ML - TOC

---

1 Concept Learning

Given data set D of samples $x_1,x_2,..,x_n$ and their classifications $c(x_1),c(x_2),..,c(x_n)$, find hypothesis h such that for every x_i in D, $h(x_i)=c(x_i)$.

Algorithms

FIND-S
- start with the most specific hypothesis h in H
- for every positive sample x:
    -if any attribute in h causes it to reject sample x,
        change it to next most specific one agreeing with x.

LIST-THEN-ELIMINATE
- start with VS=H
- remove from VS every hypothesis inconsistent with any sample from D.

CANDIDATE-ELIMINATION
- start with the most specific boundary S={<0,0,0...,0>}
- start with the most general boundary G={<?,?,?,...,?>}
- for positive sample x:
  -- remove any hypothesis from G which rejects x.
  -- replace any hypothesis in S which rejects x with its minimal generalizations accepting x
      not more general than all hypotheses in G.
- for negative sample x:
  -- remove any hypothesis from S which accepts x.
  -- replace any hypothesis in G which accepts x with its minimal specifications rejecting x
      not more specific than all hypotheses in S.

Definitions

- hypothesis g is more general than s iff g accepts any instance accepted by s.
- hypothesis h is consistent with sample x iff h(x)=c(x).
- hypothesis h is consistent with samples set D iff h(x)=c(x) for every x from D.
- version space VS(H,D)={h from H| h is consistent with D}.
- inductive bias: assumptions made by the learner required to predict the target output for unobserved samples.

inductive biases used by concept learning algorithms:

CANDIDATE ELIMINATION: the target concept c is in H, bounded by $g>=c>=s$ for $s\in S, g\in G$.

FIND-S: the target concept is in H, accepting only the observed positive samples.

2 Decision Trees

- non leaf nodes test an attribute.
- each branch corresponds to possible attribute value.
- each leaf node provides a classification.

ID3
- if all examples have the same classification c, return root node with label c.
- if no attributes were left for splitting, return root node with most common classification of the examples as the node label.
-otherwise:
    - choose the best attribute *A* for splitting (*A* reduces the entropy the most).
    - for each possible value *v* of *A*:
        - add a new branch, corresponding to *A=v*.
        - repeat the process for this branch with the examples where *A=v* and the remaining attributes.

ID3 Properties

- H is a complete space of finite discrete valued functions.
- maintains a single possible hypothesis during search (incomplete search).
- no backtracking, greedily splits at each step, might fall into local optima.
- bias: prefers shorter trees with higher gain measure closer to the root.

Measures

-Entropy (classification impurity measure):

$$Entropy(S)=\sum_{i=1,\dots,c} -\frac{|S_i|}{|S|}\log_2\frac{|S_i|}{|S|}$$ where S_i is the subset of samples classified with i.

-Gain (reduction of entropy for splitting by attribute *A*):

$$Gain(S,A)=Entropy(S)-\sum_{v\in values(A)}\frac{|S_{A=v}|}{|S|}Entropy(S_{A=v})$$

-Gain Ratio (variation of Gain for multi-valued attribute *A*):

$$GainRatio(S,A)=\frac{Gain(S,A)}{SplitInformation(S,A)}$$

split-information is the entropy with respect to *A* instead of classification.

Pruning

- Discard tree portions that worsen the estimated accuracy/cause overfitting.

Reduced Error Pruning

- split samples set to train and validation data.
- discard any subtree if this improves classification rate over validation data.
- results in the minimal most accurate subtree.

Rules Post Pruning

- grow tree until it overfits training data.
- convert each path from root to any leaf node to corresponding rule.
- generalize rules by pruning their preconditions if it improves their accuracy.
- sort rules by estimated accuracy, using them by this order.
- (estimated accuracy)=(accuracy over training set)-*z**(standard deviation).

3 Perceptrons / SVMs

discriminant function: $f: R^n \to R$
f(x)=0 defines a decision surface dichotomizing R^n into two regions: f(x)<0,f(x)>0.

two category classification:

c_1 if f(x)>0 ; c_2 if f(x)<0

multi-category classification (for k possible categories):

C_t if $\underset{i}{argmax}\{f_i(x)\}=t$ k discriminant functions voting.

