclassified records
- Initially, all N records are assigned equal weights
- Unlike bagging, weights may change at the end of boosting round
- Records that are wrongly classified will have their weights increased
- Records that are classified correctly will have their weights decreased
- Example: AdaBoost

3.8 Regression

Solving a regression problem is finding a conditional expectation or average value of y. Named in the sense of regression to the mean.

- Data: pairs of (x_i, y_i) with $x_i \in \mathbb{R}$ for $1 \leq i \leq n$
- Assumption: there is an underlying (unknown) function f(x) = y
- Gaol: find a function $h : \mathbb{R}^D \to \mathbb{R}$, such that $(\forall x)h(x) \approx f(x)$
- Usually hypothesis h is a function with parameters w
- Then teh objective becomes to find parameter values w that minimize the error of h on the given data $arg_w minE(w)$
- A popular error function is the Mean Squared Error (MSE): $E(w) = \frac{q}{2n} \sum_i (h_w(x_i) - y_i)^2$

(Multiple) Linear Regression

- $h_w(x)$ has only linear dependence on w
- Does not matter how h depends on x
- $h_w(x) = w^T x$

Non-Linear Regression - use a numeric method eg. Gradient Descent

Gradient Descent (GD) - Iterative method, where w gets gradually changed towards decreasing E(w).

Parameter α is called **learning rate**.

α too small - slow convergence (many iterations necessary), α too large - possible divergence.

Initial weight (w) determines which minimum is reached.

Stochastic Gradient Descent - consider only one training point at a time, doesn't guarantee convergence, oscillates around minimum withou settling down.

Regression Trees - Decision Trees with Numeric Output - Issues

- Impurity Measure: use variance reduction in the output attribute instead of information gain
- Leaf Nodes: reveal values as leaf nodes instead of class labels

- Performance Measure: mean swuared error instead accuracy/precision/recall

Using kNN for Regression - use (weighted) average of output values of the k neighbors instead of majority vote.

Logistic Regression is a classification method.

3.9 Neural Networks

History

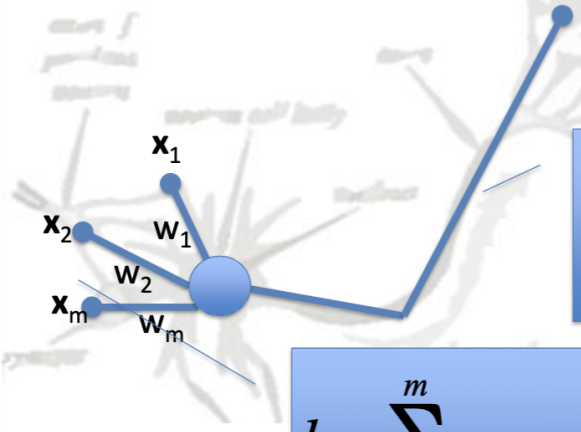
- Neuron as a computational model - McCullon and Pitts (1943)
- Perceptrons and learning algorithms - Rosenblatt (1958)
- Limitations and Perceptrons - Minsky and Papert (1969)
- "Renaissance" of ANN (1980) - Multi-layer networks and Back-Propagation

Biological Inspiration

- Human brain
- Over 10 billion neurons, each on average connected to thousands of other neurons
- Massive parallelism
- Fire several hundred times per second

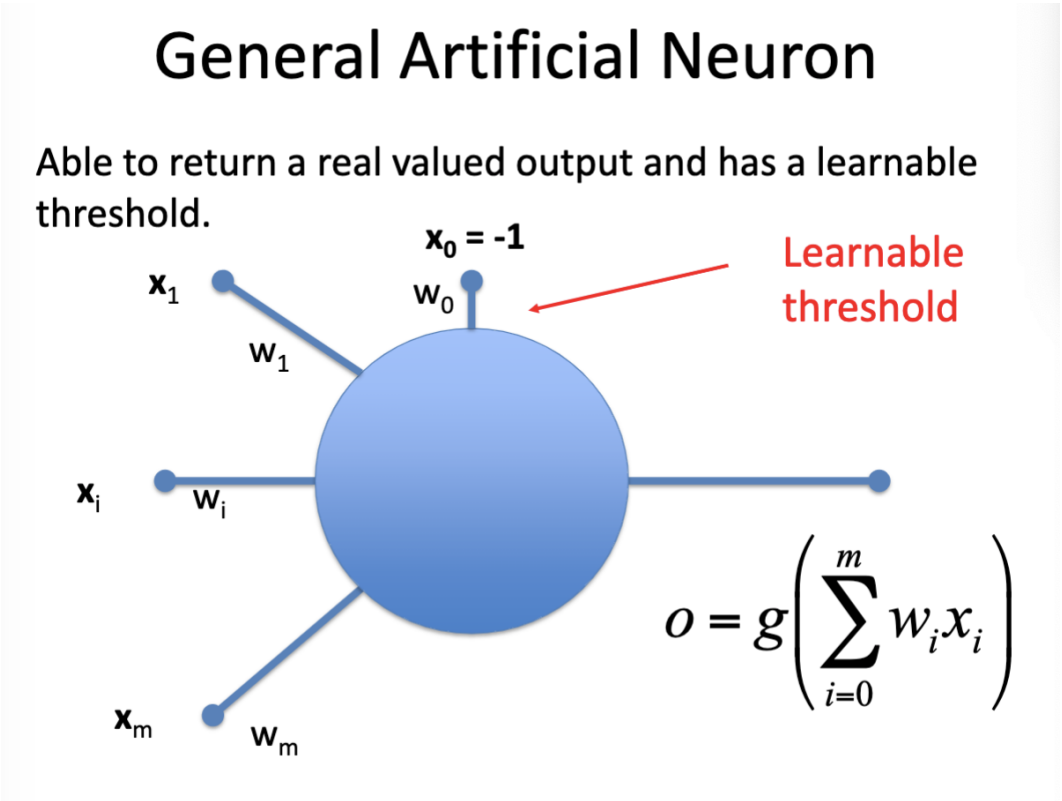
Binary Artificial Neuron

McCulloch and Pitts (1943) binary output neuron proposed as a computational model.



$$o = \begin{bmatrix} 1 & h > \theta \\ 0 & h \leq \theta \end{bmatrix}$$

$$h = \sum_{i=1}^{m} w_i x_i = \vec{w} \cdot \vec{x}$$



Perceptron

Supervised learning - learning from a labeled training data (Input features: $x_1, ..., x_m$; Output feature (label): o (either 0 or 1), possibly more than 1)

Training a Perceptron:

- We want to "train" the perceptron to give the right output given a particular input
- Given a set of labeleed samples $< x_1, ...x_m; t >$
  - Modify the weights such that given inputs and perceptron will output the value
- Should hold for (nearly) all the samples

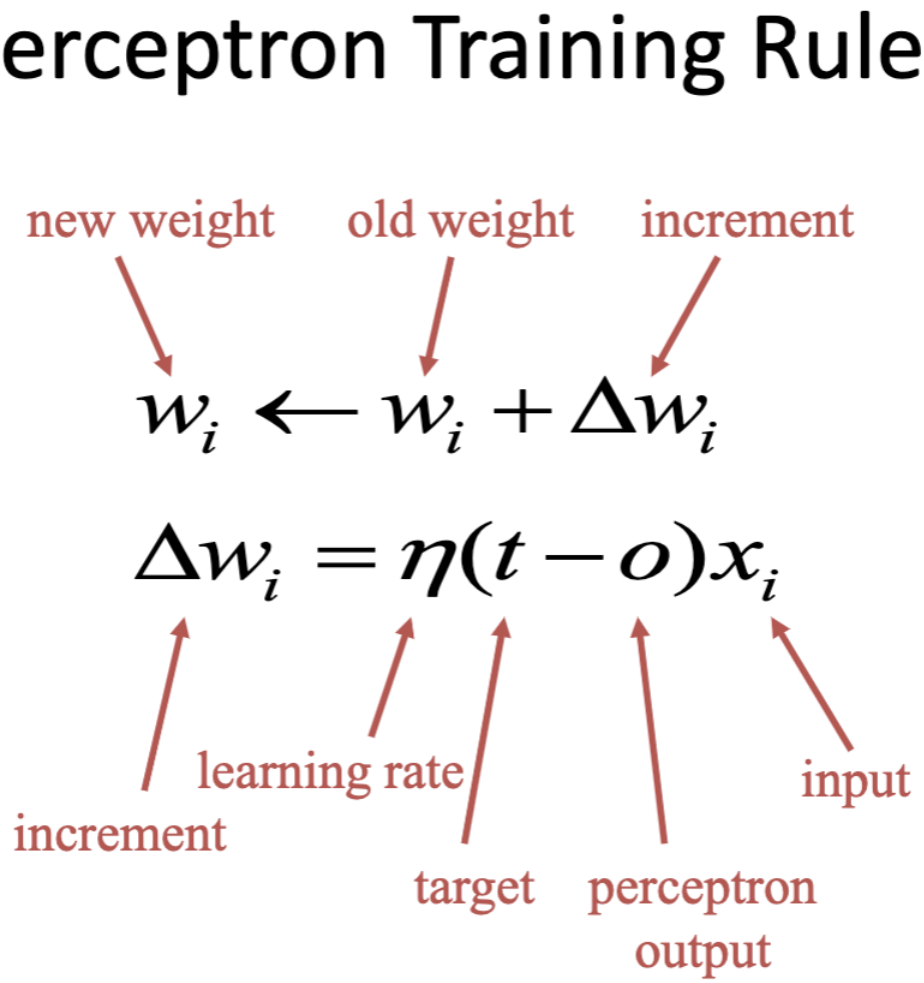
Training Algorithms

- For both algorithms - Initialize all weights ($w_i <-$ small random number), iterations are also called epochs

- Iterative - for T iterations:
  - for each input record <$x_1, \ldots, x_m$; t>:
    - Compute output $o = g\left(\sum_{i=0}^{m} w_i x_i\right)$
    - Update all *i* weights:
      - $w_i \leftarrow w_i + \eta\,(t - o)\,x_i$ // or: $\vec{w} \leftarrow \vec{w} + \eta(t-o)\vec{x}$

Learning rate

- Batch Learning - for T iterations:
  - Set $\Delta\vec{w} \leftarrow \vec{0}$
  - for each input record <$x_1, \ldots, x_m$; t>:
    - Compute output $o = g\left(\sum_{i=0}^{m} w_i x_i\right)$
    - Update weight difference: $\Delta\vec{w} \leftarrow \Delta\vec{w} + \eta(t-o)\vec{x}$
  - $\vec{w} \leftarrow \vec{w} + \Delta\vec{w}$



OR learning - calculate teh same way for all inputs to finish first iteration and then continue for T iterations or until converges or error is small enough.

Can have many units in a single layer perceptron network.

Limit - can only learn linearly separable functions (for instance cannot learn XOR)

Implications

- Serious limitations with Perceptron

- Paper by Minsky and Papert (1969)

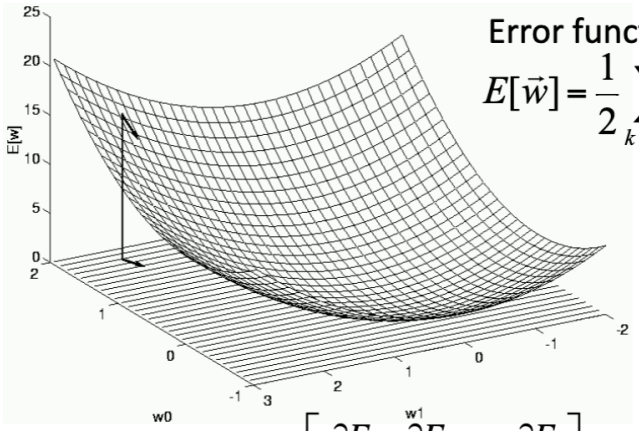
- Multi-Layer Networks
  - Later shown that can learn arbitrary functions wheeen used with non-linear activation funcitons
  - Problem: how to learn the weights in multi-layeer networks
  - Solution: Back-propagation

- Artificial Neural Networks
  - Multi-layer and Back-propagation Learning

What if data is not linearly seeparable?

