Artificial Intelligence

wu

June 29, 2019

Contents

1 scope [28%]

☒ uninformed search and informed search

☒ adversarial search: minimax, evaluation functions, alpha-beta search, stochasitc search

⊟ Basic concept for Statistical Learning and modeling

- ▫ Probability Theory
- ▫ Model selection
- ☒ The curse of Dimensionality
- ▫ Decision Theory
- ☒ Information Theory
- ▫ Probability Distribution

- ▫ Supervised Learning
  - ▫ Linear model for regression
  - ▫ Linear basis function models
  - ▫ Linear model for classification
  - ▫ Adaboosting
- ▫ Unsupervised Learning
  - ▫ K-means Clustering
  - ▫ GMM & EM algorithm
  - ▫ Principal Component Analysis
- ▫ Deep Learning
  - ▫ Stochastic Gradient Descent
  - ▫ Backpropagation
  - ▫ Feedforward Neural Network
  - ▫ Convolutional Neural Networks
  - ▫ Deep learning in CV (localization)
  - ▫ Recurrent Neural Network (LSTM, GRU)
- ▫ Reinforcement learning
  - ▫ Reinforcement learning
  - ▫ Markov Decision Process
  - ▫ Value-based Optimization; Q-learning

2 Basic concepts for filling in blanks or single choice [33%]

- ☒ Uninformed search (blind search)
  - ☒ Problem definition, initial state, actions, transition model, goal test, path cost, step cost, frontier (open list), loopy path, explored set (closed set), tree-search, graph-search, queue, completeness, optimality, time complexity, space complexity
  - ☒ Breadth-first search (BFS)

- ☒ Depth-first search (DFS)
- ☒ Uniform-cost search (UCS), Depth-limited search (DLS), iterative deepening search (IDS)

☒ informed (heuristic) search

- ☒ Heuristic function h(n), evaluation function f(n), path cost g(n)
- ☒ Best-first search: use f(n) instead of g(n)
- ☒ Greedy best-first search: f(n) = h(n)
- ☒ A* search: f(n) = g(n) + h(n), it is identical to Uniform-Cost-Search except that A* uses g+h instead of g
- ☒ Conditions for A* optimality: Admissibility and Consistency **Noteeeeeeeeee**

☒ Adversarial search

- ☒ Minimax search: min node, max node, utility
- ☒ Alpha-Beta Pruning Search: alpha value, beta value, pruning **STARRRRRRRRRRRRRRR**

☒ Uninformed search vs. informed search

- ▫ common concepts
  - ▫ train set, test set, validation set, S-fold cross-validation
  - ▫ model selection, model comparison
  - ▫ over-fitting, under-fitting, SSE error, RMS error, how to control over-fitting,
  - ▫ penalty term, regularization, shrinkage methods, curse of dimensionality,
- ▫ probability theory
  - ▫ marginal, joint distribution, conditional probability, PDF, CDF, expectation, variance, covariance and their properties
  - ▫ i.i.d, MLE- Maximum Likelihood Estimation, MAPMaximum posterior, log likelihood, Gaussian distribution, Mahalanobis distance,
  - ▫ independent parameters, conjugate prior, kernel density estimator, KNN density estimator, KNN classifier,

- ⊟ Information theory and decision theory
  - ☒ entropy, cross entropy, relative entropy (Kullback-Leibler divergence, KL divergence), mutual information
  - ☐ Naïve Bayes classifier, decision rule, reject option
- ☐ Supervised learning
- ☐ Regression: linear regression, linear basis function model, ridge regression, stochastic gradient descent, weight decay, sparse model, lasso, bias-variance tradeoff,
- ☐ Classification: linear separable, decision regions, decision boundaries, decision surfaces, 1-of-K code (one hot code), Least-squares approach, Fishers linear discriminant, the perceptron algorithm of Rosenblatt, loss function, hinge loss, The Fishers criterion, Generalized Rayleigh quotient, Perceptron criterion, probabilistic generative model, logistic sigmoid function, logit function, softmax function, probabilistic discriminative model, logistic regression
- ☐ Boosting: adaboost, committees, bagging, algorithm,
- ☐ Unsupervised learning
  - ☐ Clustering: partitional clustering, hierarchical clustering, k-means, k-medoids, limitation, MoG, E-step, M-step
  - ☐ Dimensionality Reduction: latent factors, correlation, Pearson correlation coefficient, correlation vs. independence, PCA, eigenface

3 solving problems by search

problem-solving agent

- **goal**
- **goal information** is the 1st step in problem-solving, based on the current situation and the agents performance measure
- **problem formulation** is the process of deciding what actions and states to consider, given a goal
- **search**

- **execution phase**

formulatesearchexecution
type of search

- **uninformed search algorithms**

  algorithms that are given no information about the problem other than its definition. Although some of these algorithms can solve any solvable problem, none of them can do so efficiently

- **informed search algorithms**

The types of Problem-solving by Search

- Deterministic, fully observable

  Agent knows exactly which state it will be in

  solution is a sequence

- non-observable

  Agent may have no idea where it is

  solution (if any) is a sequence

- Nondeterministic and/or partially observable

  percepts provide new information about current state

  solution is a tree or policy

  often interleave search, execution

- Unknown state space

Some assumptions about environment

- **observable**
- **discrete**: the environment is discrete
- **known**: the agent knows which states are reached by each action
- **deterministic**: each action has exactly one outcome

Problem definition

1. **Initial state**

2. **actions**

3. **Transition model**

4. **goal test**: determines whether a given state is a goal state

5. **path cost**: a function that assigns a numeric cost to each path

A solution is an **action sequence**, so search algorithms work by considering various possible action sequences.