C_t if $\underset{i}{argmax}\{f_{ij}(x),-f_{ji}(x)\}=t$ k(k-1)/2 discriminant functions pairwise voting.

linear discriminant function: $g(\vec{x})=\vec{w}^T\cdot\vec{x}+w_0$ g(x)=0 defines an hyperplane.

thresholded perceptron: $o=sign(g(\vec{x}))=sign(\vec{w}^T\cdot\vec{x}+w_0)$

training rule $\Delta w_i=\eta\sum_{d\in D}(t_d-o_d)x_{id}$

-converges into separating hyperplane if:
1) data is linearly separable
2) η is small enough

unthresholded perceptron: $o=g(\vec{x})=\vec{w}^T\cdot\vec{x}+w_0$

training rule: $\Delta w_i=\eta\sum_{d\in D}(t_d-o_d)x_{id}$

-converges into global minimum of the MSE $E(\vec{w})=\frac{1}{2}\sum_{d\in D}(t_d-o_d)^2$ if η is small enough.

generalized linear discriminant function: $g(\vec{x})=\vec{w}^T\cdot\phi(\vec{x})$
- where ϕ is some non-linear mapping $\phi: R^d \to R^{\hat{d}}$.
- g(x) will be linear in the dimension $\hat{d}$.

dual perceptron: $\vec{w}^T\cdot\vec{x}+w_0=(\sum_j a_j t_j \vec{x}_j)\cdot\vec{x}+w_0$

training rule: $a_i=a_i+\eta, w_0=w_0+\eta$ if $t_i(\sum_j a_j t_j(\vec{x}_j\cdot\vec{x}_i)+w_0)<0$

inner product $(\vec{x}_j\cdot\vec{x}_i)$ can be replaced by kernel $K(\vec{x}_j,\vec{x}_i)$ in attempt to linearly separate the problem space in a higher dimension.

Support Vector Machines

the best separating hyperplane in the mapped/original space is the one which maximizes the margin *b* (positive distance of hyperplane from closest point):

maximize *b*

subject to: $\frac{y_i(\vec{a}^T\phi(x_i)+a_0)}{\|\vec{a}\|}\geq b$

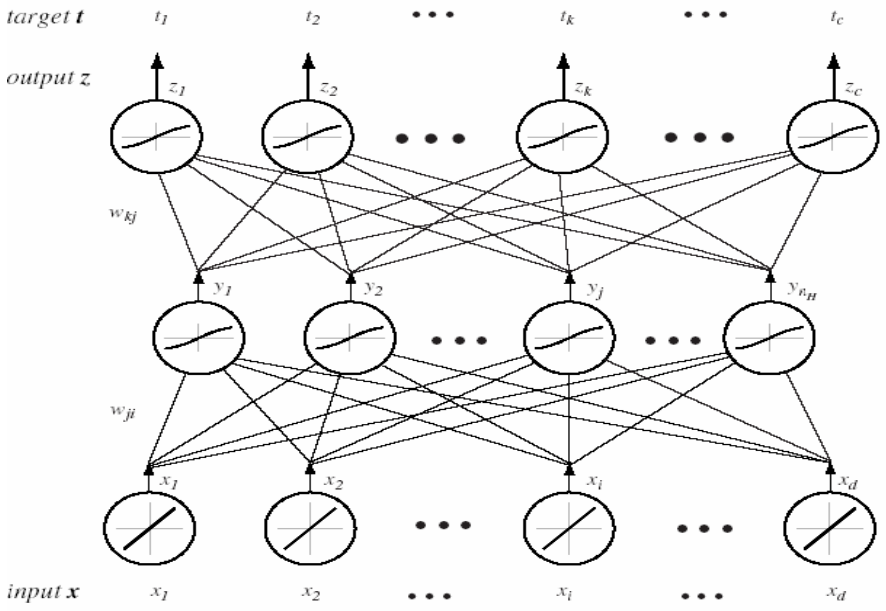
constraining $b=\frac{1}{\|\vec{a}\|}$ for a single solution would lead to:

minimize $\frac{1}{2}\|\vec{a}\|^2$

subject to: $y_i(\vec{a}^T\phi(x_i)+a_0)-1\geq 0$

this can be solved using Lagrange multipliers.