- Build multiple layer networks (hidden layer)
- Use a non-linear activation function (e.g. sigmoid or rectified linear) instead of a step function
- Note: need to use a non-linear squashing function because multi-layer linear is still linear

Idea for Training: Gradient Descent



Error function:

$$E[\vec{w}] = \frac{1}{2} \sum_{k \in O} (t_k - o_k)^2$$

Gradient:

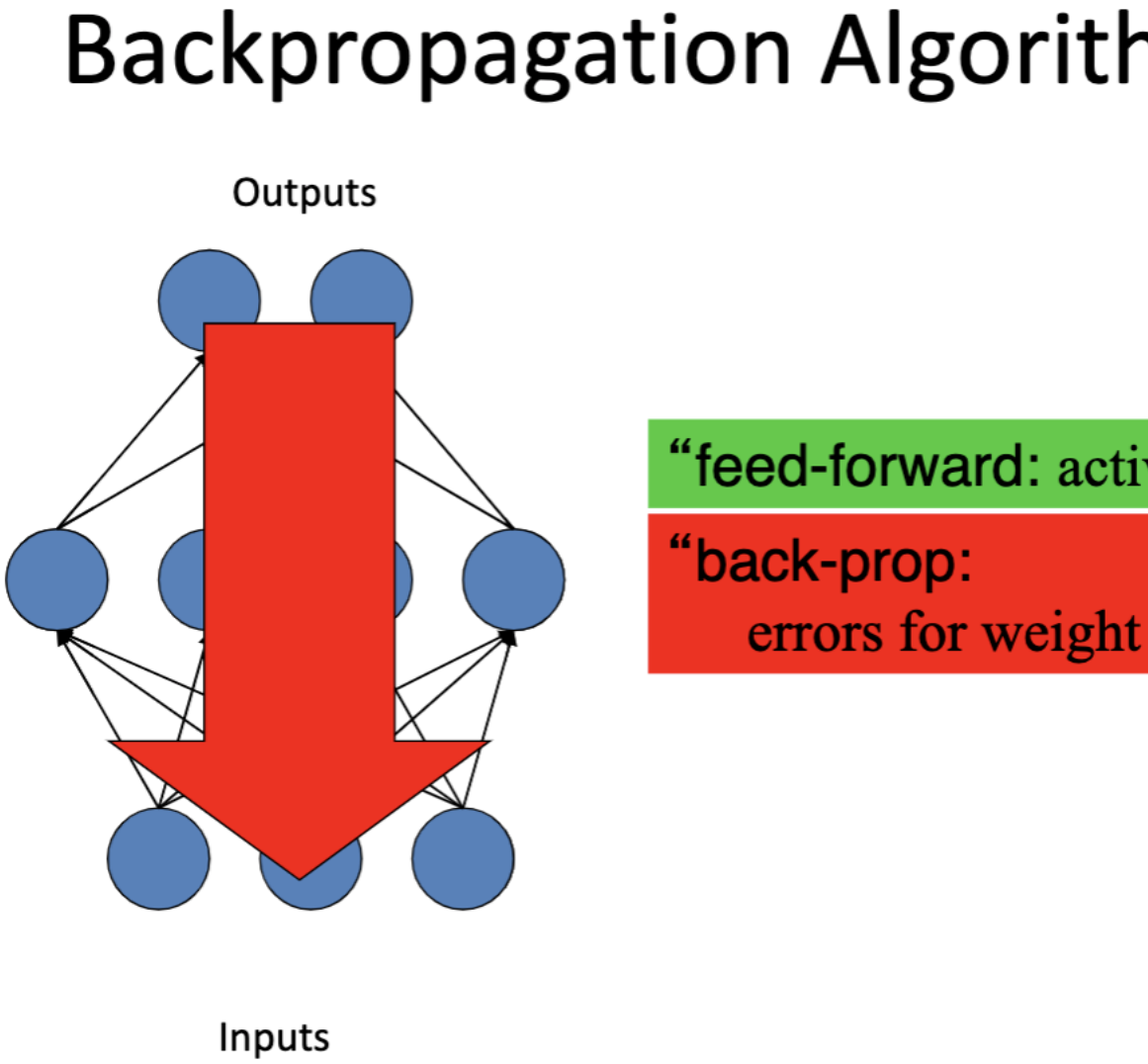
$$\nabla E[\vec{w}] = \left[\frac{\partial E}{\partial w_0}, \frac{\partial E}{\partial w_1}, \ldots, \frac{\partial E}{\partial w_n} \right]$$

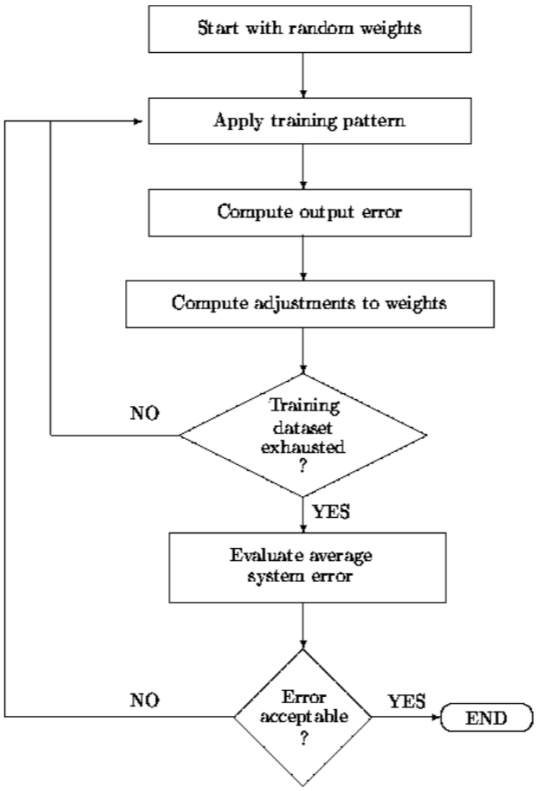
Training rule:

$$\Delta \vec{w} = -\eta \, \nabla E[\vec{w}]$$

$$\Delta w_i = -\eta \frac{\partial E}{\partial w_i}$$

Back-propagation Algorithm - generalization of updates of weights to multiple layers and multiple output units

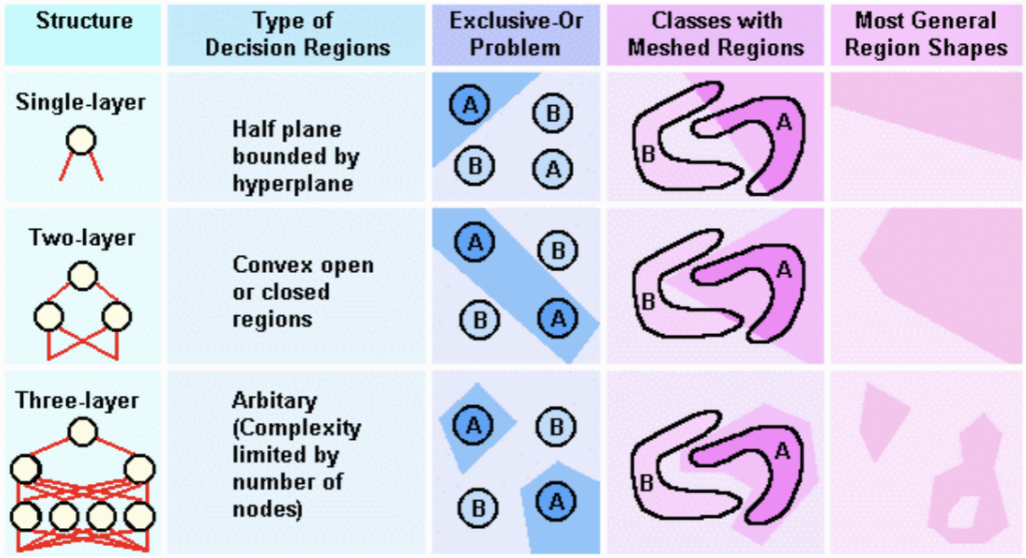




Convergence - may get stuck in local minima, weights may diverge but works well in npractice.

Representation power: 2 layer networks: any continuous function; 3 layer networks: any function

Pattern Separation and NN architecture



Practical Issues Using ANN

- Architecture
  - Input and Output Encoding
    - "Garbage in, garbage out"
    - Input features
      - Should be representative, ideally "high level" - some pre-processing might be necessary, may be helpful to "normalize"
      - Sometimes choice restricted by the data we have - may be out of our control (except normalization)
    - Output features
      - Tell what we want to know
      - Encoding
    - How to deal with nominal attributes (including the class labels)? Replace each nominal attribute $A \in \{a_1, ..., a_n\}$ with n binary attributes $A_i = \{1, if A = a_i, 0, if A \neq a_i\}$
  - Hidden layers
    - Hidden layers
      - How many (1, 2, more?) and how maany neurons in each?

        * Larger network, the fitting ability may increase
            · However, the longer the training takes,
            · Too many "degrees of freedom": ability to generalize may decrease, risk of over-fitting
            · Rule-of-thumb: total number of neurons o more than 1/10 the number of training records
    - Initialize weights
        * Initialize weights to "small" random numbers both positive and negative
        * Rule of thumb: Assuming n inputs into a neuron, initial weights in range: $\frac{-1}{\sqrt{n}} \leq w \leq \frac{1}{\sqrt{n}}$
        * May bee a good idea to train several times with different initial weights each time
            · Minimizes the risk of getting stuck in a local minimum
            · Choose the best network tried

- Training
    - Learning Speed
        * Step size parameter
            · Adjustable
            · Decrease step size as training proceeds (later iterations)
        * Use momentum
            · Idea: if a weight is succinctly changing in the same direction, then try larger steps for that weight
            · Analogy: ball rolling down a hill, building up speed
            · The speed of learning is governed by the learning rate
            · — If the rate is low, convergence is low
            · — If rate is too high, error oxcillates without reaching minimum
            · — $\Delta w_{ij}^t = \eta \delta_i y_j + \alpha \Delta w_{ij}^{t-1}, (0 \leq \alpha < 1)$ - new last term: add a fraction of last update to current
            · Momentum tend to smooth small weight error fluctuations:
            · — Accelerates the descent in steady downhill directions
            · — Stabilizes oscillationss
    - When to stop learning?
        * Do fixed number of iterations
            · May stop pre-maturely before converging
            · Take too many iterations
        * Do until error reaches an acceptable level (e.g., % of correctly classified samples, or difference between target and network output is small)

        - May never reach the preset "acceptable" level
        - Or we could potentially do better, without risking over-fitting
    - Use validation set to test network after each epoch, stop when error stops to decrease

- Testing
  - Train, Validate, Test

Enhancements of ANN's

- Much research been done in ANN over the decades
- Many more enhancements than presented here, e.g.
  - self-organizing networks
  - recurrecnt networks
  - deep neural networks (Deep Learning)
  - convolutional networks

Summary

- Pros:
  - Effective classifiers - can learn arbitrary functions
  - Relatively easy to use for many applications
- Cons:
  - Non-declarative for humans to learn from - did the network "discover" new rules we can learn?
  - Expensive to train - typically needs a lot of data and many iterations

3.10 Back Propagation

Neural network architecture

- We have defined a feed forward neural nnetwork with L layers. The output of each layer forms the input to the next layer. The input to the overall network is still $x_i = z_i^0$ while the output of the overall network is defined as $y_k(\vec{x}, \vec{w}) = z_k^{(L)}$. We also like to think of the weights in the network as either a vector $\vec{w}$ or a swquence of matrices $\vec{W}^{(L)}$
- Calculating the output from the input means that you successively apply: $a_j^{(l)} = \sum_{i=0}^{M_{l-1}} w_{ji} z_i^{(l-1)}$ and $z_j^{(l)} = h(a_j^{(l)})$
- We will only consider feed-forward neural networks. This meaans the network is acyclical and information i nlater layers cannot affect earlier layers.