Given a search tree, the set of all leaf nodes available for expansion at any given point is called the **frontier(open list)**. **Search strategy**

queues: FIFO queue, LIFO queue (stack), priority queue

Measuring problem-solving performance:

- completeness: Does it always find a solution
- optimality: How long does it take?
- time complexity
- space complexity

uninformd search: Breadth-first search, Depth-first search

Strategies that know whether one non-goal state is more promising than another are called **informed search** or **heuristic search** strategies

3.1 uninformed search

Uniform-cost search: Instead of expanding the shallowest node, uniform-cost search expands the node n with the lowest path cost g(n). This is done by storing the frontier as a priority queue ordered by g(n)

DFS stack LIFO

Depth-limited search:

Iterative deepening depth-first search: for depth = 0 to ∞ do

3.2 Informed search strategies

best-first search

- Best-first search is an instance of the general TREE-SEARCH or GRAPH-SEARCH algorithm in which a node is selected for expansion based on an **evaluation function** f(n)

- The evaluation function is construed as a cost estimate, so the node with the **lowest evaluation** is expanded first

evaluation function *f*

- Most best-first algorithms include as a component of f a **heuristic function**, denoted h(n): *estimated cost of the cheapest path from the state at node n to a goal state*
- For now, we consider h(n) to be **arbitrary, nonnegative, problem-specific** functions, with one constraint: if n is a goal node, then h(n)=0
- **Greedy best-first search** f(n) = h(n)

A* search: f(n)=g(n)+h(n), h(n) the cost to get from the node to the goal

Conditions for optimality: **Admissibility** and Consistency

- f(n) = g(n) + h(n)
- g(n) is the actual cost to reach n along the current path
- h(n) is an **admissible heuristic** function: it never overestimates the cost to reach the goal
- So, f(n) never overestimates the true cost of a solution along the current path through n

Conditions for optimality: Admissibility and **Consistency**

- h(n) c(n, a, n') + h(n')
- for every node n and every successor n of n generated by any action a, the estimated cost of reaching the goal from n is no greater than the step cost of getting to n plus the estimated cost of reaching the goal from n

Properties of A* search

- The tree-search version of A* is optimal if h(n) is admissible
- The graph-search version of A* is optimal if h(n) is consistent

4 Adversarial search

4.1 Minimax search

4.2 evaluation function

4.3 Alpha-Beta Pruning Search

4.4 Monte-Carlo Tree Search

5 Inference and Reasoning

5.1 Propositional logic

5.2 Predicate logic

5.3 First Order Inductive Learner

knowledge graph: node = entity, edge = relation. triplet (head entity, relation, tail entity)

6 Statistical learning and modeling

6.1 Machine Learning: the concept

6.1.1 Example and concept

Supervised learning problems applications in which the **training data** comprises examples of the input vectors along with their corresponding **target vectors** are known

classification and regression

Unsupervised learning problems the training data consists of a set of input vectors X **without any corresponding target values**

density estimation, clustering, hidden markov models

Reinforcement learning problem finding suitable actions to take in a given situation in order to maximize a reward. Here the learning algorithm is not given examples of optimal outputs, in contrast to supervised learning, but must instead discover them by a process of trial and error. A general feature of reinforcement learning is the trade-off between exploration and exploitation

types of machine learning

- supervised learning
  - classification: the output is categorical or nominal variable
  - regression: the output is read-valued variable
- unsupervised learning
- semi-supervised learning
- reinforcement learning
- deep learning

6.1.2 supervised learning: important concepts

- Data: labeled instances $< \boldsymbol{x}_i, \boldsymbol{y} >$
- features: attribute-value pairs which characterize each $\boldsymbol{x}$
- learning a discrete function: **classification**
- learning a continuous function: **regression**

Classification - A two-step process

- **model construction**
- **model usage**

regression

- Example: price of a used car
  $\boldsymbol{x}$: car attributes. $\boldsymbol{y} = g(\boldsymbol{x} \mid \boldsymbol{\theta})$: price. g: model. θ parameter set.

6.2 example: polynomial curve fitting

cross validation

SSE error(sum-of-square) $E(\boldsymbol{w}) = \frac{1}{2} \sum_{n=1}^{N} \{y(x_n, \boldsymbol{w}) - t_n\}^2$

RMS(root-mean-square) error $E_{RMS} = \sqrt{2E(\boldsymbol{w}^*)/N}$

How to control over-fitting

1. more train data

2. regularization
3. bayesian approach
4. cross-validation

curse of dimensionality

- Extend polynomial curve fitting approach to deal with input spaces having several variables. If we have D input variables, then a general polynomial with coefficients up to order 3 would take the form:

$$y(\boldsymbol{x}, \boldsymbol{w}) = w_0 + \sum_{i=1}^{D} w_i x_i + \sum_{i=1}^{D}\sum_{j=1}^{D} w_{ij} x_i x_j + \sum_{i=1}^{D}\sum_{j=1}^{D}\sum_{k=1}^{D} w_{ijk} x_i x_j x_k$$

6.3 probability theory review and notation

rules of probability

- **sum rule** $p(X) = \sum_{Y} p(X, Y)$
- **product rule** $p(X, Y) = p(Y|X)p(X)$