4 ANN



feed-forward back-propagation

layer	net activation	net output	sensitivity (error)	learning rule
input unit i	x_i	x_i	-	-
hidden unit j	$net_j=\Sigma_i x_i w_{ji}$	$y_j=f(net_j)$	$\delta_j=f'(net_j)\cdot\sum_k \delta_k w_{kj}$	$\Delta w_{ji}=\eta\,\delta_j x_i$
output unit k	$net_k=\Sigma_j y_j w_{kj}$	$z_k=f(net_k)$	$\delta_k=(t_k-z_k)f'(net_k)$	$\Delta w_{kj}=\eta\,\delta_k y_j$

- the training rule minimizes the MSE for every given pattern: $J=\frac{1}{2}\|\vec{t}-\vec{z}\|^2$

- generally, weights are updated in a gradient descent manner: $\Delta w=-\eta\frac{\partial J}{\partial w}=\eta\,\delta\,x$

- might fall into local minima (since error surface is not parabolic it this case).

-- try to overcome by stochastic learning (new error surface for every sample).

-- try to overcome using momentum $\Delta w_{(t+1)}=\eta\,\delta\,x+\alpha\Delta w_{(t)}$ to keep gradient descending.

-- try to overcome by training multiple networks with different initial weights.

- sigmoid activation function $\sigma(net)=\frac{1}{1+e^{-net}}$, $\sigma'(x)=\sigma(x)(1-\sigma(x))$, $0<\sigma(x)<1$

- representational power:

-- a single unit will always produce a linear decision boundary.

-- boolean function can be represented by two-layered thresholded network.

-- bounded continuous function can be approximated using one hidden layer.

-- any function can be approximated using two hidden layers.

5 Bayesian Learning

Mean, Variance, Covariance:

$$E[x]=\mu_x=\sum_{v\in vals(x)} v\cdot P(x=v) \qquad E[x]=\mu_x=\int_{-\infty}^{\infty} x\cdot P(x)\,dx$$

$$Var(x)=\sigma_x^2=E[(x-\mu_x)^2]=\sum_{v\in vals(x)} (v-\mu_x)^2\cdot P(x=v) \quad Var[x]=\sigma_x^2=\int_{-\infty}^{\infty} (x-\mu_x)^2\cdot P(x=v)\,dx$$

$$Cov(x,y)=\sigma_{xy}=\sum_{v\in vals(x)}\sum_{u\in vals(y)} (v-\mu_x)(u-\mu_y)P(x=v,y=u) \quad Cov(x,y)=\sigma_{xy}=\int_{-\infty}^{\infty} (x-\mu_x)(y-\mu_y)P(x,y)\,dx$$

-Conditional Probability: P(x|y)=P(x,y)/P(y)

-variables x and y are statistical independent iff P(x,y)=P(x)P(y)

-if x and y are independent, then they are also uncorrelated, i.e. Cov(x,y)=0.

-Law of total Probability: $P(x)=\sum_y P(x,y)=\sum_y P(x|y)P(y)$

Bayes Rule: $\underbrace{P(h|d)}_{posterior}=\underbrace{P(d|h)}_{likelihood}\cdot\underbrace{P(h)}_{prior}/\underbrace{P(d)}_{evidence}$

-Risk(the probability for wrong classification): $R(h_i|d)=\sum_{j\neq i}P(h_j|d)=1-P(h_i|d)$

- MAP hypothesis: $h_{MAP}=\underset{h}{argmax}\,P(d|h)P(h)$

-- if P(h) is assumed to be uniform for every h: $h_{MAP}=h_{ML}=\underset{h}{argmax}\,P(d|h)$

-- log likelihood classifier: $h_{MAP}=\underset{h}{argmax}\ \ln(P(d|h)\cdot P(h))=\underset{h}{argmax}\ \ln(P(d|h))+\ln(P(h))$

-- any consistent learner produces a MAP hypothesis if we assume:
1) uniform priors P(h).
2) error free training set.

Naive Bayesian Classifier: $h_{NB}=\underset{h}{argmax}\,P(a_1,a_2,..,a_n|h)\cdot P(h)=\underset{h}{argmax}\,P(h)\cdot\prod_{i=1}^{n}P(a_i|h)$

-- assumes independence of $a_1,a_2,...,a_n$ for computational efficiency.