- The generalization of this architecture is called *recurrent neural network* and is covered in a later talk in the course
- There are also weight space symmetries
- Reordering neurons in one layer does not affect the input-output relationship. This means there are many equivalent points in the weeight space

Neural network training

- In supervised learning we rely on training data to determine the values of the weights in the neural network. We assume that we have training data of the form $\{x_n, t_n\}_{n=1}^N$ where x_n is an input vector and t_n is the desired output or *target value* for that input
- Assume the network is performing regression on a single value and there is only a single neuron in the output layer and its activation function is linear $h_{M_L}^{(L)}(a_{M_L}^{(L)}) = h^{(L)}(a^{(L)}) = 1$ as $M_L = 1$. This is a natural choice for a single value regression
- Multi-valued regression and classification works similarly, but we need to choose different error functions and output activation functions
- The input and the target values are now constant so the output only depends on the weights. For a single value regression a natural choice of an error function is the sum-of-squares $E(\vec{w}) = \frac{1}{2}\sum_{n=1}^N (y(\vec{x}_n, \vec{w}) - t_n)^2$. The aim is to mimimize the error by changing the weights
- It is impossible to solve $\nabla E(\vec{w}) = 0$ directly so we need a numerical procedure - gradient decent: $\vec{w}^{\tau+1} = \vec{w}^{\tau} - \eta \nabla E(\vec{w}^{\tau})$ where η is the *learning rate* and τ is the *learning iteration*
- The error function $E(\vec{w})$ is defined with respect to the entire training data: This is called *batch learning*. The error function can be decomposed into terms specific to individual data points $\vec{x}_n$ - $E(\vec{w}) = \sum_{n=1}^N E_n(\vec{w})$ - leading to *online learning* either using *sequential* or *stochastic* gradient decent using equation from above

The backpropagation algorithm

The backpropagation algorithm

The backpropagation algorithm is a method to compute the gradient of the error function $E(\mathbf{w})$ or more specifically $E_n(\mathbf{w})$

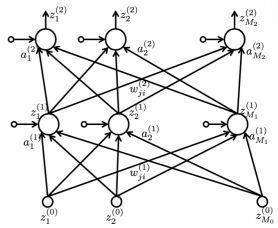
$$\nabla E_n(\mathbf{w}) = \begin{bmatrix} \frac{\partial E_n}{\partial w_1} \\ \frac{\partial E_n}{\partial w_2} \\ \vdots \\ \frac{\partial E_n}{\partial w_M} \end{bmatrix} \longleftarrow \frac{\partial E_n}{\partial w_{ji}^{(l)}}$$

The backpropagation algorithm

We have an initial value of $\mathbf{w}^{(0)}$, we have performed the forward pass for the pattern $\mathbf{x}_n$ and stored all intermediate values:

$$a_j^{(l)} = \sum_{i=0}^{M_{l-1}} w_{ji}^{(l)} z_i^{(l-1)}$$

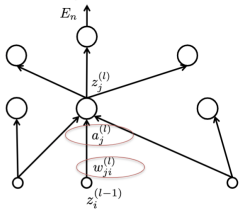
$$z_j^{(l)} = h(a_j^{(l)})$$



The backpropagation algorithm

We compute the derivative of E_n with respect to $w_{ji}^{(l)}$.
E_n is dependent on $w_{ji}^{(l)}$ only through $a_j^{(l)}$.
We use the *chain rule*:

$$\frac{\partial E_n}{\partial w_{ji}^{(l)}} = \frac{\partial E_n}{\partial a_j^{(l)}} \frac{\partial a_j^{(l)}}{\partial w_{ji}^{(l)}}$$



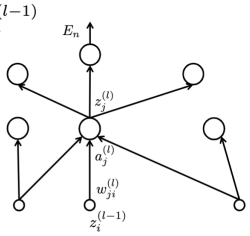
The backpropagation algorithm

Which means we can compute the derivative using

$$\frac{\partial E_n}{\partial w_{ji}^{(l)}} = \frac{\partial E_n}{\partial a_j^{(l)}} \frac{\partial a_j^{(l)}}{\partial w_{ji}^{(l)}} = \delta_j^{(l)} z_i^{(l-1)}$$

We have already computed all $z_i^{(l-1)}$ in the forward pass.

All we need is $\delta_j^{(l)}$

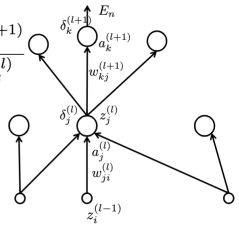


The backpropagation algorithm

The errors $\delta_j^{(l)}$'s can be computed from the $\delta_k^{(l+1)}$'s in the succeeding layer:

$$\delta_j^{(l)} = \frac{\partial E_n}{\partial a_j^{(l)}} = \sum_{k=1}^{M_{l+1}} \frac{\partial E_n}{\partial a_k^{(l+1)}} \frac{\partial a_k^{(l+1)}}{\partial a_j^{(l)}}$$

$$= \sum_{k=1}^{M_{l+1}} \delta_k^{(l+1)} \frac{\partial a_k^{(l+1)}}{\partial a_j^{(l)}}$$



The backpropagation algorithm

We define the error related to neuron *j* as:

$$\delta_j^{(l)} \equiv \frac{\partial E_n}{\partial a_j^{(l)}}$$

and we also notice that:

$$\frac{\partial a_j^{(l)}}{\partial w_{ji}^{(l)}} = \frac{\partial}{\partial w_{ji}^{(l)}} \left(\sum_{\iota=0}^{M_{l-1}} w_{j\iota}^{(l)} z_\iota^{(l-1)} \right) = z_i^{(l-1)}$$

The backpropagation algorithm

For the last layer we have

$$E_n(\mathbf{w}) = \frac{1}{2}(y(\mathbf{x}_n, \mathbf{w}) - t_n)^2$$

and a linear activation function so:

$$\delta_1^{(L)} = \frac{\partial E_n}{\partial a_1^{(L)}} = y(\mathbf{x}_n, \mathbf{w}) - t_n$$

The backpropagation algorithm

We can compute $a_k^{(l+1)}$ from $a_j^{(l)}$ by noticing that:

$$a_k^{(l+1)} = \sum_{\iota=1}^{M_{(l)}} w_{k\iota}^{(l+1)} z_\iota^{(l)} = \sum_{\iota=1}^{M_{(l)}} w_{k\iota}^{(l+1)} h(a_\iota^{(l)})$$

so

$$\frac{\partial a_k^{(l+1)}}{\partial a_j^{(l)}} = w_{kj}^{(l+1)} h'(a_j^{(l)})$$

The backpropagation algorithm

This means that we can compute the errors using

$$\delta_j^{(l)} = \sum_{k=1}^{M_{l+1}} \delta_k^{(l+1)} w_{kj}^{(l+1)} h'(a_j^{(l)})$$

Notice that the sum is taken over the first index indicating a flow of information backwards through the network.

We can now compute the $\delta's$ for all neurons in the network

The backpropagation algorithm

The backpropagation algorithm computes the derivative with respect to one pattern:

$$\nabla E_n(\mathbf{w}) = \begin{bmatrix} \frac{\partial E_n}{\partial w_1} \\ \frac{\partial E_n}{\partial w_2} \\ \vdots \\ \frac{\partial E_n}{\partial w_M} \end{bmatrix}$$

This can be used directly in stochastic gradient decent or used to accumulate

$$\nabla E(\mathbf{w}) = \sum_{n=1}^{N} \nabla E_n(\mathbf{w})$$

for batch mode training using for example:

$$\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} - \eta \nabla E(\mathbf{w}^{(\tau)})$$

The backpropagation algorithm

The following is the backpropagation algorithm:

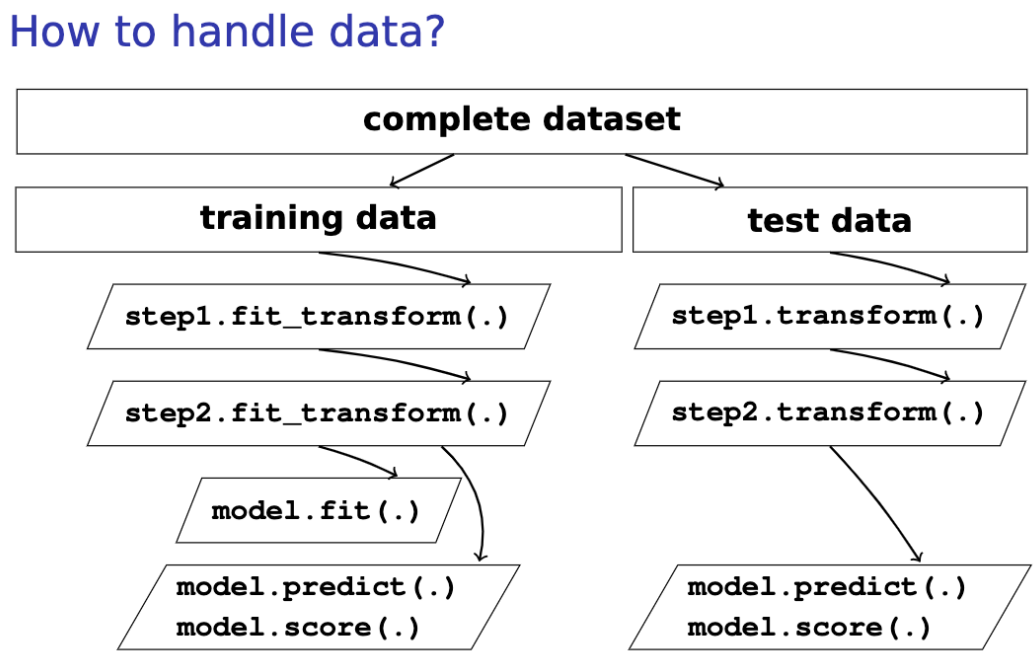
1. Apply pattern $\mathbf{x}_n$ to the net and obtain all *a's* and *z's*
2. Compute the output deltas $\delta_j^{(L)}$
3. Use the backpropagation formula to compute all deltas:
$$\delta_j^{(l)} = h'(a_j^{(l)}) \sum_{k=1}^{M_{l+1}} \delta_k^{(l+1)} w_{kj}^{(l+1)}$$
4. Use this formula to obtain the derivative:
$$\frac{\partial E_n}{\partial w_{ji}^{(l)}} = \delta_j^{(l)} z_i^{(l-1)}$$

The backpropagation algorithm

Relevant issues not covered:

- Computing the Hessian for the Newton method
- Regularization
- Momentum
- The vanishing gradient problem
- Specifics of the activation function

3.11 Data Workflow



4 Unsupervised Learning

Unsupervised Learning: learning patterns in the input without feedback

4.1 Association Rule Mining

Given a set of transactions, find rules that will predict the occurrence of an item based on the occurrences of other items in the transaction.

Definitions - Frequent Itemset

- Itemset - a collection of one or more items (eg. {Milk, Bread, Daiper}; k-itemset, an itemset that contains k items
- Support count (σ) - frequency of occurrence of an itemset (eg. σ({Milk, Bread, Diaper}) = 2
- Support - fraction of transactions that contain an itemset (eg. s({Milk, Bread, Diaper}) = 2/5

- Frequent Itemset - an itemset whose support is greater than or equal to a minsup threshold

Definitions - Association Rule

- Association Rule - an implication expression of the form X $\rightarrow$ Y where X and Y are itemsets (eg. {Milk, Diaper} $\rightarrow$ {Beer}
- Rule Evaluation Metrics
  - Support (s) - Fraction of transactions that contain both X and Y - $s = \frac{\sigma(Milk, Diaper, Beer)}{|T|} = \frac{2}{5} = 0.4$
  - Confidence (c) - Measures how often items in Y appear in transactions that contain X - $c = \frac{\sigma(Milk, Diaper, Beer)}{\sigma(Milk, Diaper)} = \frac{2}{3} = 0.67$

Given a set of transactions T, the goal of association rule mining is to find all rules having support $\geq$ *minsup* threshold and confidence $\geq$ *minconf* threshold.