Bayes' Theorem: $p(Y|X) = \frac{p(X|Y)p(Y)}{p(X)}$. Using sum rule $p(X) = \sum_{Y} p(X|Y)p(Y)$

probability densities.

$$p(x \in (a, b)) = \int_a^b p(x)dx$$
$$P(z) = \int_{-\infty}^{z} p(x)dx$$
$$\int_{-\infty}^{\infty} p(x)dx = 1 \quad p(x) \leqslant 0$$

$p(x)$ must satisfy two conditions

$$p(x) \leqslant 0$$
$$\int_{-\infty}^{\infty} p(x)dx = 1$$

expectation $\mathbb{E}[f] = \begin{cases} \sum_x p(x)f(x) & \text{discrete variables} \\ \int p(x)f(x)dx & \text{continuous variables} \end{cases}$. In either cases, $\mathbb{E}[f] \approx \frac{1}{N}\sum_{n=1}^{N} f(x_n)$. **conditional expectation**: $\mathbb{E}_x[f|y] = \sum_x p(x|y)f(x)$.

The **variance** of $f(x)$ is

$$\begin{aligned} var[f] &= \mathbb{E}[(f(x) - \mathbb{E}[f(x)])^2] \\ &= \mathbb{E}[f(x)^2 - 2f(x)\mathbb{E}[f(x)] + \mathbb{E}[f(x)]^2] \\ &= \mathbb{E}[f(x)^2] - \mathbb{E}[f(x)]^2 \end{aligned}$$

The **covariance** is

$$\begin{aligned} cov[x, y] &= \mathbb{E}_{x,y}[(x - \mathbb{E}[x])(y - \mathbb{E}[y])] \\ &= \mathbb{E}_{x,y}[xy] - \mathbb{E}[x]\mathbb{E}[y] \end{aligned}$$

$$\mathbb{V}[X] = \sigma_X^2 = \mathbb{E}[(X - \mathbb{E}[X])^2] = \mathbb{E}[X^2] - \mathbb{E}[X]^2$$

$$\mathbb{V}[\sum_{i=1}^{n} X_i] = \sum_{i=1}^{n} \mathbb{V}[X_i] + \sum_{i \neq j} \mathrm{Cov}[X_i, X_j]$$

$$\begin{aligned} &\mathrm{Cov}[X, X] = \mathbb{V}[X] \\ &\mathrm{Cov}[aX, bY] = ab\mathrm{Cov}[X, Y] \\ &\mathrm{Cov}[X + a, Y + b] = \mathrm{Cov}[X, Y] \end{aligned}$$

the variance of the sum of two independent random variables is the sum of variance. Given

X	probability
x_1	p_1
$\dots$	$\dots$
x_n	p_n

Y	probability
y_1	q_1
$\dots$	$\dots$
y_m	q_m

$$var(X+Y) = var(X) + var(Y)$$

In case of two vectors of random variables $\boldsymbol{x}$ and $\boldsymbol{y}$, the covariance is a matrix

$$\begin{aligned} cov[\boldsymbol{x}, \boldsymbol{y}] &= \mathbb{E}_{\boldsymbol{x},\boldsymbol{y}}[(\boldsymbol{x} - \mathbb{E}[\boldsymbol{x}])(\boldsymbol{y}^T - \mathbb{E}[\boldsymbol{y}^T])] \\ &= \mathbb{E}_{\boldsymbol{x},\boldsymbol{y}}[\boldsymbol{x}\boldsymbol{y}^T] - \mathbb{E}[\boldsymbol{x}]\mathbb{E}[\boldsymbol{y}^T] \end{aligned}$$

Bayesian probabilities: $P(A|B) = \frac{P(B|A)P(A)}{P(B)}$, $p(\mathcal{D}) = \int p(\mathcal{D}|\boldsymbol{w})p(\boldsymbol{w})\mathrm{d}\boldsymbol{w}$. For a data set $\mathcal{D} = \{t_1, \ldots, t_n\}$ and assumption w, $p(w|\mathcal{D}) = \frac{p(\mathcal{D}|w)p(w)}{p(\mathcal{D})}$. $p(w)$ is **prior probability**, $p(\mathcal{D}|w)$ is **likelihood** (the probability $\mathcal{D}$ happens). Hence

$$\text{posterior} \propto \text{likelihood} \times \text{prior}$$

Gaussian distribution.

$$\mathcal{N}(x|\mu, \sigma^2) = \frac{1}{(2\pi\sigma^2)^{1/2}} \exp\left\{-\frac{1}{2\sigma^2}(x-\mu)^2\right\}$$

μ is called **mean**, σ^2 is called **variance**, σ **standard deviation**, $\beta = 1/\sigma^2$ **precision**

$$\begin{aligned} \mathbb{E}[x] &= \int_{-\infty}^{\infty} \mathcal{N}(x|\mu, \sigma^2) x dx = \mu \\ \mathbb{E}[x^2] &= \int_{-\infty}^{\infty} \mathcal{N}(x|\mu, \sigma^2) x^2 dx = \mu^2 + \sigma^2 \\ var[x] &= \mathbb{E}[x^2] - \mathbb{E}[x]^2 = \sigma^2 \end{aligned}$$

For D-dimensional vector $\boldsymbol{x}$ of continuous variables

$$\mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{D/2}} \frac{1}{|\boldsymbol{\Sigma}|^{1/2}} \exp\left\{-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\boldsymbol{x}-\boldsymbol{\mu})\right\}$$