-- M-Estimate for likelihoods: $P(a_i|h)=\frac{n_{(a=a_i\wedge t=h)}+mP(a=a_i)}{n_{(t=h)}+m}$

-- Laplace Estimate for likelihoods: $P(a_i|h)=\frac{n_{(a=a_i\wedge t=h)}+1}{n_{(t=h)}+|values(a)|}$

6 Density Estimation

- Normal distribution: $p(x)\approx N(\mu,\sigma^2)=\frac{1}{\sqrt[d]{2\pi}\,\sigma}e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$ $p(\vec{x})\approx N(\vec{\mu},\Sigma)=\frac{1}{\sqrt[d]{2\pi}\cdot|\Sigma|^{1/2}}e^{-\frac{(\vec{x}-\vec{\mu})^T\Sigma^{-1}(\vec{x}-\vec{\mu})}{2}}$

-- CLT: the dist of *n* samples mean selected from some dist $F(\mu,\sigma^2)$ is $N(\mu,\sigma^2/n)$.

-- if $p(x)\approx N(\mu_x,\sigma_x^{\,2})$, $p(y)\approx N(\mu_y,\sigma_y^{\,2})$ then $p(\alpha x+\beta y)\approx N(\alpha\mu_x+\beta\mu_y,(\alpha\sigma_x)^2+(\beta\sigma_y)^2)$.

-- if $p(\vec{x})\approx N(\vec{\mu},\Sigma)$, $\vec{y}=A^T\cdot\vec{x}$ then $p(\vec{y})\approx N(A^T\cdot\vec{\mu},A^T\Sigma A)$.

- Discriminant function: $g_i(\vec{x})=P(\vec{x}|A_i)\cdot P(A_i)=\frac{1}{\sqrt[d]{2\pi}\cdot|\Sigma_i|^{1/2}}e^{-\frac{(\vec{x}-\vec{\mu}_i)^T\Sigma_i^{-1}(\vec{x}-\vec{\mu}_i)}{2}}\cdot P(A_i)$

- Log-Likelihood discriminant:

$$g_i(\vec{x})=\ln(P(\vec{x}|A_i)\cdot P(A_i))=-\frac{1}{2}(\vec{x}-\vec{\mu}_i)^T\Sigma_i^{-1}(\vec{x}-\vec{\mu}_i)-\frac{d}{2}\ln(2\pi)-\frac{1}{2}\ln|\Sigma_i|+\ln(P(A_i))$$

- Covariance Matrix: $\Sigma=E[(\vec{x}-\vec{\mu})^T\cdot(\vec{x}-\vec{\mu})]$
$(\Sigma)_{i,j}=(\Sigma)_{j,i}=\sigma_{ij}$

-- measures the covariance for every pair of components of any sample vector x.
-- the sample components are uncorrelated iff Σ is diagonal.

Cov	Discriminant function	decision boundary
$\Sigma_i=\sigma^2 I$	$g_i(x)=-(x^tx-2\mu_ix+\mu_i^t\mu_i)/2\sigma^2+\ln(P(A_i))$ linear since $x^tx/2\sigma^2$ is same for any i and can be omitted	$x=\frac{1}{2}(\mu_i+\mu_j)-(\sigma/(\mu_i-\mu_j))^2\ln(P(A_i)/P(A_j))(\mu_i-\mu_j)$ d-dimensional hyperplane closer to the class with the lower prior P(A) and orthogonal to the line linking the means.
$\Sigma_i=\Sigma$	$g_i(x)=\frac{1}{2}(x-\mu_i)^T\Sigma^{-1}(x-\mu_i)+\ln(P(A_i))$ square of mahalanobis distance from the mean plus some bias.	$x=\frac{1}{2}(\mu_i+\mu_j)-((\mu_i-\mu_j)^T\Sigma^{-1}(\mu_i-\mu_j))^{-1}\ln(P(A_i)/P(A_j))(\mu_i-\mu_j)$ same as the former case, but not necessarily orthogonal to the line linking the means.
$\Sigma_i\neq\Sigma_j$	general log-likelihood,quadratic.	complex, non-linear.