Brute force approach:

- List all possible association rules
- Compute the support and confidence for each rule
- Prune rules that fail the *minsup* and *minconf* thresholds
- **Computationally prohibitive**

Mining Association Rules Rules originating from the same itemset have identical support but can have different confidence. Thus we may decouple the support and confidence requirements.

Two step approach:

1. Frequent Itemset Generation - generate all itemsets whose support $\geq$ minsup
2. Rule Generation - generate high confidence rules from each frequent itemset, where each rule is a binary partitioning of a frequent itemset

Frequent itemset generation is still computationally expensive.

Frequent Itemset Generation Given d items, there are 2^d possible candidate itemsets.

Brute-force approach:

- Each itemset in the lattice is a candidate frequent itemset
- Count the support of each candidate by scanning the database

- Match each transaction against every candidate
- Complexity $\sim$ O(NMw) => Expensive since $M = 2^d$!!!

Computational Complexity
Given d unique items, total number of itemsets is 2^d and total number of possible association rules is

$$R = \sum_{k=1}^{d-1}[\binom{d}{k} x \sum_{j=1}^{d-k} \binom{d-k}{j}] = 3^d - 2^{d+1} + 1$$

Frequent Itemset Generation Strategies

- Reduce the number of candidates (M) - use pruning techniques to reduce M in $M = 2^d$
- Reduce the number of transactions (N) - reduce size of N as the size of the itemset increases, used by DHP and vertical-based mining algorithms
- Reduce teh number of comparisons (NM) - use efficient dataa structures to store the candidates or transactions, no need to match every candidate against every transaction

Reducing Number of candidates
Apriori principle - if an itemset is frequent then all of its subsets must also be frequent. The principle holds due to this property of the support measure: $\forall X, Y : (X \subseteq Y) \Rightarrow s(X) \geq s(Y)$ - support of an itemset never exceeds the support of its subsets, this is known as teh anti-monotone property of support. To reduce number of comparisons you can scan the database of transactions to determine the support of each candidate itemset and store the candidates in a hash structure (instead of matching each transaction against every candidate, match it against candidates contained in the hashed bucket). See more in the Association Rule Mining slide pdf.

Factors Affecting Complexity

- Choice of minimum support threshold
  - lowering support threshold results in more frequent itemsets
  - this may increase number of candidates and max length of frequent itemsets
- Dimensionality (number of items) of the data set
  - more space is needed to store support count of each item
  - if number of frequeent items also increases, both computation and I/O costs may also increase
- Size of database

  - since Apriori makes multiple passes, run time of algorithm may increase with number of transactions

- Average transaction width
  - transaction width increases with denser datasets
  - this may increase max length of frequent itemsets and traversals of hash tree (number of subsets in a transaction increases with its width)

Rule generation in the pdf on Association Rule Mining

Pattern Evaluation

- Association rule algorithms tend to produce too many rules - many of them are uninteresting or redundant, redundant {A,B,C} → {D} and {A,B} → {D} have same support and confidence
- Interestingness measures can be used to prune/rank the derived patterns
- In the original formulation of association rules, support and confidence are the only measures used

Computing Interestingness Measure

Computing Interestingness Measure

- Given a rule X → Y, information needed to compute rule interestingness can be obtained from a contingency table

Contingency table for X → Y

	Y	$\overline{Y}$	
X	f_{11}	f_{10}	f_{1+}
$\overline{X}$	f_{01}	f_{00}	f_{o+}
	f_{+1}	f_{+0}	|T|

f_{11}: support of X and Y
f_{10}: support of X and $\overline{Y}$
f_{01}: support of $\overline{X}$ and Y
f_{00}: support of $\overline{X}$ and $\overline{Y}$

Used to define various measures

- support, confidence, lift, Gini, J-measure, etc.

Drawback of Confidence

	Coffee	$\overline{\text{Coffee}}$	
Tea	15	5	20
$\overline{\text{Tea}}$	75	5	80
	90	10	100

Association Rule: Tea → Coffee

Confidence= P(Coffee|Tea) = 0.75

but P(Coffee) = 0.9

⇒ Although confidence is high, rule is misleading

⇒ P(Coffee|$\overline{\text{Tea}}$) = 0.9375

Statistical Independence

Statistical Independence

- Population of 1000 students
  - 600 students know how to swim (S)
  - 700 students know how to bike (B)
  - 420 students know how to swim and bike (S,B)

  - $P(S \wedge B) = 420/1000 = 0.42$
  - $P(S) \times P(B) = 0.6 \times 0.7 = 0.42$

  - $P(S \wedge B) = P(S) \times P(B)$ => Statistical independence
  - $P(S \wedge B) > P(S) \times P(B)$ => Positively correlated
  - $P(S \wedge B) < P(S) \times P(B)$ => Negatively correlated

Statistical-based Measures

- Measures that take into account statistical dependence

$$Lift = \frac{P(Y \mid X)}{P(Y)}$$

$$Interest = \frac{P(X,Y)}{P(X)P(Y)}$$

$$PS = P(X,Y) - P(X)P(Y)$$

$$\phi - coefficient = \frac{P(X,Y) - P(X)P(Y)}{\sqrt{P(X)[1-P(X)]P(Y)[1-P(Y)]}}$$

Example: Lift/Interest

	Coffee	$\overline{\text{Coffee}}$	
Tea	15	5	20
$\overline{\text{Tea}}$	75	5	80
	90	10	100

Association Rule: Tea → Coffee

Confidence= P(Coffee|Tea) = 0.75

but P(Coffee) = 0.9

⇒ Lift = 0.75/0.9= 0.8333 (< 1, therefore is negatively associated)

Drawback of Lift & Interest

	Y	$\overline{Y}$	
X	10	0	10
$\overline{X}$	0	90	90
	10	90	100

$$Lift = \frac{0.1}{(0.1)(0.1)} = 10$$

	Y	$\overline{Y}$	
X	90	0	90
$\overline{X}$	0	10	10
	90	10	100

$$Lift = \frac{0.9}{(0.9)(0.9)} = 1.11$$

Statistical independence:

If P(X,Y)=P(X)P(Y) => Lift = 1

Support-based Pruning

- Most of the association rule mining aalgorithms use support measure to prune rules and itemsets
- Study effect of dupport pruning on correlation of itemsets by generating 10000 random contingency tables, do computer support and pairwise correlation for each table, and apply support-based pruning and examine the tables that are removed - see results in slides

4.2 Cluster Analysis

Cluster analysis - finding groups of objects such that the objects in a group will be similar (or related) to one another and different from (or unrelated to) the objects in other groups. **Intra-cluster** distances are minimized and **Inter-cluster** distances are maximized.
Group together related documents for browsing, group genes, and proteins that have similar functionality, or group stocks with similar price fluctuations.
Reduces the size of large datasets.

What is not cluster analysis?

- Supervized classification - have class label info

- Simple segmentation - dividing students into different registration groups alphabetically by last name
- Results of a query - groupings are a result of an external specification
- Graph paritioning - some mutual relevance and synergy, but areas are not identical

Notion of a cluster can be Ambiguous.

A clustering is a set of clusters, there is an important distinction between hierarchical and partitional sets of clusters.

1. Paritional clustering - a division of data objects into non-overlapping subsets (clusters) such that each data object is in exactly one subset
2. Hierarchical clustering - a set of nested clusters organized as a hierarchical tree

Examples in the slide packet.

Other Distinctions Between Sets of Clusters

- Exclusive vs non-exclusive
  - In non-exclusive clusterings, points may belong to multiple clusters
  - Can represent multiple classes or 'border' points
- Fuzzy vs non-fuzzy
  - In fuzzy clustering, a point belongs to every cluster with some weight between 0 and 1
  - Weights must sum to 1
  - Probabilistic clustering has similar characteristics
- Partial vs complete - in some cases, we only want to cluster some of the data
- Heterogeneous vs homogeneous - cluster of widely different sizes, shapes, and densities

Types of Clusters

- Well-separated clusters
  - A cluster is a set of points such that any point in a cluster is closer (or more similar) to every other point in the cluster than to any point not in the cluster
- Center-based clusters

  - Aa cluster is a set of objects such that an object in a cluster is closer (more similar) to the "center" of a cluster, than to the center of any other cluster
  - The center of a cluster is often a centroid, the average of all the points in the cluster, or a medoid, the most "representative" point of a cluster

- Contiguous clusters
  - Aa cluster is a set of objects such that an object in a cluster is closer (more similar) to the "center" of a cluster, than to the center of any other cluster
  - Nearest neighbor or Transitive)

- Density-based clusters
  - A cluster is a dense region of points, which is separated by low-density regions, from other regions of high density
  - Used when the clusters are irregular or intertwined and when noise and outliers are present

- Property or Conceptual
  - Finds clusters that share some common property or represent a particular concept
  - Also called Shared Property

- Described by an Objective Function
  - Finds clusters that minimize or maximize an objective function
  - Enumerate all possible ways of dividing the points into clusters and evaluate the 'goodness' of each potential set of clusters by using the given objective function
  - Can have global or local objectives
    - Hierarchical clustering algorithms typically have local objectives
    - Partitional algorithms typically have global objectives
  - A variation of the global objective function approach is to fit the data to a parameterized model
    - Parameters for the model are determined from the data
    - Mixture models assume that the data is a 'mixture' of a number of statistical distributions

Characteristics of the Input Data are Important

- Type of proximity or density measure - this is a derived measure, but central to clustering

- Sparseness - dictates type of similarity, adds to efficiency
- Attribute type - dictates type of similarity
- Type of data - dictates type of similarity, other characteristics e.g. auto-correlation
- Dimensionality
- Noise and Outliers
- Type of Distribution

K-means Clustering

- Partitional clustering approach
- Each cluster is associated with a centroid (center point)
- Each point is assigned to the cluster with the closest centroid
- Number of clusters, K, must be specified
- Basic algorithm is simple - https://www.geeksforgeeks.org/k-means-clustering-introduction/
- Initial centroids are often chosen randomly - clusters produced vary from one run to another
- The centroid is (typically) the mean of the points in the cluster
- 'Closeness' is measured by Euclidean distance, cosine similarity, correlation, etc.
- K-means will converge for common similarity measures mentioned above
- Most of the convergence happens in the first few iterations - often the stopping condition is changed to 'until relatively few points change clusters'
- Complexity is O(n*K*I*d)
- There is optimal and sub-optimal clustering
- Choosing initial centroids is very important

Evaluating K-means Clusters

- Most common measure is Sum of Squared Error (SSE)
  - For each point, the error is the distance to the nearest cluster
  - To get SSE, we square these errors and sum them.

$$SSE = \sum_{i=1}^{K} \sum_{x \in C_i} dist^2(m_i, x)$$

    - x is a data point in cluster C_i and m_i is the representative point for cluster C_i
  - SSE is minimized if m_i is the mean of the points in C_i
  - From two clusterings, we can choose the one with the smaller error
  - One easy way to reduce SSE is to increase K
    - However, a good clustering with smaller K can have a lower SSE than a poor clustering with higher K

Problems with Selecting Initial Points

- If there are K 'real' clusters then the chance of selecting one centroid from each cluster is small.
  - Chance is relatively small when K is large
  - If clusters are the same size, n, then

$$P = \frac{\text{number of ways to select one centroid from each cluster}}{\text{number of ways to select } K \text{ centroids}} = \frac{K! n^K}{(Kn)^K} = \frac{K!}{K^K}$$

  - For example, if K = 10, then probability = $10!/10^{10} = 0.00036$
  - Sometimes the initial centroids will readjust themselves in 'right' way, and sometimes they don't
  - Consider an example of five pairs of clusters

Solutions to Initial Centroids Problem

- Multiple runs - helps but probability is not on your side
- Sample and use hierarchical clustering to determine initial centroids
- Select more than k initial centroids and then seelct amonf these initial centroids - select most widely separated
- Postprocessing
- Bisecting K-means - not as susceptible to initialization issues

Handlig Empty Clusters

- Basic K-means algorithm can yield empty clusters
- Several strategies
  - Choose the point that contributes most to SSE
  - Choose a point from the cluster with the highest SSE
  - If there are several empty clusteers, the above can be repeated several times

Updating Centers Incrementally

- In the basic K-means algorithm, centroids are updated after all points are assigned to a centroid
- An alternative is to update the centroids after each assignment (incremental approach)
  - Each assignment updates zero or two centroids
  - More expensive
  - Introduces an order dependency
  - Never get an empty cluster
  - Can use "weights" to change the impact

Pre-processinng

- Normalize the data
- Eliminate outliers

Post-processinng

- Eliminate small clusters that may represent outliers
- Split 'loose' clusters aka clusters with relatively high SSE
- Merge clusters that are 'close' and that have relatively low SSE
- Can use these steps during the clustering process - ISODATA

Bisecting K-means Algorithm - variant of K-means that can produce a partitional or a hierarchical clustering.

