|  |  |  |
| --- | --- | --- |
| Grow Tree (S)  If (y = 0 for all (x,y) in S) {return new leaf (0)}  Else{ if (y=1 for all (x,y) in S) {return new leaf(1)}  Elase{ choose best attribute x(j)=0; S1=all<x,y) in S with X(j) = 1; return new node(xj,Grow(S0),GrowTree(S1)}}    Training, Validation set  Random Noise | Concepts:  Occam’s Razor  Pure  Over fitting  Stop Grow  Missing attribute  Non-Boolean attributes  Noise-Variance-Bias  Generalization  Optimization  Post-Prune | Gain(S,A) = H(S) – sum(p(v)H(S(v) for v in value(A): Expected entropy)  Gain = Curr impurity – New impurity  Max Gain: reduction in impurity  Weight each bin by amount of data in it  H(V) = sum (-P(H=v)lg(P(H=v)), for v=0..1)  Laplace normalization  Conjugate Prior  Maximum A Posteriori |
| Reduced-Error Pruning:  Split data into training and validation set  Do until further pruning is harmful   1. Evaluation impact on validation set of pruning each possible node (pulse those below it) 2. Greedily remove the one that most improves validation set accuracy   Remove pre-conditions that improve the estimated accuracy  Sort rules using their estimated accuracy    Hypothesis spaces: nested collection of hypothesis, decision trees, rules, neural networks, cases | Concepts:  Statistical test  Regularization  Complexity Penalty  Set of rules,  Multiway Split  Representation  Algorithm,Data,Programs  Experience, Experiment  Performance, Task  Supervised/Unsupervised  Parameteric/ Nonparameteric  Reinforcement Learning    Forecasting or prediction  Classification/ Ranking  Outlier/ trend/ Pattern | Gain ration for attributes with many values = Gain(S,A)SplitInFormation(S,A)  SplitInformation(S,A)=-sum(|S(i)|/|S|log|S(i)|/|S|  S(i) subset of S for which A has value v(i)  Treat missing values as another value  Assign most common value  Assign common value based on class that the example belongs to  Assign probability p(i) to each possible value v(i) |
| Instant based learning, Target function  Concept: Boolean function  Generative vs. discriminative  Positive/negative examples  Classifier: discrete value function  Version space: the space not ruled out yet  Representation, Optimization, Evaluation  Feature Engineering (properties)  Iterative process  Begin with random  Repeat until now error  Representation: instances, hyper planes, decision trees, sets of rules, neural networks, graphical models  Evaluation: Accuracy, precision/recall, squared error, likelihood, posterior probability, margin  Optimization: Greedy search, Branch and bound, Gradient descent, Quasi-Newton, Linear programming, quadratic programming  Minkowski distance, incremental growth, computational cost, curse      Alpha learning rate  Logistic: 1/(1+exp(-z))        MAP estimate:        Stochastic gradient descent (incremental)  Batch Gradient Descent  Softmax regression              PAC Learning: Probably approximately correct: The only reasonable expectation of a learner is with high probability it learns a close approximation to the target concept. With probability at least a system learn a concept with error at most .  X: set of instances, H: set of hypotheses, C: set of possible target concepts. Training examples generated by fixed unknown distribution D.  True error: , probability that h will misclassify an instance drawn at random according to D.  Training error: on training misclassify  True error: over future random instances  C concept class is **PAC Learnable** by learner L, using hypothesis space H iff for all c in C, distribution D over X; learner L by sampling random examples from D, will with probability at least  output a hypothesis such that , time polynomial in  Sample complexity: # training examples are required for a problem of given size?  **Consistent Learner**: it always output a hypothesis with zero error on D whenever H contains such a hypothesis => bound the # of examples needed to ensure the version-space (subset of hypotheses in H consistent with training data D) contains no hypotheses with unacceptably high error  **-exhausted version space**: iff every hypothesis in it has true error less than or equal to , there are enough training examples to guarantee than any consistent hypothesis has error at most  **Haussler 1988 theorem:** if the hypothesis space H is finite, and D is a sequence of independent random examples for some target concept c, then for any , the probability that the version space is not -exhausted is less than or equal to  the probability that any is consistent for all m:      Any consistent learner, given at least  examples will produce a result that is PAC (sufficient, but not necessary, overestimate)  **Exmple1**.Conjunction over n Boolean , as each feature can appear positively, negatively or not appear (missing)  **Exmple2**.Tennis with 1 attribute of 3 value, 9 attribute with 2 values, conjunction of features:  while, example in domain are  **Exmple3**.Boolean function over n Boolean features such as the hypothesis space of DNF or decision trees. .  **Agnostic Learning**: don’t assume , Hoeffding bounds:  **Vapnik-Chervonenkis(VC) dimension**: VC(H)  **Shattering Instances**: A hypothesis space is said to shatter a set of instances iff for every partition of the instances into positive and negative there is a hypothesis that produces that partition.    Since  partitions of m instances, in order for H to shatter instances: . VC(H)=2  Unhbiased hyp. Space shatters the entire instance space; shattered more => more expressive => less biased hypth space  VC(H): the size of the largest finite subset of X shattered by H. could be infinity.  If at least one subset of X of size d that can be shattered then , if no subset of size d can be shattered , VC of hyper plane in d-dim space is d+1, axis parallel rectangle: VC(H)=2d  PAC:  General lower bound on the minimum number of examples necessary for PAC learning (Ehrenfeucht et al. 1989): any concept class C such that VC(H)>2, any learner L and any    **EM:** Start with random parameters: Repeat until convergence, \* **1** Complete the incomplete data using current parameters. \* **2** Update the parameters based on the completed data STEP 1: computes expected sufficient statistics; how likely the completion is (E-step) STEP 2: maximizes the likelihood (M-step), use weights from prev step  weighted(Expect.) MLE | Concepts:  Similarity/ Preplexity  Cond/ Joint Prob/Likelihd  Collaborative Filtering  Feature Selection  Max Likelihood  Grad Discent  Close Form  Expectation Max  Markove Dec Process  Dimensionality reduction  Inductive Learning  Hidden Function  Unseen Vector  Simple Rule  Removal of uncertainty  Start small and enlarge hypothesis  Prior Knowledge (Language to Express)  Rule grammars and stochastic models  Calculate at log scale  Transformation  Kernel  Support vector machine  Decision boundary  Voroni diagram  Differentiability  Aka sigmoid      Direct/ indirect computation  Tanh(x)=1-e^(-2x)/(1+e^(-2x))  d/d(x)tanh(x)=1-tanh^2(x)  When model incorrect, LR is less biased: does not assume cond indep  Naïve bayes: O(log n)  LR: O(n), trade off train test  GNB converge more quickly to its asymptote  Objective Loss function                    Dual problem  Slack Variable, Majority rule    There exists a distribution of D and target concept in C such that if L observes fewer than:    Examples, then with probability at least , L outputs a hypothesis having error greater than  In practice what matters most is: (1) prior knowledge (2) Data distribution (3) Amount of training data  **Avoid over fitting by:** Regularization, pruning, inclusion of penalty, cross validation  **Occam’s razor**: one should not use classifiers that are more complicated than necessary  Evolution: we have a strong selection pressure to be computationally simple  Satisficing (Herb Simon): Creating an adequate though possibly non-optimal solution  **Measure of expressiveness**: |H| or VC(H), quantifying bias and relating it to generalization  **Worse case upper bound**  Feature engineering, Data cleaning, Suitable evaluation method, suitable learning algorithm (optimization)  K-fold cross validation: recycle the data  **Classifier Ensemble**: combine classifiers  Statistical process of generating data, not rep of training set  Bias: measure accuracy  Variance: measures precision    Averaging reduces variance:    Average models => **Bagging**: Bootstrap aggregation (draw with replacement), create k bootstrap, train distinct classifier on each, classify new instances by majority vote/average    In practice correlation = smaller improvement  **Bagging** helps **unstable** classifiers such as Neural Network and decision tree, and it may not improve stable classifiers such as k-nn, NB  **Boosting** helps weak learners (slightly better than chance prediction on any data set): (1) Weights all training sample (2) Train model on training set (3) Compute error of model on training set (4) increase weights on training cases model gets wrong (5) train