Parametric techniques:

-Maximum Likelihood Estimation:

given the likelihood of samples drawn independently from set D: $p(D|\theta)=\prod_{\vec{x}\in D}p(\vec{x}|\theta)$

the log likelihood is: $l(\theta)=\ln(p(D|\theta))=\sum_{\vec{x}\in D}\ln(p(\vec{x}|\theta))$

find the dist parameters maximizing the log-likelihood: $\hat{\theta}=\underset{\theta}{argmax}\ l(\theta)$

the gradient at $\hat{\theta}$ should be is 0: $\frac{\partial l(\theta)}{\partial\theta}=\sum_{\vec{x}\in D}\frac{d\ \ln(p(\vec{x}|\theta))}{d\theta}=0$

--for normal dist: $\hat{\theta}=(\hat{\mu},\hat{\Sigma})=(\frac{1}{|D|}\sum_{\vec{x}\in D}\vec{x}\ ,\ \frac{1}{|D|}\sum_{\vec{x}\in D}(\vec{x}-\hat{\mu})^T(\vec{x}-\hat{\mu}))$ such that $\Sigma_{true}=\frac{|D|}{|D|-1}\hat{\Sigma}$

-Bayesian Estimation: $p(x|D)=\int p(x|\theta)\,p(\theta|D)\,d\theta$

Non Parametric techniques (no assumptions on true distributions):

n=num of samples, k=num samples falling in same region with x, V=region volume.

-Parzen Window: $\hat{p}(x)=\frac{1}{n}\sum_{i=1}^{n}\frac{1}{h^d}\varphi(\frac{\vec{x}-\vec{x}_i}{h})$ $\hat{p}(x|c)=\frac{1}{n_c}\sum_{\vec{x}_c\in c}\frac{1}{h^d}\varphi(\frac{\vec{x}-\vec{x}_c}{h})$ $\hat{p}(x)\underset{h\to0,n\to\infty}{\to}p(x)$

V is predetermined by h^d while k varies by this region.

-K nearest neighbors: $\hat{p}(x)\approx\frac{k}{nV}$ $P(A_i|x)=\frac{k_i}{k}$ $P(x|A_i)=\frac{k_i}{n_iV}$ $P(A_i)=\frac{k_i}{n_i}$

k is constant while V depends on distance from k^{th} closest neighbor.

7 Dimensionality

Filter method: select features according to some objective function (i.e. correlation with target: $\rho_{xy}=\frac{\sigma_{xy}}{\sigma_x \sigma_y}$)

Wrapper method: select features according to classification performance using each subset
- infeasible (requires n!/(s!(n-s)!) trials for subset from size s=1,...,n). may use greedy search methods.
- forward selection: choose each time the next feature most improving classification accuracy.
- backward selection: start with all features, remove next feature least reducing classification accuracy.
- combine random search of features, may overcome local-minima.
- slower performance than filters.

Feature extraction methods (transform the problem space into reduced dimension space):

PCA

- orthogonal transformation of the problem space into dimensionally lower one.
- new center of new axes would be the samples mean.
- axes are rotated in such that (principal) components of transformed sample will descend by variance.
- y_i=A(x_i-m) where: m=samples mean; A contains the d' highest eigenvalued eigenvectors of the scatter matrix.
- criteria: the distance from any sample to its projection on m: $J(a_1,...,a_n,\vec{e})=\sum_{k=1,...,n}\|(\vec{m}+a_k\vec{e})-\vec{x}_k\|^2$
- finds d' components of the samples with highest variance from mean (with greatest scatter).

FLD

- project multi-dimensional samples onto a single line: $y_i=w^T x_i$.
- criteria: maximize the between class scatter / within class scatter ratio: $J(\vec{w})=\frac{|\tilde{m}_1-\tilde{m}_2|^2}{\tilde{s}_1^2+\tilde{s}_2^2}$

  where: $\tilde{m}_i=\frac{1}{|C_i|}\sum_{\vec{x}\in C_i}\vec{x}$, $\tilde{s}_i=\sum_{\vec{x}\in C_i}(\vec{w}^T\vec{x}-\tilde{m}_i)^2$

  notation: $S_W=\sum_{\vec{x}\in C_1}(\vec{x}-\tilde{m}_1)(\vec{x}-\tilde{m}_1)^T+\sum_{\vec{x}\in C_2}(\vec{x}-\tilde{m}_2)(\vec{x}-\tilde{m}_2)^T$, $S_B=(\tilde{m}_1-\tilde{m}_2)(\tilde{m}_1-\tilde{m}_2)^T$