Limitations of K-means - K-means has problems when clusters are of differing sizes, densities, and non-globular shapes. K-means has problems when the data contains outliers.

One solution is to use many clusters, find parts of the cluster and then put together.

Hierarchical Clustering

- Produces a set of nested clusters organized as a hierarchical tree
- Can be visualized as a dendrogram - a tree like diagram that records the sequences of merges or splits

Strengths of Hierarchical Clustering

- Do not have to assume any particular number of clusters - any desired number of clusters can be obtained by 'cutting' the dendogram at the proper level

- They may correspond to meaaningful taxonomies - example in biological sciences like the animal kingdom

Two main types of hierarchical clustering

- Agglomerative
  - Start eith the points as individual clusters
  - At each step, merge the closest pair of clusters until only one cluster (or k clusters) remain
  - More popular hierarchical clustering technique
  - Basic algorithm is straightforward
    1. Compute the proximity matrix
    2. Let each data point be a cluster
    3. Repeat
    4. Merge the two closest clusters
    5. Update the proximity matrix
    6. Until only one single cluster remains
  - Key operation is the computation of the proximity of two clusters - different approaches to defining the distance between clusters distinguish the different algorithms
- Divisive
  - Start with one, all-inclusive cluster
  - At each step, split a cluster until each cluster contains only one point (or there are k clusters)
- Traaditional hierarchical algorithms use a similarity or distance matrix - merge or split one cluster at a time

Cluster Similarity: Ward's Method

- Similarity of two clusters is based onn the increase in squared error when two clusters are merged - similar to group average if distance between points is distance squared
- Less susceptible to noise and outliers
- Biased towards globular clusters
- Hierarchical analogue of K-means - can be used to initialize K-means

Hierarchical Clustering: Time and Space requirements

- $O(N^2)$ space since it uses the proximity matrix.
  - N is the number of points.
- $O(N^3)$ time in many cases
  - There are N steps and at each step the size, N^2, proximity matrix must be updated and searched
  - Complexity can be reduced to $O(N^2 \log(N))$ time for some approaches

DBSCAN - density-based algorithm

- Density = number of points withinn a specified radius (Eps)
- A point is a core point if it has more than a specified number of points (MinPts) within Eps - these are points that are at the interior of a cluster
- A border point has fewer thann MinPts within Eps, but is in the neighborhood of a core point
- A noise point is any point that is not a core point or a border point
- Algorithm
  - Label all points as either core, border or noise
  - Eliminate noise points
  - Put an edge between all core points that are within Eps of each other
  - Make each group of connnected core points into a separate cluster
  - Assign each border point to the cluster of one of the core points within Eps distance
- Resistance to noise
- Can handle clusters of different shapes and sizes
- Doesn't work well with varying densities or high-dimensional data
- Determining EPS and MinnPts

- Idea is that for points in a cluster, their k^{th} nearest neighbors are at roughly the same distance
- Noise poinnts have the k^{th} nearest neighbor at farther distance
- So, plot sorteed distance of every point to its k^{th} nearest neighbor

Cluster Validity

- For supervized classification we have a variety of measures to evaluate how good our model is - accuracy, precision, recall
- For clusters analysis, the analogous question is how to evaluate the "goodness" of the resulting clusters?
- But "clusters are in the eye of the beholder"
- We evaluate them to avoid finding patterns in noise, compare clustering algorithms, compare two sets of clusters, compare two clusters

Different Aspects of Cluster Validation

1. Determining the clustering tendency of a set of data i.e. distinguishing whether non-random structure actually exists in the data
2. Comparing the results of a cluster analysis to externally known results e.g. to externally given class labels
3. Evaluating how well the results of a cluster analysis fit the data without reference to external info - use only the data
4. Comparing the results of two different sets of cluster analyses to determine which is better
5. Determining the 'correct' number of clusters

For 2,3, and 4 we can further distinguish whether we want to evaluate the entire clustering or just individual clusters.

Measures of Cluster Validity

- Numerical measures that are applied to judge various aspects of cluster validity, are classified into the following three types
  - External Index: Used to measure the extent to which cluster labels match externally supplied class labels - Entropy
  - nternal Index: Used to measure the goodness of a clustering structure without respect to external information - Sum of Squared Error (SSE)
  - Relative Index: Used to compare two different clusterings or clusters - often an external or internal index is used for this function, e.g., SSE or entropy
- Sometimes these are referred to as criteria instead of indices
  - However, sometimes criterion is the general strategy and index is the numerical measure that implements the criterion

4.3 Principal Component Analysis

- Produces a low-dimensional representation of a dataset
- New features are a linear combination of the original ones
- New features have maximal variance and are mutuallyuncorrelated
- Useful for feature extraction/selection and datavisualization

First Principal Component: details

- The first principal component of a set of features $X_1, ..., X_d$ is the normalized linear combination of the features $z_1 = \phi_{11} * x_1 + .. + \phi_{d1} * x_d$ that has the largest variance
- Normalized means that $\sum_{j=1}^{d} \phi_{j1}^2 = 1$
- Note: Without normalization we could get an arbitrarily large variance by increase any ϕ_{j1}
- $\phi_{11}, ..., \phi_{d1}$ are called the loadings of the first principal component
- $\phi_1 = (\phi_{11}...\phi_{d1})^T$ is caled the loading vector of the first principal component

Computation of Principal Components

- Suppose we have a data set X with n instances and d attributes/variables
- Assume data is mean centered, i.e., each variable X_i ha smean zero
- The first principal component for sample i is $z_{i1} = \phi_{11}x_{i1} + ... + \phi_{d1}x_{id}$
- Goal: find normalized ϕ_{j1} such that variance of z_{i1} is maximal
- Since x_{ij} have zero mean (for each j), so do z_{i1} hence $variance(Z_1) = \frac{1}{n}\sum_{i=1}^{n} z_{i1}^2$

More information on the slides
How many principal components should weuse?

- Find out with cross-validation, if possible (e.g., in a supervised-learning setting, when using PCA for feature extraction)
- Look at the proportion of the variance in the data explained by each principal component

Summary

- PCA produces a low-dimensional representation of a dataset where new features are a linear combination of the original ones
- Idea: new features are uncorrelated and maximize the variance

- Requires data to be normalized (zero mean and similar standard deviation of all attributes)
- Proportion of variance explained is a measure for the importance of a principal component
- PCA is useful for feature extraction/selection and data visualization

5 Reinforcement Learning

Reinforcement Learning: learning from reward for a sequence of actions

5.1 rl1

What is Reinforcement Learning

- Agent-oriented learning—learning by interacting with an environment to achieve a goal - more realistic and ambitious than other kinds of machine learning
- Learning by trial and error, with only delayed evaluative feedback (reward) - the kind of machine learning most like natural learning, learning that can tell for itself when it is right or wrong
- The beginnings of a science of mind that is neither natural science nor applications technology

The RL Interface

The RL Interface

- Environment may be unknown, nonlinear, stochastic and complex
- Agent learns a policy mapping states to actions
  - Seeking to maximize its cumulative reward in the long run

Signature Challenges of RL

- Evaluative feedback (reward)
- Sequentiality, delayed consequences
- Need for trial and error, to explore as well as exploit
- Non-stationarity
- The fleeting nature of time and online data

In all success cases, performance was better than could be obtained by any other method, and was obtained without human instruction, examples are on the slides online.

5.2 rl2

The k-armed Bandit Problem

- On each of an infinite sequence of time steps, t=1, 2, 3, ..., you choose an action A_t from k possibilities, and receive a real-valued reward R_t

- The reward depends only on the action taken;it is indentically, independently distributed (i.i.d.): $q_*(a) = E[R_t|A_t = a], \forall a \in \{1, ..., k\}$ (true values)
- These true values are unknown. The distribution is unknown
- Nevertheless, you must maximize your total reward
- You must both try actions to learn their values (explore), and prefer those that appear best (exploit)

The Exploration/Exploitation Dilemma
You can't explore and exploit but you need to do both, can never stop exploring but maybe you should explore less with time or maybe not.

Action-Value Methods - Methods that learn action-value estimates and nothing else.

ϵ-Greedy Action Selection

- In greedy action selection, you always exploit
- In ϵ-greedy, you are usually greedy, but with probability ϵ you instead pick an action at random (possibly the greedy action again)
- This is perhaps the simplest way to balance exploration and exploitation

A simple bandit algorithm

Initialize, for $a = 1$ to k:
$Q(a) \leftarrow 0$
$N(a) \leftarrow 0$

Repeat forever:
$$A \leftarrow \begin{cases} \arg\max_a Q(a) & \text{with probability } 1 - \varepsilon \quad \text{(breaking ties randomly)} \\ \text{a random action} & \text{with probability } \varepsilon \end{cases}$$
$R \leftarrow bandit(A)$
$N(A) \leftarrow N(A) + 1$
$Q(A) \leftarrow Q(A) + \frac{1}{N(A)}\big[R - Q(A)\big]$

Averaging ⟶ learning r

- To simplify notation, let us focus on one action
  - We consider only its rewards, and its estimate after $n+1$

$$Q_n \doteq \frac{R_1 + R_2 + \cdots + R_{n-1}}{n-1}$$

- How can we do this incrementally (without storing all t
- Could store a running sum and count (and divide), or e

$$Q_{n+1} = Q_n + \frac{1}{n}\Big[R_n - Q_n\Big]$$

- This is a standard form for learning/update rules:

$$NewEstimate \leftarrow OldEstimate \; + \; StepSize \; \Big[Target - O$$

Averaging → learnning rule

Initialize action values optimistically.

Upper Confidence Bound (UCB action selection

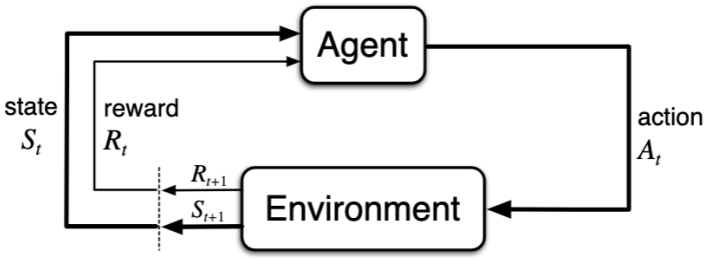
- A clever way of reducing exploration over time
- Estimate an upper bound on the true action values
- Select the action with the largest (estimated) upper bound

Conclustion

- These are all simple methods but they are complicated enough - we will build on them, we should understand them completely, there are still open questions
- Our first algorithms that learn from evaluative feedback and thus must balance exploration and exploitation
- Our first algorithms that appear to have a goal - that learn to maximize reward by trial and error

5.3 rl3

The Agent-Environment Interface



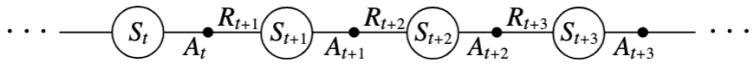
Agent and environment interact at discrete time steps: $t = 0, 1, 2, 3, \ldots$

Agent observes state at step t: $S_t \in \mathcal{S}$

produces action at step t : $A_t \in \mathcal{A}(S_t)$

gets resulting reward: $R_{t+1} \in \mathcal{R} \subset \mathbb{R}$

and resulting next state: $S_{t+1} \in \mathcal{S}^+$



Markov Decision Process

- If a reinforcement learning task has the Markov Property, is it baasically a Markov Decision Process (MDP)
- If state and action sets are finite, it is a finite MDP
- To definne a finite MDP, you need to give state and action sets as well as one-step "dynamics"

Reinforcement learning methods specify how the agent chaanges its policy as a result of experience. Roughly the agent's goal is to get as much reward as it can over the long run.