To determine values for the unknown parameters given μ and σ^2 by maximizing the likelihood function. Use log.

$$\begin{aligned} P(\boldsymbol{X}|\mu, \sigma^2) &= \prod_{n=1}^{N} \mathcal{N}(x_n|\mu, \sigma^2) \\ \Rightarrow \ln P(\boldsymbol{X}|\mu, \sigma^2) &= -\frac{1}{2\sigma^2} \sum_{n=1}^{N} (x_n - \mu)^2 - \frac{N}{2} \ln \sigma^2 - \frac{N}{2} \ln(2\pi) \end{aligned}$$

Hence $\mu_{ML} = \frac{1}{N}\sum_{n=1}^{N} x_n$, $\sigma^2_{ML} = \frac{1}{N}\sum_{n=1}^{N}(x_n - \mu_{ML})^2$ by partial derivative. $\mathbb{E}[\sigma^2_{ML}] = (\frac{N-1}{N})\sigma^2$

Maximum likelihood estimator for mean is unbiased, that is, $\mathbb{E}(\mu_{ML}) = \mu$. Maximum likelihood estimator for variance is biased. $\mathbb{E}(\sigma^2_{ML}) = \mathbb{E}(x^2) - \mathbb{E}(\mu^2_{ML}) = \frac{N-1}{N}\sigma^2_x$

6.4 information theory

entropy: measuring uncertainty of a random variable X. $H(X) = H(p) = -\sum_{x\in\Omega} p(x)\log p(x)$ where Ω is all possible values and define $0\log 0 = 0, \log = \log_2$

$H(X) = \sum_{x\in\Omega} p(x)\log_2 \frac{1}{p(x)} = E(\log_2 \frac{1}{p(x)})$. And "information of x"="#bits to code x"=$-\log p(x)$

Kullback-Leibler divergence: comparing two distributions $D_{KL}(p||q) = H(p,q) - H(p) = -\int p(\boldsymbol{x}) \ln\left\{\frac{q(\boldsymbol{x})}{p(\boldsymbol{x})} d\boldsymbol{x}\right.$

`https://www.youtube.com/watch?v=ErfnhcEV1O8`

mutual information $I[\boldsymbol{x},\boldsymbol{y}] = \mathrm{KL}(p(\boldsymbol{x},\boldsymbol{y})||p(\boldsymbol{x})p(\boldsymbol{y})) = H(\boldsymbol{y}) - H[\boldsymbol{y}|\boldsymbol{x}]$

6.5 The gaussian distribution

$$\begin{aligned}\Delta^2 &= (x-\mu)^T\Sigma^{-1}(x-\mu)\\ &= (x-\mu)^T U\Lambda^{-1}U^T(x-\mu)\\ &= (U^T(x-\mu))^T\Lambda^{-1}(U^T(x-\mu)) = y^T\Lambda^{-1}y\end{aligned}$$

$\Sigma u_i = \lambda_i u_i$ where $i = i,\ldots,D$.

$$\Sigma U = \Sigma(u_1,\ldots,u_D) = (u_1,\ldots,u_D)\begin{pmatrix}\lambda_1 & \ldots & 0\\ \vdots & \ddots & \vdots\\ 0 & \ldots & \lambda_D\end{pmatrix} = U\Lambda$$

$\forall i,j \in \{1,\ldots,D\}$,

$$u_i^T u_j = I_{ij} = \begin{cases}1 & \text{if } i = j\\ 0\end{cases}$$

$$U^T U = I$$

So U is orthogonal, $\Sigma U U^T = U\Lambda U^T = \sum_{i=1}^{D} \lambda_i u_i u_i^T$, and $\Sigma^T = U\Lambda^{-1}U^T$

$$\Delta^2 = \boldsymbol{y}^T \Lambda^{-1} \boldsymbol{y} \xrightarrow{y_i = \boldsymbol{u}_i^T(\boldsymbol{x}-\boldsymbol{\mu})} \sum_{i=1}^{D} \frac{y_i^2}{\lambda_i}$$

GIven a square matrix $A \in \mathbb{R}^{n\times n}, x \in \mathbb{R}^n$, $x^T Ax$ is called a **quadratic form**

$$x^T Ax = \sum_{i=1}^{n} x_i (Ax)_{i=} \sum_{i=1}^{n} x_i (\sum_{j=1}^{n} A_{ij} x_j) = \sum_{i=1}^{n}\sum_{j=1}^{n} A_{ij} x_i x_j$$

$$x^T Ax = (x^T Ax)^T = x^T (1/2A + 1/2A^T)x$$

6.6 model selection

cross-validation

split training data into **training set** and **validation set**. Train different models on training set and choose model with minimum error on validation set.

