**Constraint Satisfaction Problems (CSPs)**

Complete Solutions- all variables are assigned values

Consistent Solutions- do not violate any constraints

Unary Constraints- involve a single variable

Binary Constraints- involve pairs of variables

Higher-order Constraints­- involve 3 or more variables

-hill-climbing, simulated annealing, genetic algorithms typically work with complete states

Min-Conflicts Heuristic- choose value that violates the fewest constraints, f(n) = total number of violated constraints

Local Search (Min-Conflicts Algorithm)- advantages are that it is simple and fast, disadvantages are that it might not search entire search space, doesn’t allow worse moves, not complete

while state not consistent do

1. Pick a variable, var that has constraint(s) violated
2. Find value, v, for var that minimizes the total number of violated constraints (over all variables)
3. Var = v

Standard Tree Search Formulation- states are defined by all the values assigned so far, successor function = assign a value to an unassigned variable, goal test = the current assignment is complete and consistent, all variables assigned a value and all constraints satisfied

Backtracking w/ Consistency Checking- perform consistency checking when node is generated, successor function assigns a value to an unassigned variable that does not conflict with all current assignments

while not all vars in state assigned a value do

pick a variable (randomly or with a heuristic)

if it has a value that does not violate any constraints

then assign that value

else

go back to previous var and assign it another value

Backtracking Search- Depth-first search algorithm which goes down one var at a time, at deadend it backs up to last var whose value can be changed without violating any constraints, algorithm is complete and could expand entire search space

Improving Backtracking Efficiency- heuristics can give huge gains in speed

Most-Constrained Variable- choose the variable with the fewest legal values (called the minimum remaining values heuristic)

Most-Constraining Variable- choose the variable with the most constraints on the remaining variables (called degree heuristic)

Least-Constraining Value- the one that rules out the fewest values in the remaining variables

Forward Checking Algorithm- keep track of remaining legal values for all variables, deadend when any variable has no legal values

Constraint Propagation- repeatedly enforces constraints for all variables, when you delete a value from a variable’s domain, check all variables connected to it and if any of them change, delete all inconsistent values connected to theme

Arc Consistency- simplest form of propagation that makes each arc consistent, detects failure earlier than forward checking

**Neural Networks**

Linear Threshold Unit- step function

Sigmoid Function- Logistic function (1/(a+e^(-x))

Rectified Linear Function­- step function but linear function for the step

Perceptron Learning Rule- DeltaW = alpha\*x\*(T-O), where X is input, performs gradient descent(hill-climbing) in “weight space”

Perceptron Convergence Theorem- if a set of examples is learnable, then PLR will find an appropriate set of eights, if a solution exists, PLR’s gradient descent is guaranteed to find an optimal solution for any 1-layer neural network

Limits of Perceptron Learning- perceptron’s output is determined by the separating hyperplane defined by (w1x1) + …. + (wnxn) = b, can only learn functions that are linearly separable

Multi-Layer, Feed-Forward Neural Nets- uses back propagation for learning weights in these networks, gradient-descent algorithm to minimize the total error on the training set, must use Simoid Function to update weights and not Step Function because derivative is 0

Sum Squared Error (E)- Ei = (T1-O1)^2 + …. + (Tt-Ot)^2

Updated Weights- for weights between hidden and output units

**Support Vector Machines**

Maximum Margin Linear Classifier- the linear classifier with the maximum margin

Support Vectors- the data points that the margin pushes against

Plus-plane- w\*x + b = +1

Minus-plane- w\*x + b = -1

Vector w is perpendicular to plus and minus-plane

­Computing the Margin- w\*(x+) + b = +1, w\*(x-) + b = -1, x+ = x- + lambda \*w, ||x+ - x-|| = M, M = 2/||w||

More than Two Classes- can only handle two-class problems

Non Linearly-Separable Data- allow a few points on tahe wrong side (slack variables) or map data to a higher dimensional space, and do linear classification there (kernel trick)

Slack Variables- minimize ||w||^2 + C(# train errors), if C is too big means very similar to LSVM and may use many support vectors and overfit, C too small means we allow misclassifications in the training data and we may underfit

**Uncertainty**

Fully Joint Probability Distribution- making a joint distribution of N variables, you must list all combinations of values(if each var has k values,there are k^N combinations), k^N – 1 degrees of freedom

Probability for Discrete Events- an agent’s uncertainty is represented by P(A=a), a single probability called an unconditional or prior probability

The Axioms of Probability- P(A v B) = P(A)+P(B)- P(A,B), 0<= P(A) <= 1, P(-A) = 1-P(A)

Full Joint Probability Distribution Table- for n random variables, each taking k values, has k^n entries

Conditional Probability- P(-a | e) = 1 – P(a | e), P(A | B) = P(A,B) / P(B)

Product Rule- P(A,B) = P(A|B)P(B)

Normalization- P(A | B) = P(A,B) / P(B)

The Chain Rule- P(A1,A2,….,An) = P(A1)\*P(A2|A1)\*P(A3|A2,A1)\*…\*P(An|A1,A2,….,An-1)