Always seek to maximize the expected return. **Total rewaard** is the sum of all future reward inn teh episode. **Discounted reward** is the sum of all future discounted reward. **Average rewaard** is average reward per time step.

Episodic Tasks - interaction breaks naturally into episodes. We almost always use simple total reward from beginning state to terminal state.

Continuing Tasks - interaction does not have natural episodes but just goes on and on. We then use discounted return using $0 \leq \gamma \leq 1$ discount rate (gamma). Gamma closer to 0 is shorsighted and closer to 1 is farsighted. Typically $\gamma = 0.9$.

A Trick to Unify Notation for Returns

- In episodic tasks, we number the time steps of each episode starting from 0
- We usually do not have to distinguish between episodes so instead of writing $S_{t,j}$ for states in episode j, we write just S_t
- Think of each episode as ending in an absorbing state that always produces reward of zero
- We can cover all cases by writing $G_t = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$ where γ can be 1 only if a zero reward absorbing state is always reached.

4 value functions

	state values	action values
prediction	v_π	q_π
control	v_*	q_*

- All theoretical objects, mathematical ideals (expected values)
- Distinct from their estimates:

$$V_t(s) \qquad Q_t(s,a)$$

Optimal Value Functions

- For finite MDPs, policies can be partially ordered: $\pi \geq \pi'$ if and only if $v_\pi(s) \geq v_{\pi'}(s)$ for all $s \in S$
- There are always one or more policies that are better than or equal to all the others. These are the optimal policies. We denote them all π_*
- Optimal policies share the same optimal state-value function: $v_*(s) = max_\pi v_\pi(s)$ for all $s \in S$

- Optimal policies also share the same optimal action-value function: $q_*(s,a) = max_\pi q_\pi(s,a)$ for all $s \in S$ and $a \in A$. This is the expected return for taking action a in state sand thereafter following an optimal policy.

Optimal value policies make everything easier. Optimal action-value functions help the agent not have to do a one-step-ahead search.

Bellman Optimality Equation - check slides.

Solving the Bellman Optimality Equation

- Finding an optimal policy by solving the Bellman Optimality Equation requires the following:
  - accurate knowledge of environment dynamics
  - we have enough space and time to do the computation
  - the Markov Property
- How much space and time do we need? polynomial in number of states (via dynamic programming methods; Chapter 4) BUT, number of states is often huge (e.g., backgammon has about 1020 states)
- We usually have to settle for approximations
- Many RL methods can be understood as approximately solving the Bellman Optimality Equation.

5.4 rl4 - Dynamic Programming

Generalized Policy Iteration (GPI) - any interaction of policy evaluation and policy improvement independent of their granularity.

Asynchronous DP

- All the DP methods described so far require exhaustive sweeps of the entire state set
- Asynchronous DP does not use sweeps. Instead it works like this: Repeat until convergence criterion is met - Pick a state at random and apply the appropriate backup
- Still need lots of computation, but does not get locked into hopelessly long sweeps
- Can you select states to backup intelligently? YES: an agent's experience can act as a guide.

Efficiency of DP

- To find an optimal policy is polynomial in the number of states...

- BUT, the number of states is often astronomical, e.g., often growing exponentially with the number of state variables (what Bellman called "the curse of dimensionality")
- In practice, classical DP can be applied to problems with a few millions of states
- Asynchronous DP can be applied to larger problems, and is appropriate for parallel computation
- It is surprisingly easy to come up with MDPs for which DP methods are not practical

Summary

- Policy evaluation: backups without a max
- Policy improvement: form a greedy policy, if only locally
- Policy iteration: alternate the above two processes
- Value iteration: backups with a max
- Full backups (to be contrasted later with sample backups)
- Generalized Policy Iteration (GPI)
- Asynchronous DP: a way to avoid exhaustive sweeps
- Bootstrapping: updating estimates based on other estimates
- Biggest limitation of DP is that it requires a probability model (as opposed to a generative or simulation model)

5.5 rl5 - Monte Carlo Methods

- Monte Carlo methods are learning methods: Experience →
- Monte Carlo methods can be useed in two ways:
  - model-free: No model necessary and still attains optimality
  - simulated: Needs only a simulation, not a full model
- Monte Carlo methods learn from complete sample returns - only defined for episodic tasks
- Like an associative version of a bandit method
- Goal: learn $v_\pi(s)$
- Given: some number of episodes under π which contains s
- Idea: average returns observed after visits to s

- Every-visit MC: average returns for every time s is visited inn ann episode
- First-visit MC: average returns only for first time s is visited in an episode
- Both converge asymptotically

Blackjack in the slides.

Monte Carlo Estimation of Action Values (Q)

- Monte Carlo is most useful when a model is not available - we want to learn q_*
- $q_\pi(s, a)$ - average reutrn starting from state s and action a following π
- Converges asymptotically if every state-action pair is visited
- Exploring starts: Every state-action pair has a non-zero probability of being the starting pair

MC policy iteration: Policy evaluation using MC methods followed by policy improvement
Policy improvement step: greedify with respect to value (or action-value) function

Convergence of MC control on the slides.

Off-policy methods

- Learn the value of the target policy π from experience due to behavior policy μ
- For example, π is the greedy policy (and ultimately the optimal policy) while μ is exploratory (e.g., ϵ-soft)
- In general, we only require coverage i.e., that μ generates behavior that covers or includes π, $\mu(a|s) > 0$ for every s,a at which $\pi(a|s) > 0$
- Idea: importance sampling - weight each return by the ratio of the probabilities of the trajectory under the two piles

Summary

- MC has several advantages over DP:
  - Can learn directly from interaction with environment
  - No need for full models
  - No need to learn about ALL states (no bootstrapping)
  - Less harmed by violating Markov property (later in book)

- MC methods provide an alternate policy evaluation process
- One issue to watch for: maintaining sufficient exploration - exploring starts, soft policies
- distinction between on-policy and off-policy methods
- ordinary vs weighted IS
- return-specific ideas for reducing IS vaariance - discounting-aware vs per-reward IS

5.6 rl6 - Temporal Difference Learning

TD methods bootstrap and sample

- Bootstrapping: update involves an estimate of the value function
  - TD and DP methods bootstrap
  - MC methods do not bootstrap
- Sampling: update does not involve an expected value
  - TD and MC method sample
  - Classical DP does not sample

Updating our predictions Goal: update the prediction of total time leaving from office, while driving home.
With MC we would need to wait for termination (arriving home) then calculate G_t for each step of the episode, then apply our updates.

Advantages of TD learning

- TD methods do not require a model of the environment, only experience
- TD methods can be fully incremental
  - Make updates before knowing the final outcome
  - Requires less memory
  - Requires less peak computation
- You can learn without the final outcome, from incomplete sequences
- Both MC and TD converge (under certain assumptions to be detailed later), but which is faster? TD converges faster because it computes the true certainty-equivalence estimate.

Batch updating: train completely on a finite amount of data, e.g. train repeatedly on 10 episodes until convergence. Compute updates according to TD or MC, but only update estimates after each complete pass through the data. For any finite Markov prediction task, under batch updating, TD converges for sufficiently small α. Constant- MC also converges under these conditions, but to a different answer.

Advantages of TD

- If the process is Markov, then we expect the TD estimate to produce lower error on future data
- This helps explain why TD methods converge more quickly than MC in the batch setting
- TD(0) makes progress towards the certainty-equivalence estimate without explicitly building the model!

Sars: On-Policy TD Control - Turn this into a control method by always updating the policy to be greedy with repsect to the current estimate.

Q-learning: Off-Policy TD Control - does better in cliff walking//
Summary

- Introduced one-step tabular model-free TD methods
- These methods bootstrap and sample, combining aspects of DP and MC methods
- TD methods are computationally congenial
- If the world is truly Markov, then TD methods will learn faster than MC methods
- MC methods have lower error on past data, but higher error on future data

5.7 rl9_10 - On-policy Prediction and Control with Approximation

On-policy Prediction with Approximation

Value function approximation (VFA) replaces the table with a general parameterized form.
Stochastic Gradient Descent (SGD) is the idea behind most approximate learning.
A natural objective in VFA is to minimize the Mean Square Value Error.
State aggregation is the simplest kind of VFA - states are partitioned into disjoint subsets (groups), one component θ is allocated to each group.

Gradient MC works well on the 1000-state random walk usinng state aggregation.
Gradient TD is less accurate than MC on the 1000-state random walk usinng state aggregation.
Bootstrapping still greatly speeds learning.
With binary features, a continuous state space can be coarsly coded, adding generalization.
The width of the receptive fields determines breadth of generalization.
Tile codinng is coarse coding for digital computers with rectangular receptive fields, controlled overlap. Nevertheless, tile coding is very flexible. Tile coding works better than state aggregation on the 1000-state random walk.
Smooth-edged receptive fields are a little different but increase computational compllexity.

- Value-function approximation by stochastic gradient descent enables RL to be applied to arbitrarily large state spaces
- Most algorithms just carry over the Targets from the tabular case
- With bootstrapping (TD), we don't get true gradient descent methods
  - this complicates the analysis
  - but the linear, on-policy case is still guaranteed convergent
  - and learning is still much faster
- For continuous state spaces, coarse/tile coding is a good strategy
- For ambitious AI, artificial neural networks are an interesting strategy

On-policy Control with Approximation

Value function approximation (VFA) replaces the table with a general parameterized form
(Semi-)gradient methods carry over to control in the usual on-policy GPI way

- Always learn the action-value function of the current policy
- Always act near-greedily wrt the current action-value estimates

Values learned while solving Mountain-Car with tile coding function approximation

- (Semi-)gradient methods carry over to control in the usual way (Mountain Car example)
- n-step methods carry over too, with the usual tradeoffs
- A new average-reward setting, with differential value functions and differential algorithms (Queuing example (tabular))
- The discounting setting is deprecated

6 Evolutionary Computing

6.1 ec1 - Introduction

Positioning of EC

- EC is part of computer science
- EC is not part of life sciences/biology
- Biology delivered inspo and terminology
- EC can be applied in biological research

Evolution = Problem Solving; Environment = Problem; Individual = Candidate Solution; Fitness = Quality

Fitness = chances for survival and reproduction
Quality = chance for seeding new solutions

Adaptive lnadscape metaphor

- Can envisage population with ntraits as existing in a n+1-dimensional space (landscape) with height corresponding to fitness
- Each different individual (phenotype) represents a single point on the landscape
- Population is therefore a "cloud" of points, moving on the landscape over time as it evolves - adaptation
- Selection "pushes" population up the landscape
- Genetic drift:
  - random variations in feature distribution (+ or -) arising from sampling error
  - can cause the population "melt down" hills, thus crossing valleys and leaving local optima

Problem type 1: Optimization - we have a model of our system and seek inputs that give us a specified goal.
Optimization example: Satellite Structure

- Optimised satellite designs for NASA to maximize vibration isolation
- Evolving: design structures
- Fitness: vibration resistance
- Evolutionary "creativity"

Problem type 2: Modeling - we have corresponding sets of inputs outputs and seek model thatdeliverscorrect output for every known input, evollutionary machine learnning.

Modelling example: loan applicant creditibility

- British bank evolved creditability model to predict loan paying behavior of new applicants
- Evolving: prediction models
- Fitness: model accuracy on historical data

Problem type 3: Simulation - We have a given model and wish to know the outputs that arise under different input conditions, Often used to answer "what-if" questions in evolving dynamic environments, evolutionary economics, artificial life.