6.7 decision theory

Suppose we have an input vector $\boldsymbol{x}$ together with a corresponding vector $\boldsymbol{t}$ of target variables and our goal is to predict $\boldsymbol{t}$ given new value for $\boldsymbol{x}$. The joint probability distribution $p(\boldsymbol{x}, \boldsymbol{t})$ provides a complete summary of the uncertainty with these variables

7 Statistical learning and modeling - Supervised learning

7.1 Basic concepts

- **Linearly separable**
  - decision regions:
    input space is divided into several regions
  - decision boundaries:
    - under linear models, it's a linear function
    - (D-1)-dimensional hyper-plane within the D-dimensional input space

- **representation of class labels**
  - Two classes K = 2
  - K classes
    - 1-of-K coding scheme $\boldsymbol{t} = (0, 0, 1, 0, 0)^T$
  - Predict discrete class labels
    - linear model prediction $y(\boldsymbol{x}) = \boldsymbol{w}^T\boldsymbol{x} + w_0$ w: weight vector, w_0 bias/threshold
    - nonlinear function $f(.) : R \to (0, 1)$
    - generalized linear models $y(\boldsymbol{x}) = f(\boldsymbol{w}^T\boldsymbol{x} + w_0)$ f:activation function
    - dicision surface $y(\boldsymbol{x}) = \text{constant} \to \boldsymbol{w}^T\boldsymbol{x} + w_0 = \text{constant}$
- **Three classification approaches**
  - discriminant function
    - least squares approach
    - fisher's linear discriminant
    - the perceptron algorithm of rosenblatt
  - use discriminant functions directly and don't compute probabilities

    Given discriminant functions $f_1(\boldsymbol{x}), \ldots, f_K(\boldsymbol{x})$. Classify $\boldsymbol{x}$ as class $\mathcal{C}_k$ iff $f_k(\boldsymbol{x}) > f_j(\boldsymbol{x}), \forall j \neq k$
    - **least-squares approach**: making the model predictions as close as possible to a set of target values
    - **fisher's linear discriminant**: maximum class separation in the ouput space
    - **the perceptron algorithm of rosenblatt**
  - generative approach
    - model the class-conditional densities and the class priors
    - compute posterior probabilities through Bayes's theorem

$$\underbrace{p(\mathcal{C}_k|\boldsymbol{x})}_{\text{posterior for class}} = \frac{\overbrace{p(\boldsymbol{x}|\mathcal{C}_k)}^{\text{class conditional density}} \overbrace{p(\mathcal{C}_k)}^{\text{class prior}}}{p(\boldsymbol{x})} = \frac{p(\boldsymbol{x}|\mathcal{C}_k)p(\mathcal{C}_k)}{\sum_j p(\boldsymbol{x}|\mathcal{C}_j)p(\mathcal{C}_j)}$$

7.2 discriminant functions

7.2.1 Two classes

- Linear discriminant function $y(\boldsymbol{x}) = \boldsymbol{w}^T\boldsymbol{x} + w_0$
  - Dicision surface $\Omega : y(\boldsymbol{x}) = 0$
  - the normal distant from the origin to the dicision surface $\frac{\boldsymbol{w}^T\boldsymbol{x}}{\|\boldsymbol{w}\|} = -\frac{w_0}{\|\boldsymbol{w}\|}$
  - if x_A, x_B lie on the decision surface $y(\boldsymbol{x}_A) = y(\boldsymbol{x}_B) = 0$, then $\boldsymbol{w}^T(\boldsymbol{x}_A - \boldsymbol{x}_B) = 0$. hence w is orthogonal to every vector lying within . $\frac{\boldsymbol{w}}{\|\boldsymbol{w}\|}$ is the normal vector of
  - $\boldsymbol{x} = \boldsymbol{x}_\perp + r\frac{\boldsymbol{w}}{\|\boldsymbol{w}\|}$ hence $r = \frac{y(\boldsymbol{x})}{\|\boldsymbol{w}\|}$. $y(\boldsymbol{x}_\perp) = 0 \rightarrow \boldsymbol{w}^T\boldsymbol{x} = -w_0 + r\frac{\boldsymbol{w}^T\boldsymbol{w}}{\|\boldsymbol{w}\|}$
  - $\tilde{\boldsymbol{w}} = (w_0, \boldsymbol{w}), \tilde{\boldsymbol{x}} = (x_0, \boldsymbol{x}), y(\boldsymbol{x}) = \tilde{\boldsymbol{w}}^T\tilde{\boldsymbol{x}}$

7.2.2 K-class

- One-versus-the-rest classifier K - 1 classifiers each of which solves a two-class problem
- One-versus-one classifier K(K-1)/2 binary discriminant functions
- single K-class discriminant comprising K linear functions $y_k(\boldsymbol{x}) = \boldsymbol{w}_k^T\boldsymbol{x} + w_{k_0}$
  - assigning a point x to class $\mathcal{C}_k$ if $y_k(\boldsymbol{x} > y_j(\boldsymbol{x}))$ for all jk
  - dicision boundary between class $\mathcal{C}_k, \mathcal{C}_j$ is given $y_k(\boldsymbol{x}) = y_j(\boldsymbol{x}) \rightarrow (\boldsymbol{w}_k - \boldsymbol{w}_j)^T\boldsymbol{x} + (w_{k_0} - w_{j_0}) = 0$
  - $\mathcal{R}_k$ is singly connected convex
  - $\hat{\boldsymbol{x}} = \lambda\boldsymbol{x}_A + (1-\lambda)\boldsymbol{x}_B$ where $0 \leqslant \lambda \leqslant 1$, $y_k(\hat{\boldsymbol{x}}) = \lambda y_k(\boldsymbol{x}_A) + (1-\lambda)y_k(\boldsymbol{x}_B)$ and hence $\hat{x}$ also lies inside $\mathcal{R}_k$

7.2.3 Learning the parameters of linear discriminant functions

1. Linear basis function models **linear regression**: $y(\boldsymbol{x}, \boldsymbol{w}) = w_0 + w_1x_1 + \cdots + w_Dx_D = \boldsymbol{w}^T\boldsymbol{x}$.