  hence: $J(\vec{w})=\frac{\vec{w}^T S_B \vec{w}}{\vec{w}^T S_W \vec{w}}$

  $\frac{\partial J(\vec{w})}{\partial \vec{w}}=0\Rightarrow\vec{w}=S_W^{-1}(\tilde{m}_1-\tilde{m}_2)$
- resulting discriminant function: C_1 *if* $\vec{w}^T\vec{x}>w_0$; C_2 *otherwise* .

MDS

- dimensionality reduction between spaces according to minimal loss of distances proportions between items.
- objective function to be minimized is some sort of strain function: $E=\sum_{i<j}(d(\vec{x}_i,\vec{x}_j)-d(\vec{y}_i,\vec{y}_j))^2$.
- instead of a scatter matrix, use dissimilarity/distance matrix S: $S_{ij}=d(x_i,x_j)=d(x_j,x_i)$.
- useful in visualization of similarities/dissimilarities of items in one space as distances in another space.

8 Computational Learning Theory

Evaluating Hypothesis

True hypothesis error over samples with dist : $error_{dist}(h)=P_{x\in dist}(h(x)\neq c(x))$

Test error of *n* samples where *r* of them are misclassified by h: $error_S(h)=r/n$

Estimation of true error within confidence interval of N%: $error_{dist}(h)\leq\frac{1}{n}(r+z_N(\frac{r}{n})(1-\frac{r}{n}))$

PAC-Learnability

concepts class C of problem space X is *PAC-learnable* by learner L using H if for all $c\in C$, distributions of X, L will produce an hypothesis h for which $error_{dist}(h)\leq\epsilon$ within probability (1-δ) such that 0<δ<1/2, 0<ε<1/2 and in polynomial time with respect to 1/ δ,1/ε, length(any c in C) and length(any x in X).

VC dimension

VC(H)=maximal size of subset of X shattered by H (any possible dichotomy of X is satisfied by some $h\in H$).

$H=\{a<x<b\|a,b\in R\}$	X=R	VC(H)=2
$H=\{\vec{w}^T\vec{x}+b\|\vec{w}\in R^d,b\in R\}$	X=R^d	VC(H)=*d*+1
H=ANN of *s d*-inputs perceptrons	X=R^d	VC(H)<=2*s*(*r*+1)log(*e***s*)
H=conjunction of *n*-boolean literals	X=$\{0,1\}^n$	VC(H)=*n*

Sample Complexity

1) The non-agnostic case (assuming that the target concept is in H):

-The probability that VS(H,D) may contain an ε-bad hypothesis (for which $error_{dist}(h)\geq\epsilon$) is at most $|H|e^{-\epsilon|D|}$

-- This is the upper bound for failure probability of learner L to PAC-learn the concept class C: $|H|e^{-\epsilon|D|}\leq\delta$

-- This will provide us the minimal number of sample required for PAC-Learnability: $|D|\geq\frac{1}{\epsilon}(\ln|H|-\ln(\frac{1}{\delta}))$

2) The agnostic case (no assumption that the target concept is in H):

-The probability of an *agnostic learner* to choose hypothesis h for which $error_S(h)-error_{dist}(h)>\epsilon$: $|H|e^{-2|D|\epsilon^2}$

-- This is a lower bound for minimal number of samples for PAC-learnability $|D|\geq\frac{1}{2\epsilon^2}(\ln|H|-\ln(\frac{1}{\delta}))$

3) The infinite hypothesis space case:

- Number of training samples *sufficient* for PAC-learnability: $|D|\geq\frac{1}{\epsilon}(4\log_2(2/\delta)+8VC(H)\log_2(13/\epsilon))$

- Number of training samples *required at least* for PAC-learnability: $max[\frac{1}{\epsilon}\log(1/\delta),\frac{VC(C)-1}{2\epsilon}]$

-- This is true for any C,δ,ε holding VC(C)>=2, 0< δ<0.01, 0<ε<0.125

PAC-Learnability Criterias:

1) Existence of a consistent/agnostic learner.
2) Number of samples satisfies the appropriate bound and is polynomial.
3) Processing time of the learner for each sample is polynomial.