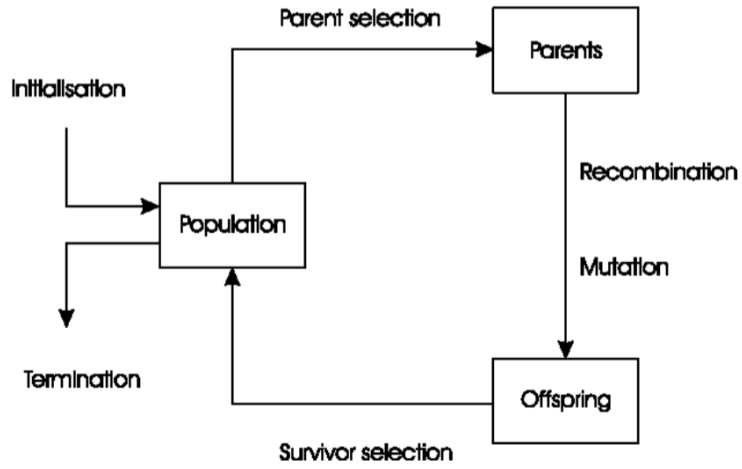
6.2 ec2 - What is an Evolutionary Algorithm?

Recap of EC metaphor

- A population of individuals exists in an environment with limited resources
- **Competition** for those resources causes selection of those fitter individuals that are better adapted to the environment
- These individuals act as seeds for the generation of new individuals through recombination and mutation
- The new individuals have their fitness evaluated and compete (possibly also with parents) for survival.
- Over time Natural selection causes a rise in the fitness of the population
- EAs fall into the category of "generate and test" algorithms
- They are stochastic,population-based algorithms
- Variation operators (recombination and mutation) create the necessary diversity and thereby facilitate novelty
- Selection reduces diversity and acts as a force pushing quality

General Scheme of EAs

General Scheme of EAs



What are different types of EAs

- Historically different flavours of EAs have been associated with different representations
  - Binary strings : Genetic Algorithms
  - Real-valued vectors : Evolution Strategies
  - Finite state Machines: Evolutionary Programming
  - LISP trees: Genetic Programming
- These differences are largely irrelevant, best strategy
  - choose representation to suit problem
  - choose variation operators to suit representation
- Selection operators only use fitness and so are independent of representation

Representations

- Candidate solutions (individuals) exist in phenotype space
- They are encoded in chromosomes, which exist in genotypespace

  - Encoding : phenotype => genotype (not necessarily one to one)
  - Decoding : genotype => phenotype (must be one to one)

- Chromosomes contain genes, which are in (usually fixed) positions called loci(sing. locus) and have a value (allele)
- **In order to find the global optimum, every feasible solution must be represented in genotype space**

Evaluation (Fitness) Function

- Represents the requirements that the population should adapt to
- a.k.a. quality function or objective function
- Assigns a single real-valued fitness to each phenotype which forms the basis for selection - so the more discrimination (different values) the better
- Typically we talk about fitness being maximised - some problems may be best posed as minimisation problems, but conversion is trivial

Population

- Holds (representations of) possible solutions
- Usually has a fixed size and is a multiset of genotypes
- Some sophisticated EAs also assert a spatial structure on the population e.g.,a grid
- Selection operators usually take whole population into account i.e.,reproductive probabilities are relative to current generation
- Diversity of a population refers to the number of different fitnesses/phenotypes/genotypes present (note not the same thing)

Parent Selection Mechanism

- Assigns variable probabilities of individuals acting as parents depending on their fitnesses
- Usually probabilistic
  - high quality solutions more likely to become parents than low quality
  - but not guaranteed
  - even worst in current population usually has non-zero probability of becoming a parent
- This stochastic nature can aid escape from local optima

Variation Operators

- Role is to generate new candidate solutions
- Usually divided into two types according to their arity (number of inputs):
  - Arity 1 : mutation operators
  - Arity > 1 : Recombination operators
  - Arity $= 2$ typically called crossover
- There has been much debate about relative importance of recombination and mutation
  - Nowadays most EAs use both
  - Choice of particular variation operators is representation dependant

Mutation

- Acts on one genotype and delivers another
- Element of randomness is essential and differentiates it from other unary heuristic operators
- Importance ascribed depends on representation and dialect:
  - Binary GAs - background operator responsible for preserving and introducing diversity
  - EP for FSM's/continuous variables - only search operator
  - GP - hardly used
- May guarantee connectedness of search space and hence convergence proofs

Recombination

- Merges information from parents into offspring
- Choice of what information to merge is stochastic
- Most offspring may be worse, or the same as the parents
- Hope is that some are better by combining elements of genotypes that lead to good traits
- Principle has been used for millennia by breeders of plants and livestock

Survivor Selection

- a.k.a. replacement
- Most EAs use fixed population size so need a way of going from (parents + offspring) to next generation
- Often deterministic

  - Fitness based: e.g.,rank parents+offspring and take best
  - Age based: make as many offspring as parents and delete all parents
- Sometimes do combination (elitism)

Initialization - usually done at random

- Need to ensure even spread and mixture of possible allele values
- Can include existing solutions, or use problem-specific heuristics, to "seed" the population

Termination - condition checked every generation

- Reaching some (known/hoped for) fitness
- Reaching some maximum allowed number of generations
- Reaching some minimum level of diversity
- Reaching some specified number of generations without fitness improvement

See the 8 Queens problem in the slides.

Typical behavior of an EA
Early phase: quasi-random population distribution
Mid-phase: population arranged around/on hills
Late phase: population concentrated on high hills

Evolutionary Algorithms in Context

- There are many views on the use of EAs as robust problem solving tools
- For most problems a problem-specific tool may:
  - perform better than a generic search algorithm on most instances,
  - have limited utility,
  - not do well on all instances
- Goal is to provide robust tools that provide:
  - evenly good performance
  - over a range of problems and instances

6.3 ec3 - Genetic Algorithms

Attributed features - not too fast, good heuristic for combinatorial problems

7 Quiz

7.1 Quiz 1

1. Which of these statements is true about nominal (categorical) values?
    A. Their median can be calculated in a meaningful way.
    B. They can be ordered in a meaningful way.
    C. **Their mode can be calculated in a meaningful way.**
    D. Their mean can be calculated in a meaningful way.

2. Which of the following statements is not true about ordinal values?
    A. **The mean of such values can be calculated.**
    B. They are categorical.
    C. The median of such values can be calculated.
    D. They can be ordered in a meaningful way.

3. What is the dimensionality of record data?
    A. the range of possible values of the attributes
    B. 2, because record data is a table
    C. the number of records
    D. **the number of attributes**

4. Supervised learning differs from unsupervised learning in that supervised learning requires ...
    A. **at least one output attribute**
    B. output attributes to be categorical
    C. at least one input attribute
    D. input attributes to be categorical

5. Which of these measures is most sensitive to outliers?
    A. **mean**
    B. mode
    C. median
    D. interquartile range

6. Which of these is not a measure for the spread of the data?
    A. interquartile range
    B. variance
    C. standard deviation
    D. **mean**

7.2 Quiz 2

1. An internal node in a decision tree represents ...
    A. a class
    B. a number of instances
    C. an attribute value
    D. **a test specification**

2. A leaf node in a decision tree represents ...
    A. **a class**
    B. a test specification
    C. a number of instances
    D. an attribute value

3. Which of the following cases indicates overfitting?
    A. high training error, low test error
    B. low training error, high test error
    C. **low training error, high test error**
    D. low training error, low test error
    E. high test error, the training error does not matter
    F. high training error, high test error

4. Which of the following cases indicates underfitting?
    A. low training error, high test error
    B. high test error, the training error does not matter
    C. high training error, low test error
    D. **high training error, high test error**
    E. low training error, low test error

5. You are training a decision tree classifier with a minimal number of instances n that are required for making a new decision node. The results from cross-validation indicate overfitting. To reduce overfitting you ...
    A. decrease n
    B. change some other parameter, because n does not influence overfitting
    C. try out both larger and smaller n, because it is not clear which one will reduce overfitting
    D. **increase n**

7.3 Quiz 3

1. Accuracy is a misleading measure for the performance of a classifier when ...
    A. the classifier is underfitting
    B. the classifier is overfitting
    C. we have many different classes
    D. **we do not have similar numbers of instances for each class**

2. Assume a classification problem with three classes A, B, and C. The false positives of a classifier wrt. class A are ...
    A. **the instances in classes B or C that are classified as A.**
    B. depending on whether we want the false positives of A vs. B or A vs. C.
    C. The instances in B classified as C.
    D. The instances in A classified as A.
    E. the instances in class A that are classified as B or C.

3. Assume a classification problem with three classes A, B, and C. The false negatives of a classifier wrt. class A are ...
    A. The instances in B classified as C and the instances in C classified as B.
    B. depending on whether we want the false negatives of A vs. B or A vs. C.
    C. **the instances in class A that are classified as B or C.**
    D. the instances in classes B or C that are classified as A.

4. Our training data consists of the points plotted in the following chart 1. Which class would kNN with the Euclidian distance predict for the unlabeled data point (x,y)=(3,6) with k=3 (assuming majority voting with equal weights among neighbors).
    A. **A**
    B. C
    C. no prediction because value is out of range
    D. B

5. Our training data consists of the points plotted in the following chart 2. Which class would kNN with the Euclidian distance predict for the unlabeled data point (x,y)=(6,1) with k=10 (assuming majority voting with equal weights among neighbors).
    A. A

B. **C**

C. B

D. 50% B and 50% A

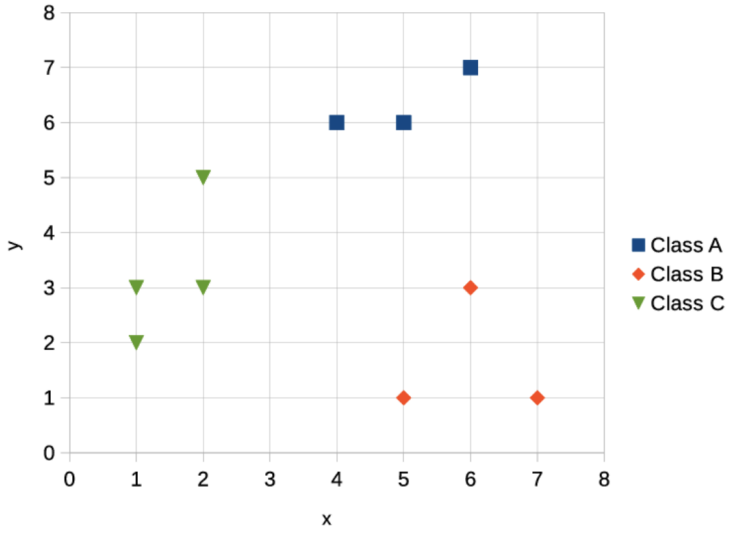
6. You have trained a kNN classifier and notice that it is underfitting. How can you try to address the problem? (Mark all that apply)

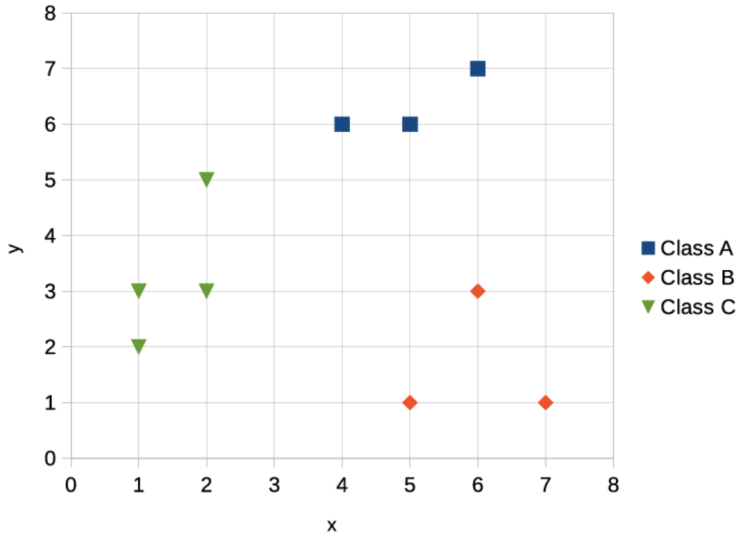
   A. **decrease k**

   B. **filter out irrelevant attributes**

   C. **filter out redundant attributes**

   D. increase k





7.4 Quiz 4

1. What is the main underlying assumption that is used for learning a naive-Bayes classifier?
    A. **All attributes are conditionally independent given the class label.**
    B. None of the others.
    C. All attributes have discrete values.
    D. All attributes are independent from the class.