   For nonlinear functions ϕ_j, $y(\boldsymbol{x}, \boldsymbol{w}) = w_0 + \sum_{j=1}^{M-1} w_j\phi_j(\boldsymbol{x}) = \boldsymbol{w}^T\boldsymbol{\phi}(\boldsymbol{x})$ where $\phi_j(\boldsymbol{x})$ are **basis functions**

2. **parameter optimization via maximum likelihood**

Assume target variable t is given by a deterministic function $y(\boldsymbol{x}, \boldsymbol{w})$ with additive Gaussian noice so that $t = y(\boldsymbol{x}, \boldsymbol{w}) + \epsilon$ where ϵ is a zero mean Gaussian random variable with precision β, hence we can write

$$p(t|\boldsymbol{x}, \boldsymbol{w}, \beta) = \mathcal{N}(t|y(\boldsymbol{x}, \boldsymbol{w}), \beta^{-1})$$

and $\mathbb{E}(t|\boldsymbol{x}) = \int tp(t|\boldsymbol{x})dt = y(\boldsymbol{x}, \boldsymbol{w})$

For data set $\boldsymbol{X} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}, \boldsymbol{t} = (t_1, \ldots, t_n)^T$, $p(\boldsymbol{t}|\boldsymbol{X}, \boldsymbol{w}, \beta) = \prod_{n=1}^{N} \mathcal{N}(t_n|\boldsymbol{w}^T\boldsymbol{\phi}(\boldsymbol{x}_n), \beta^{-1})$

$$\ln p(\boldsymbol{t}|\boldsymbol{w}, \beta) = \sum_{n=1}^{N} \ln \mathcal{N}(t_n|\boldsymbol{w}^T\boldsymbol{\phi}(\boldsymbol{x}_n), \beta^{-1}) = \frac{N}{2}\ln\beta - \frac{N}{2}\ln(2\pi) - \beta E_D(\boldsymbol{w})$$

$E_D(\boldsymbol{w}) = \frac{1}{2}\sum_{n=1}^{N}\left\{t_n - \boldsymbol{w}^T\boldsymbol{\phi}(\boldsymbol{x}_n)\right\}^2 = \frac{1}{2}\|t - \Phi\boldsymbol{w}\|$ is sum-of-squares error function

solve $\boldsymbol{w}$ by maximum likelihood.

$$\nabla \ln p(\boldsymbol{t}|\boldsymbol{w}, \beta) = \sum_{n=1}^{N}\left\{t_n - \boldsymbol{w}^T\boldsymbol{\phi}(\boldsymbol{x}_n)\right\}\phi(\boldsymbol{x}_n)^T$$

$$0 = \sum_{n=1}^{N} t_n\boldsymbol{\phi}(\boldsymbol{x}_n)^T - \boldsymbol{w}^T(\sum_{n=1}^{N}\boldsymbol{\phi}(\boldsymbol{x}_n)\boldsymbol{\phi}(\boldsymbol{x}_n)^T)$$

Hence we get

$$\boldsymbol{w}_{ML} = (\boldsymbol{\Phi}^T\boldsymbol{\Phi})^{-1}\boldsymbol{\Phi}^T\boldsymbol{t}$$

Φ is **design matrix**.

$$\Phi = \begin{pmatrix} \phi_0(\boldsymbol{x}_1) & \phi_1(\boldsymbol{x}_1) & \ldots & \phi_{M-1}(\boldsymbol{x}_1) \\ \phi_0(\boldsymbol{x}_2) & \phi_1(\boldsymbol{x}_2) & \ldots & \phi_{M-1}(\boldsymbol{x}_2) \\ \vdots & \vdots & \ddots & \vdots \\ \phi_0(\boldsymbol{x}_N) & \phi_1(\boldsymbol{x}_N) & \ldots & \phi_{M-1}(\boldsymbol{x}_N) \end{pmatrix}$$

For bias parameter w_0. $E_D(\boldsymbol{w}) = \frac{1}{2}\sum_{n=1}^{N}\{t_n - w_0 - \sum_{j=1}^{M-1} w_j\phi_j(\boldsymbol{x}_n)\}^2$.

Hence $w_0 = \bar{t} - \sum_{j=1}^{M-1} w_j\bar{\phi_j}$, $\bar{t} = \frac{1}{N}\sum_{n=1}^{N} t_n$, $\bar{\phi_j} = \frac{1}{N}\sum_{n=1}^{N}\phi_j(\boldsymbol{x}_n)$.

$fracN2\beta = E_D(\boldsymbol{w}).\ \frac{1}{\beta_{ML}} = \frac{1}{N}\sum_{n=1}^{N}\left\{t_n - \boldsymbol{w}_{ML}^T\boldsymbol{\phi}(\boldsymbol{x}_n)\right\}^2$

3. Least-squares approach

   - Problem
     - Each class $\mathcal{C}_k$ is described by its own linear model $y_k(\boldsymbol{x}) = \boldsymbol{w}_k^T\boldsymbol{x} + w_{k0}$
     - group together: $y(\boldsymbol{x}) = \widetilde{\boldsymbol{W}}^T\tilde{\boldsymbol{x}},\ \tilde{\boldsymbol{w}}_k = (w_{k0}, \boldsymbol{w}_k^T)^T,\ \tilde{\boldsymbol{x}} = (1, \boldsymbol{x}^T)^T$
   - Learning
     - minimizing SSE function sum-of-squares $SSE = \sum_{i=1}^{n}(y_i - f(x_i))^2$ $E_D(\widetilde{\boldsymbol{W}}) = 1/2\mathrm{Tr}\{(\widetilde{\boldsymbol{X}}\widetilde{\boldsymbol{W}} - \boldsymbol{T})^T(\widetilde{\boldsymbol{X}}\widetilde{\boldsymbol{W}} - \boldsymbol{T})\}$
       $\widetilde{\boldsymbol{W}} = (\widetilde{\boldsymbol{X}}^T\widetilde{\boldsymbol{X}})^{-1}\widetilde{\boldsymbol{X}}^T\boldsymbol{T}$