Conditionalized Version of Chain Rule- P(A,B|C)=P(A|B,C)P(B|C)

Inference with Bayes’s Rule- P(F|H) = P(F,H)/P(H)= [P(F)\*P(H|F)]/P(H)

Conditionalized version of Bayes’s Rule- P(A|B,C) = P(B|A,C)P(A|C)/P(B|C)

Addition/Conditioning Rule- P(A) = P(A,B) + P(A,-B), P(A) = P(A|B)P(B) + P(A|-B)P(-B)

Independence- two events A,B are independent if P(A,B) = P(A) \* P(B), P(A,-B) = P(A) \* P(-B) ….., P(A|B) = P(A), P(B|A) = P(B), P(A|-B) = P(A)

Conditionally Independent- A and B are conditionally independent given C means… P(A|B,C) = P(A|C), P(B|A,C) = P(B|C), P(A,B| C) = P(A|C)\*P(B|C)

Naïve Bayes Classifier- Assume k classes and n evidence variables, each with m possible values, k-1 values needed for computing P(Y=c), (m-1)k values needed for computing P(X=v|Y=c) for each evidence variable X, so (k-1) + n(m-1)k values needed instead of exponential size FJPD table

Add-1 Smoothing- ensures that each conditional probability is greater than 0, computer conditional probabilities as P(A=i|Y=c)= (Nic + 1)/(Nc + m), where Nic is number of times attribute A has value I in all training instances with class c, Nc is number of training instances with class c, and assume attribute A has m possible values; compute prior probabilities as P(X=c) = (Nc+1)/(n+k), where k is possible classes for class variable Y and n is size of training set

**Bayesian Networks**

Bayesian Network- assumes conditional independence, if network is sparse, each node has most M parents, then only needs O(Nk^M) parameters

Casual Chain- A->B->C (A is independent of C given B)

Common Cause- B<-A->C (B and C are independent given A)

Common Effect- A->C<-B (A and B are independent)

Computing Joint Probabilities- P(a,b) = P(a,b,c,d)+P(a,b,c,-d)+P(a,b,-c,d)+P(a,b,-c,-d)

Inference by Enumeration- P(B|J,M)=P(B,J,M)/P(J,M)

**Speech Recognition**

Phones- human languages are limited to a set of about 40 to 50 distinct sounds called phones

Phonemes- are equivalence classes of phones that can’t be distinguished from each other in a given language

Speech Recognition Model- P(Words|Signal) = P(Signal|Words)\*P(Words)/ P(Signal), P(words) is the language model and P(Signal|Words) is acoustic model

Language Model- P(Words) is a joint probability of Words, P(Words) = P(w1,w2,w3,w4,w5,w6,…,wn) (solved w/ chain)

First-Order Markov Assumption- Probability of a word depends only on the previous word, P(Wi|w1….Wi-1) = P(Wi|Wi-1), therefore language model simplifies to P(w1,w2,w3,w4,….,wn) = P(w1)P(w2|w1)P(w3|w2)… which is called the bigram model

Trigram Model- P(wi|w1,w2,…,wi-1)= P(wi|wi-1,wi-2)

Probabilistic Finite State Machine- an almost fully connected directed graph, joint probability is estimated for the bigram model by starting at START and multiplying the probabilities of the arcs

Acoustic Model- P(Signal|Words) is divided into two probabilities of P(Phones|Word) and P(Signal|Phones), P(Phones|Word) is computed with Markov Model and P(Signal|Phones) is computed with Hidden Markov Model

1st Order Markov Assumption- state Qt+1 is conditionally independent of everything given Qt, in other words P(Qt+1|Qt,Qt-1…,Q1) = P(Qt+1|Qt)

State Transition Matrix- describes state to state associated with a graph, sum of values in row = 1

PI Vector- the prior probabilities of the initial state at time t= 0

Hidden Markov Model- arcs connecting hidden states and observable states represent the probability of generating an observed state given that the Markov process is in a hidden state

Observation Likelihood Matrix- stores probabilities associated with arcs from hidden states to observable states

**Face Detection and Recognition**

Ensemble Methods- aggregation of predictions of multiple classifiers with the goal of improving accuracy by reducing the variance of an estimated prediction function, combining multiple classifiers often produces higher accuracy than any individual classifer

Decision Stump- each weak classifier is called this

Bagging- create classifiers using different training sets(used for Random Forests) , where each training set is created by bootstrapping, ie drawing examples from all possible training examples

Boosting- sequential production of classifiers, where each classifier is dependent on the previous one, make examples misclassified by current classifier more important in the next classifier, a class of ensemble methods for sequentially producing multiple weak classifiers, where each classifier is dependent on the previous ones

Adaboost- assume 2-class classification problem (2 classes are +1,-1; computes function g(x) = alpha1h1(x) + alpha2h2()x +… where h(x) is +1 or -1, final classifier is C(x) = sign(g(x)), weighted majority combination of all the week classifiers

Viola-Jones Real-Time Face Detector- want a very small # of false positives