2. A support vector machine can deal with data that is not linearly separable by ... (select all that apply)
    A. automatically finding outliers and ignoring them during training.
    B. **using a kernel function to learn a linear decision boundary in a higher dimensional space than the input space.**
    C. normalizing the input attributes.
    D. **introducing a slack parameter for each example and minimizing the total amount of slack.**

3. Which of the following statements about the bagging technique is not true?
    A. It is less susceptible to noisy data than using a single classifier.
    B. The training samples all have the same weight.
    C. The output of the ensemble is a majority vote of the individual classifiers.
    D. **The weak classifiers are weighted by the prediction accuracy.**

4. Which of the following statements about Adaboost / the boosting technique is true?
    A. **The predicted class is determined by a weighted voting scheme.**
    B. The training samples all have the same weight.
    C. The weak classifiers in the ensemble need to be of the same type (e.g., all decision trees).
    D. Adaboost is very efficient at learning because the members of the ensemble can be trained in parallel.

5. Regression differs from classification in that we ...
    A. iteratively try to improve our model.
    B. can use gradient descent.

C. **learn a function with numeric output.**

D. learn a function with parameters.

6. You run gradient descent with learning rate $\alpha = 0.3$ for 15 iterations and compute $E(\vec{w})$ after every iteration. You find that $E(\vec{w})$ increases over time. Which conclusions seems most plausible?

   A. **It would be better to decrease the learning rate.**

   B. The kind of function we try to learn does not fit the data well.

   C. It would be better to increase the learning rate.

   D. The current learning rate is just fine.

7. True or False? When using gradient descent to fit a function $h_{\vec{w}}(\vec{x}) = y$ to data, it does not matter which initial values we choose for the parameters $\vec{w}$.

   A. True

   B. **False**

7.5 Quiz 5

1. Which statement is true about the K-Means algorithm?

   A. The output attribute must be categorical.

   B. Attribute values may be either categorical or numeric.

   C. All attribute values must be categorical.

   D. **All attributes must be numeric.**

2. The K-Means algorithm terminates when ...

   A. the number of instances in each cluster for the current iteration is identical to the number of instances in each cluster of the previous iteration.

   B. **the cluster centers for the current iteration are identical to the cluster centers for the previous iteration.**

   C. a user-defined minimum value for the sum of squared distances between instances and their corresponding cluster center is seen.

   D. the number of clusters formed for the current iteration is identical to the number of clusters formed in the previous iteration.

3. For the following data set and initial centroids, which two points are more likely as positions of the centroids after running the KMeans algorithm? See 1 image below.

   A. (1,-1) and (1,1)

   B. (-1,-1) and (1,1)

   C. **(-1,0) and (1,0)**

D. (0,-1) and (0,1)

4. For the following data set and initial centroids, which two points are more likely as positions of the centroids after running the KMeans algorithm? See 2

   A. **(0,-4) and (0,4)**

   B. (-2,-4) and (2,4)

   C. (-2,0) and (2,0)

   D. (-2,2) and (2,-2)

5. Which of the following is not true? Principal component analysis maps data into a lower dimensional space such that ...

   A. **the new features have a high correlation with the target output.**

   B. the new features are linear combinations of the original ones.

   C. the new features are uncorrelated.

   D. the new features explain most of the variance in the data.

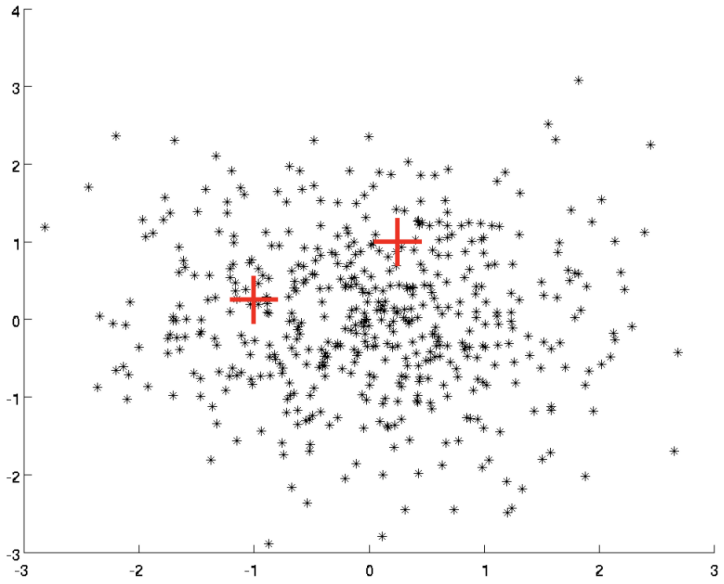


Figure 1: 5-3

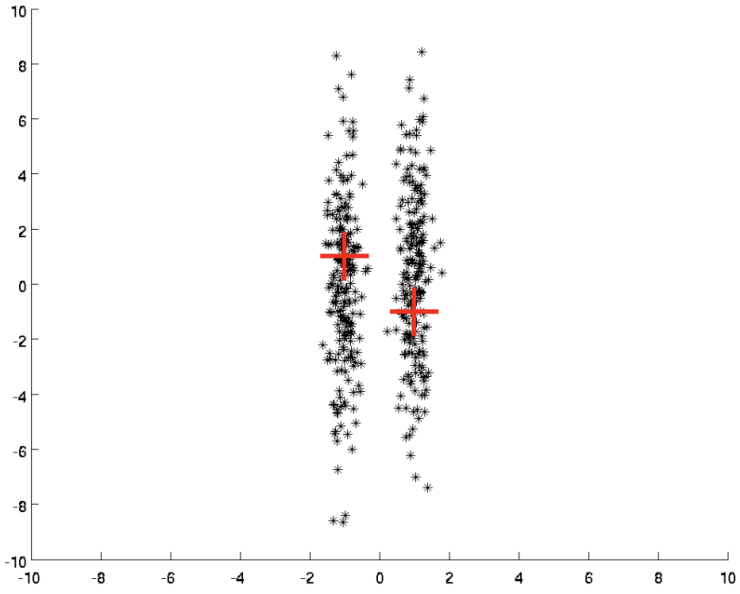


Figure 2: 5-4

7.6 Quiz 7 - Reinforcement Learning

1. In reinforcement learning we often need to balance exploitation vs. exploration. In this context, what is exploitation?
    A. **selecting the action with highest estimated value**
    B. achieving the highest cumulative reward
    C. greedily selecting the optimal action
    D. finding a loophole in the problem that allows us to always win
2. Match the symbols in the Bellman equation

$$v_\pi(s) = \Sigma_a \pi(a|s) \Sigma_{s`,r} p(s`, r|s, a)[r + \gamma v_\pi(s`)]$$

    to their respective meaning.

    - $v_\pi(s)$ - expected return in state s
    - $\pi(a|s)$ - policy
    - $p(s`, r|s, a)$ - transition probability of the Markov process
    - r - reward for the state transition
    - γ - discount factor for future rewards

3. Any policy that, in every state s, is greedily selecting the action a with the highest $q_*(s, a)$, is an optimal policy.
    A. **True**
    B. False
4. Since policies are only partially ordered, there can be many different optimal policies with different value functions.
    A. True
    B. **False**

7.7 Quiz 8 - Reinforcement Learning 2

1. Which of the following statements about Value Iteration are true?
    A. Value iteration only converges, if the learning rate LaTeX: α is small enough
    B. **Value iteration is a dynamic programming method for computing an optimal policy.**
    C. One advantage of Value Iteration is that it only needs to look at each state once.
    D. **Value Iteration does not have the exploration / exploitation dilemma, because it always considers all possible actions.**

2. Which of the following conditions need to be fulfilled to be able to use Value Iteration for a particular problem?
    A. **The state space needs to be finite and reasonably small.**
    B. All state variables (attributes) need to be numeric and normalized.
    C. **We need a Markov model of the environment with know transition probabilities.**
    D. **There must only be finitely many possible actions.**
3. Which of the following statements about Monte Carlo Control are true?
    A. Monte Carlo methods only work in Casinos.
    B. **Monte Carlo methods need to use exploration because they only look at one possible action and successor state at a time.**
    C. Monte Carlo methods work by usually picking the action that lead to the successor state with the highest estimated value.
    D. **Monte Carlo methods learn the values of states or actions from observing returns of complete episodes.**
4. Which of the following statements about Bootstrapping are true?
    A. **Dynamic Programming methods use bootstrapping.**
    B. Bootstrapping can only be used in deterministic environments (environments where it is know which successor state we reach with a particular action).
    C. **Bootstrapping means that we use estimated values of states or actions to update the estimates of the values of other states or actions**
    D. Monte Carlo methods use bootstrapping.

7.8 Quiz 9

1. Which reinforcement learning method belongs to this update equation:

$$Q(s,a) \Leftarrow Q(s,a) + \alpha[R + \gamma max_{a'} Q(a', s') - Q(s,a)]$$

    A. on-policy Monte-Carlo control
    B. **Q-Learning**
    C. dynamic programming value iteration
    D. Sarsa
    E. TD(0) policy evaluation

2. Which reinforcement learning method belongs to this update equation:

$$V(s) \Leftarrow max_a \Sigma_{s',r} p(s', r|s, a)[r + \gamma V(s')]$$

   A. **dynamic programming value iteration**
   B. TD(0) policy evaluation
   C. on-policy Monte-Carlo control
   D. Q-Learning
   E. Sarsa

3. Which reinforcement learning method belongs to this update equation:

$$Q(s, a) \Leftarrow Q(s, a) + \alpha[R + \gamma Q(a', s') - Q(s, a)]$$

   A. n-policy Monte-Carlo control
   B. Monte-Carlo policy evaluation
   C. **Sarsa**
   D. Q-Learning
   E. dynamic programming value iteration

4. Which reinforcement learning method belongs to this update equation:

$$Q(s, a) \Leftarrow Q(s, a) + \alpha[G - Q(s, a)]$$

   A. TD(0) policy evaluation
   B. dynamic programming value iteration
   C. **on-policy Monte-Carlo control**
   D. Sarsa
   E. Q-Learning

5. Which of the following three algorithms will learn a policy closest to the optimal policy for a given MDP, assuming an epsilon-greedy policy with epsilon=0.1 is used for generating episodes of the MDP and the learning rate is small enough?

   A. On-Policy Monte-Carlo Control
   B. Sarsa
   C. **Q-Learning**

7.9 Quiz 10

1. Which of the following statements are true about the initialization phase of an evolutionary algorithm?
    A. **Initialization is concerned with generating candidate solutions.**
    B. **Heuristics for generating candidates can be applied.**
    C. Mutation of candidates is normally also taking place during the initialization.
    D. **Individuals are normally generated randomly.**
2. Which of the following statements are true about the variation operators of an evolutionary algorithm?
    A. Variation operators act on population level.
    B. Variation operators are used for parent selection
    C. **Variation operators are used to generate off-spring**
    D. **Crossover and mutation are variation operators.**
3. Which of the following statements are true about recombination in the context of evolutionary algorithms?
    A. **Recombination is mainly responsible for exploration.**
    B. Recombination is also known as mutation.
    C. **Recombination combines elements of two or more genotypes.**
    D. Recombination operators are independent of the representation of candidate solutions.
4. Which of the following statements are true about survivor selection in the context of evolutionary algorithms?
    A. Survivor selection is also known as replacement strategy.
    B. **Survivor selection can be fitness based.**
    C. **Survivor selection can be age based.**
    D. Survivor selection is often stochastic.
5. Selection pressure describes the idea that individuals with higher fitness have a higher chance to reproduce and/or survive. In evolutionary algorithms which statements about selection pressure are true?
    A. **Fitness-proportionate selection can lead to too much selection pressure.**
    B. **Rank-based selection can adjust and control the pressure.**

C. Selection pressure should be high to avoid premature convergence.

D. **Fitness-proportionate selection can lead to loss of selection pressure.**