4. fisher's linear discriminant

   from the view of dimensionality reduction $y \geqslant -w_0$ as class $\mathcal{C}_1$

   $m_1 = \frac{1}{N_1}\sum_{n\in\mathcal{C}_1} x_n, m_2 = \frac{1}{N_2}\sum_{n\in\mathcal{C}_2} x_n \xrightarrow{y=\boldsymbol{w}^T\boldsymbol{x}} m_2 - m_1 = \boldsymbol{w}^T(\boldsymbol{m}_2 - \boldsymbol{m}_1)$

5. the perceptron algorithm of rosenblatt

7.3 probalibilistic generative models

A probabilistic view of classification from simple assumptions about the distribution of the data

$$\begin{aligned} p(\mathcal{C}_1|\boldsymbol{x}) &= \frac{p(\boldsymbol{x}|\mathcal{C}_1)p(\mathcal{C}_1)}{p(\boldsymbol{x}|\mathcal{C}_1)p(\mathcal{C}_1) + p(\boldsymbol{x}|\mathcal{C}_2)p(\mathcal{C}_2)} \\ &= \frac{1}{1+\exp(-a)} = \sigma(a) \end{aligned}$$

where

$$a = \ln\frac{p(\boldsymbol{x}|\mathcal{C}_1)p(\mathcal{C}_1)}{p(\boldsymbol{x}|\mathcal{C}_2)p(\mathcal{C}_2)}$$

and $\sigma(a)$ is the **logistic sigmoid** function defined by

$$\sigma(a) = \frac{1}{1 + \exp(-a)}$$

and $\sigma(-a) = 1 - \sigma(a)$, its inverse is **logit** function

$$a = \ln(\frac{\sigma}{1-\sigma})$$

For case of $K > 2$ classes, we have the following **multi-class generalization**

$$p(\mathcal{C}_k|\boldsymbol{x}) = \frac{p(\boldsymbol{x}|\mathcal{C}_k)p(\mathcal{C}_k)}{\sum_j p(\boldsymbol{x}|\mathcal{C}_j)p(\mathcal{C}_j)} = \frac{\exp(a_k)}{\sum_j \exp(a_j)}, a_k = \ln\left[p(\boldsymbol{x}|\mathcal{C}_k)p(\mathcal{C}_k)\right]$$

The **normalized exponential** is known as the **softmax function** as it represents a *smoothed version of the max function*

$$\text{if } a_k \ll a_j, \forall j \neq k, \text{then } p(\mathcal{C}_k|\boldsymbol{x}) \approx 1, p(\mathcal{C}_j|\boldsymbol{x}) \approx 0$$

For **continuous inputs**, assume

$$p(\boldsymbol{x}|\mathcal{C}_k) = \frac{1}{(2\pi)^{D/2}} \frac{1}{|\boldsymbol{\Sigma}|^{1/2}} \exp\left\{-\frac{1}{2}(\boldsymbol{x} - \boldsymbol{\mu}_k)^T \boldsymbol{\Sigma}^{-1} (\boldsymbol{x} - \boldsymbol{\mu}_k)\right\}$$

1. 2 classes

$$\begin{aligned} p(\mathcal{C}_1|\boldsymbol{x}) &= \sigma(\boldsymbol{w}^T\boldsymbol{x} + w_0) \\ \boldsymbol{w} &= \boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2) \\ w_0 &= -\frac{1}{2}\boldsymbol{\mu}_1^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_1 + \frac{1}{2}\boldsymbol{\mu}_2^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_2 + \ln\frac{p(\mathcal{C}_1)}{p(\mathcal{C}_2)} \end{aligned}$$

2. K classes

$$\begin{aligned} a_k(\boldsymbol{x}) &= \boldsymbol{w}_k^T\boldsymbol{x} + w_{k0} \\ \boldsymbol{w}_k &= \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_k \\ w_{k0} &= -\frac{1}{2}\boldsymbol{\mu}_k^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_k + \ln p(\mathcal{C}_k) \end{aligned}$$

7.4 probabilistic discriminative models

7.5 Boosting

Originally designed for classification problems.