9 Unsupervised Learning

Parametric

EM:
- Initialize the distribution parameters
- Repeat until *convergence:*
  E-Step: estimate the [E]xpected value of the unknown variables, given the current parameter estimate:
  $Q(h'|h) \leftarrow E[\ln P(Y|h')|h,X]$ $Y=XUZ$ (X=observable data, Z=unobservable data)
  M-Step: re-estimate the distribution parameters to [M]aximize the likelihood of the data, given the expected estimates of the unknown variables: $h \leftarrow \text{argmax}_{h'} Q(h'|h)$

EM for K-Gaussian means:

1) Expectation: $E[z_{ij}]=P(A_j|x_i)=\frac{p(x_i|\mu_j,\Sigma_j)P(A_j)}{\sum_{k=1}^{c} p(x_i|\mu_k,\Sigma_k)P(A_k)}$ since E[ln P(Y|h')] is a function of $E[z_{ij}]$.

2) Maximization: $\mu_j^{next}=\frac{\sum_{i=1}^{|X|} E[z_{ij}]x_i}{\sum_{i=1}^{|X|} E[z_{ij}]}$, $\Sigma_j^{next}=\frac{\sum_{i=1}^{|X|} E[z_{ij}](x_i-\mu_j^{next})^2}{\sum_{i=1}^{|X|} E[z_{ij}]}$

K-means Clustering

- initialize the means: $\mu_1, \mu_2,..., \mu_k$
- repeat until no change in $\mu_1, \mu_2,..., \mu_k$:
  - classify each sample with its closest mean
  - recompute $\mu_1, \mu_2,..., \mu_k$ as the means of the new clusters

Properties:
-- K-means EM using Euclidean distances from means instead of Mahalanobis and uniform priors $P(A_i)$.
-- Convergence is assured only for Euclidean spaces.
-- Fuzzy version: replace the 0/1 membership with probability criteria: $P(A_i|x_j)=||x_j-\mu_i||^{2/(1-b)}/\Sigma_{r=1,...,c}||x_j-\mu_r||^{2/(1-b)}$.

Non-Parametric

Metrics:
  Criterion functions to be optimized : S_W, S_B, $S_T=S_W+S_B$
  Inter-Clusters Similarity measures: d_{min}, d_{max}, d_{avg}, d_{means}

Hierarchical Clustering

- Find most similar clusters and join them together as a single cluster, repeat this procedure *c* times.

Mean-Shift

Move Parzen Window towards maximum increase in density.

- move from any point in the direction of the gradient: x(t+1)=x(t)+MSV(x(t)). $MSV(x) \approx \nabla(\frac{1}{nh^d}\sum_{i=1}^{n} K(\frac{x-x_i}{h}))$
- as a result, the Parzen window mean would shift towards center of mass.

--- Clustering: Classify points converging towards same center of mass the same.
--- Filtering: Repeat mean shift for each point x_i till convergence to some mass z_i . use z_i instead of x_i.

Quick-Sheet

Derivatives

$\frac{d}{dx} x = 1$

$\frac{d}{dx} x^n = n\, x^{(n-1)}$

$\frac{d}{dx} e^x = e^x$

$\frac{d}{dx} b^x = b^x \ln(b)$

$\frac{d}{dx} \ln(x) = 1/x$

Derivatives Identities

$\frac{d}{dx} c\, f(x) = c\, \frac{d}{dx} f(x)$

$\frac{d}{dx} (f(x) + g(x)) = \frac{d}{dx} f(x) + \frac{d}{dx} g(x)$

$\frac{d}{dx} f(g(x)) = \frac{d}{dg} f(g) * \frac{d}{dx} g(x)$ (chain rule)

$\frac{d}{dx} f(x)g(x) = f'(x)g(x) + f(x)g'(x)$ (product rule)

$\frac{d}{dx} f(x)/g(x) = (f'(x)g(x) - f(x)g'(x))/g^2(x)$ (quotient rule)

Logarithms

$log_b(mn) = log_b(m) + log_b(n)$

$log_b(m/n) = log_b(m) - log_b(n)$

$log_b(m^n) = n \cdot log_b(m)$

$ln(m) = log_e(m)$