new model on reweighted training set (6) repeat more than 100 times (7) final model: weighted prediction of each model    **AdaBoost**  (1) Initialize the data weighting coefficientsfor n=1..N  (2) for m=1…M:  (a) Fit a classifier to train data by minimizing the weighted error func:    (b) Evaluate the quantities:    And then use these to evaluate:    (C)Update the data weighting coefficients:    : No errors  All errors:  Random  If in Ada-boost weights are hard to deal with => Resample (Draw a bootstrap sample from the data, with prob of drawing each example prop. to its weight)  **Elimination:** Select an Ordering, multiply factors, , sum out  **Sufficient statistics** | P(D|theta)=thetah^alpha(H)(1-thetah)^alpha(T)  Beta priors for binomial  Posterior Distribution  Gaussian: mean of sample: mean, std of the sample variance (bias, correction)  Conditional independence  Position in the document does not matter (ignore word order)  Bag of words  First order markov assumption:  P(X,Y|Z)=P(X|Z)P(Y|Z)  Naïve Bayes: Given class features are independent  P(y)=count(Y=y)/ sum(Count(Y=y’))  Likelihood  P(X(i)=x|Y=y)=Count(X(i)=x, Y=y)/sum(Count(X(i)=x’,Y=y))  Mean of beta distrib:  a(H)+b(H)-1/(a(H)+b(H)+a(T)+b(T)-2)  Laplace’s estimate:  Pretend to saw outcome k extra times:  P(Lapla,k,x)=c(x)+k/N+k|X|  Laplace for conditionals:  P(Lapl,k,x|y)=c(x,y)+ k/(c(y)+k|X|)  Smooth each condition independently  Gaussian Naïve Bayes:  P(X(i)=x|Y=y(k))=N(mu(ik),s(ik))  Mean and var like normal          LR: no closed-form, concave, opt with grad ascent,  GNB needs less data  LR gets better to better solution in limit      Reparability, Convergence        **Boosting** may hurt performance on noisy data**. Bagging** is easy to parallelize.  **k-mean iterative clustering:**  Pick K random points as cluster centers (means) –Alternate:  •Assign data instances to closest cluster center  •Change the cluster center to the average of its assigned points  –Stop when no points’ assignments change  used for **segmentation**  Two stages each iteration:  –Update assignments: fix means c, change assignments a  –Update means: fix assignments a, change means c  Converges as it only decrease **total distance** of points from cluster mean, as the min square error Euclidean distance is mean  **Agglomerative Clustering**  •Agglomerative clustering:  –First merge very similar instances –Incrementally build larger clusters out of smaller clusters •Algorithm:  –Maintain a set of clusters –Initially, each instance in its own cluster –Repeat: •Pick the two closest clusters •Merge them into a new cluster •Stop when there’s only one cluster left •Produces not one clustering, but a family of clusterings represented by a dendrogram  How closest?Many options:  –Closest pair (single-link clustering) –Farthest pair (complete-link clustering)  –Average of all pairs –Ward’s method (min variance, like k-means) •Find pair of clusters that leads to minimum increase in total within cluster distance after merging  **EM: Soft Clustering**: gives probabilities that an instance belongs to each of clusters. Tell generative story P(X|Z)P(Z), Finite mixture model:      The membership weight express our uncertainty about which of the “K” components generated the vector of the data  **Gaussian Mixture Model (GMM)**:      •Start with random parameters  •Find a class for each example (E-step) –Since we are using probabilistic classification, each example will be given a vector of probabilities  •Now we have a supervised learning problem. Estimate the parameters of the model using the maximum likelihood method (M-step) •Iterate between the E-step and M-step until convergence  E-step: (Yields a N x K matrix)  –Compute  for all data points indexed by “i” and all mixture components indexed by “k.”  •M-step: –Use the membership weights and data to compute the new parameters      Probability rule: Sum rule, product rule (chain), conditional, independence, conditional independence, Bayes rule, joint or marginal distribution,  Bayesian NW: compact representation, conditional indep, DAG causeffect, Inference NP hard, Variable Elimination, Importance Sampling, MCMC, Belief propag    (Domain size x(i)-1)\*(All possible combination of x(1)…x(j)) for CPT(x(i)|x(1)…x(j)) parameters  **Connect all child together** complexity: n\*exp(max(children))  **MPE** replaces sum out with max-out  **Factor graph:** mult income factor, sum out, junction tree, **Likelihood (** conditioned on parent) var: |