Motivation: a procedure that combines the outputs of many "weak" classifiers to produce a strong/accurate classifier

7.5.1 AdaBoost

8 unsupervised learning - clustering em and PCA

8.1 K-means clustering

- Distortion measure $J = \sum_{n=1}^{N} \sum_{k=1}^{K} r_{nk} \|\boldsymbol{x}_n - \boldsymbol{\mu}_k\|^2$

8.2 Mixtures of Gaussians

- Definition:

$$p(\boldsymbol{x}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \quad \sum_{k=1}^{k} \pi_k = 1 \quad 0 \leqslant \pi_k \leqslant 1$$

- introduce a K-dimensional binary random variable $\boldsymbol{z} = (z_1, \ldots, z_k)^T$

$$z_k \in \{0, 1\} \quad \sum_k z_k = 1 \quad p(z_k = 1) = \pi_k$$

Hence $p(\boldsymbol{z}) = \prod_{k=1}^{K} \pi_k^{z_k}$, $\boldsymbol{z}$ is **latent variable** (inferred from other observed variables)

If $p(\boldsymbol{x}|z_k = 1) = \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma})$, then $p(\boldsymbol{x}|\boldsymbol{z}) = \prod_{k=1}^{K} \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)^{z_k}$

- **equivalent formulation** of the Gaussian mixture.

$$\boxed{p(\boldsymbol{x}) = \sum_{\boldsymbol{z}} p(\boldsymbol{x}|\boldsymbol{z})p(\boldsymbol{z}) = \sum_{\boldsymbol{z}} \prod_{k=1}^{K} \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)^{z_k}}$$

$$= \sum_{j=1}^{K} \prod_{k=1}^{K} \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)^{I_{kj}} \quad I_{kj} = \begin{cases} 1 & \text{if } k = j \\ 0 & \text{otherwise} \end{cases}$$

$$= \sum_{j=1}^{K} \pi_j \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)$$

responsibility:

$$\gamma(z_k) = p(z_k = 1|\boldsymbol{x}) = \frac{p(z_k = 1)p(\boldsymbol{x}|z_k = 1)}{\Sigma_{j=1}^{K} p(z_j = 1)p(\boldsymbol{x}|z_j = 1)} = \frac{\pi_k \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma})}{\Sigma_{j=1}^{K} \pi_j \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_j \boldsymbol{\Sigma}_j)}$$

Expectation-Maximization algorithm for GMM. $p(\boldsymbol{X}|) = \prod p(\boldsymbol{x})$

$\ln p(\boldsymbol{X}|\pi, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \Sigma_{n=1}^{N} \ln \left\{ \Sigma_{k=1}^{K} \pi_k \mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \right\}$

1. E step

$$\gamma(z_{nk}) = \frac{\pi_k \mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\Sigma_j \pi_j \mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}$$

2. M step

   - solve $\boldsymbol{\mu}_k$

$$\frac{\partial \ln p(\boldsymbol{X}|\pi, \boldsymbol{\mu}, \boldsymbol{\Sigma})}{\partial \boldsymbol{\mu}_k} = 0$$

$$0 = -\frac{\pi_k \mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\Sigma_j \pi_j \mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)} \boldsymbol{\Sigma}_k^{-1}(\boldsymbol{x}_n - \boldsymbol{\mu}_k)$$

$$\boldsymbol{\mu}_k = \frac{1}{N_k} \Sigma_{n=1}^{N} \gamma(z_{nk}) \boldsymbol{x}_n$$

$$N_k = \Sigma_{n=1}^{N} \gamma(z_{nk})$$

   - solve $\boldsymbol{\Sigma}_k$

$$\frac{\partial \ln p(\boldsymbol{X}|\pi, \boldsymbol{\mu}, \boldsymbol{\Sigma})}{\partial \boldsymbol{\Sigma}_k} = 0$$

$$\boldsymbol{\Sigma}_k = \frac{1}{N_k} \Sigma_{n=1}^{N} \gamma(z_{nk})(\boldsymbol{x}_n - \boldsymbol{\mu}_k)(\boldsymbol{x}_n - \boldsymbol{\mu}_k)^T$$

EM for Gaussian Mixtures

1. initialize the means $\boldsymbol{\mu}_k$, covariances $\boldsymbol{\Sigma}_k$ and mixing coefficients π_k
2. E step
3. M step
4. evaluate the log likelihood

$$\ln p(\boldsymbol{X}|\pi, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \Sigma_{n=1}^{N} \ln \left\{ \Sigma_{k=1}^{K} \pi_k \mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \right\}$$

and check for convergence of either the parameters or the log likelihood. If the convergence criterion is not satisfied return to step 2

8.3 An alternative view of EM

8.3.1 the general EM algorithm

The log likelihood of a discrete latent variables model

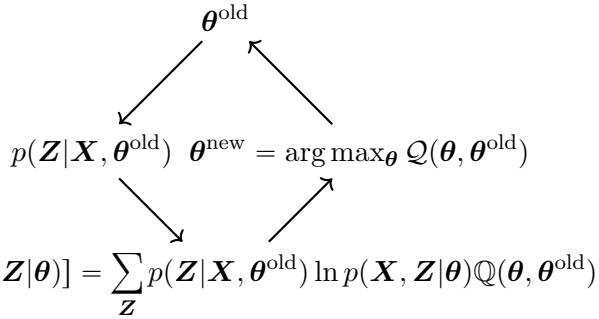
$$\ln p(\boldsymbol{X}|\theta) = \ln \left\{ \Sigma_{\boldsymbol{Z}} p(\boldsymbol{X}, \boldsymbol{Z}|\boldsymbol{\theta}) \right\}$$

the goal of EM algorithm is to find maximum likelihood solution for models having latent variables

For the complete data set $\{\boldsymbol{X}, \boldsymbol{Z}\}$, the likelihood function

$$\ln p(\boldsymbol{X}|\boldsymbol{\theta}) \Longrightarrow \ln p(\boldsymbol{X}, \boldsymbol{Z}|\boldsymbol{\theta})$$

For the incomplete data set $\{\boldsymbol{X}\}$, we adopt the following steps to find



maximum likelihood solution

9 reinforcement learning

10 wef

10.1 wfe

K-means