Notes: DP/A*/Information

Alan Yuille
Department of Statistics, UCLA
Los Angeles, CA 90095
yuille@stat.ucla.edu

Abstract

1 Introduction

Dynamic Programming, A*, and Conditional Entropy.

2 Graphical Models for Deformable Templates

Recall that we can formulate Deformable Templates by defining distributions on graphs. We use fixed graphs with nodes with state variables $\{x_i : i = 1, ..., M\}$. (For simplicity, these can be the positions of nodes of the deformable template but they could also include other attributes such as orientation). The image is $I(x)$ and we can apply a filter $f(.)$ to it (e.g., and edge detector).

We can write down a Markov Model:

$$P(X|I) = \frac{1}{Z} \exp\{-\sum_i \phi_i(f(I;x)) - \sum_{ij} \phi_{ij}(x_i, x_j)\}. \tag{1}$$

The first term $\phi_i(f(I;x))$ is an appearance model and the second term $\phi_{ij}(x_i, x_j)$ specifies the geometric prior. The appearance model is typically discriminative – but, in principle, it could be sampled from to obtain features (which might not correspond to an allowable image – e.g., there might not be an image which is completely consistent with the features sampled from the model).

The geometric prior can be learnt. SC-Zhu used minimax entropy to learn priors for generic contours (using the derivatives of the curves as features – and even connections across symmetry axes). But here we are concerned with deformable templates.

Coughlan *et al* hand-designed a template by making it a deformation of a specific hand (his own hand). This was enforced to have a linear structure so that the graph nodes can be ordered and $\phi_{ij}(x_i, x_j) = 0$ unless $j = i + 1$. This gives an open structure on which dynamic programming can be performed over the full energy $E(X) = \sum_i \phi_i(f(I;x)) + \sum_i \phi_{i,i+1}(x_i, x_{i+1})$.

DP is performed as follows. Express the energy as $E(X) = \sum_i \psi_i(x_{i-1}, x_i)$. Define $m_1(x_1) = \min_{s_1 \in S_1} \psi_1(s_i, x_1)$, where S_1 is the set of values s_i that node x_0 can take (e.g., all positions in the image). Recursively compute $m_t(x_t) = \min_{s_i \in S}\{m_{t-1}(s_i) + \psi_{t+1,t}(s_i, x_t)\}$. The optimal value $E(X^*) = \arg\min E(X)$ is obtained by $\min_{s \in S} m_M(s)$ (proof by induction). To compute each $m_t(x_t)$ takes $O(k^2)$ operations where $k = |S|$ and to compute all operations is $O(k^2 M)$. To find the optimal path we need to trace back. Let $x_M^* = \arg\min_{s \in S} m_M(s)$, then recursively $x_t^* = \arg\min_{s \in S}\{m_t(s) + \psi_{t,t+1}(s, x_{t+1}^*)\}$ (break ties arbitrarily). It is easy to check that the configuration

$X^* = (x_0^*, ..., x_M^*)$ is optimal. The computation is $O(k^2 M)$ – but this can be large since the state space of each x_t can be very big (e.g., every point in the image).

Coughlan *et al* did the following. **Learning**: there was no explicit learning but Coughlan did sample from the prior to generate samples of the object (sampling for a linear graph like this is efficient because the linear structure can be exploited) and adjust the model parameters so that the samples looked like hands. **Inference**: use pruned dynamic programming. First, restrict the size of the state space S by only allowing points where an edge is detected above threshold – to prevent errors when points are undetected must ensure that the observation potential $\phi_i(f(I(x_i)))$ is robust). Second, prune out paths whose energy values fall below a threshold (e.g. remove any state x_t such that $m_t(x_t) > T$). It was easy to select thresholds for pruning which gave a very fast algorithm with very few false positives. (Coughlan's model was extended to include orientation and to allow for occlusion – see paper).

3 A*

A* is a method for constructing algorithms where the goal is to get from a start node to a goal node by selecting a path through the graph to maximize (or minimize) a performance function. This was a problem studied extensively by the early Artificial Intelligence community. There were two extreme strategies – (i) depth first search that explores a long single path, and (ii) breadth first search which explores many paths.

A* keeps a set N_t of nodes which is initialized to the start node $N_0 = \{S\}$, where S is the start node. The set of nodes is increased by exploring to neighbouring nodes. At each node A there is a cost $C(A, S)$ to get there from the start node S and an estimated cost $C_E(A, G)$, computed by a heuristic, to get from A to the goal state G. A* grows N_t by selecting the node that is a neighbour to N_t and has maximal total cost $C(B, S) + C_E(B, G)$ and adds this to N_t.

The A* heuristic is *admissible* if the heuristic cost $C_E(A, G)$ is an upper bound of the (unknown) try cost. In this case, A* is guaranteed to converge to the correct answer (i.e. the best path is found by growing N_t until G is included – and then to stop and not search for any other paths). Dynamic Programming can be interpreted as a form of A* (where the heuristic is so large that the algorithm explores in breadth).

In practice, admissible A* may be too conservative a strategy. The heuristic means that you will rarely be satisfied with any path (including the true one) since you are always comparing it to the best value it could take. As inadmissible heuristic can perform faster and may only result in very rare errors. Essentially, an admissible A* heuristic guards you against worst case situations which may never occur. We discuss this in the next section.

4 Geman and Jedynak Model and Order Parameter Analysis

The Geman and Jedynak model was used to detect roads (highways) from aerial photographs. It specifies a road by a sequence of arcs segments a_i defined on a tree. Each segment is of fixed size (e.g. 8 pixels long). The tree is constructed by allowing an arc to split into three segments – one straight ahead, one five degrees to the left, one five degrees to the right. There is a probably distribution on these splits, which specifies the prior on the geometry of a road (note: highways are designed to have certain geometries – e.g., to have small curvature and to minimize changes in curvature). There are feature detectors which are intended to detect road segments – these are characterized by their probability distributions evaluated on and off roads.

Formally, we let X denote a connected sequence of arcs – this is a hypothesis for the road. For an arc a in the image, we define a test Y_a which consists of applying a filter to the image and measuring the output y_a. The distribution $P(Y_a = y_a|on), P(Y_a = y_a|off)$ are the distributions of filter responses conditioned on whether arc a is, or is not, on the road. These distributions can be learnt from training data (in practice, G&J discretized the filter response and measured the histograms on and off the roads). There is also a probability $\psi(a_j, a_{j+1})$ on the geometry which specifies the probability that arc a_{j+1} follows arc a_j (i.e. zero probability unless the arcs are connected, and non-zero probabilities for going straight or five degrees to the left or right).

We can derive a conditional distribution (similar to derivation for Snakes) to get:

$$\log P(X|Y) = \sum_i \log\{\frac{P(y_i|on)}{P(y_i|off)}\} + \sum_i \psi(a_i, a_{i+1}). \quad (2)$$

The goal is to find $X^* = \arg\max P(X|Y)$. The starting arc of the road is specified.

This can be formulated as A* search. This involve searching the tree of possible arcs. Suppose the road consists of a total of N arcs. We connect the end point of all the 3^N possible paths to a goal node (to make this suitable for A*).

The cost of a path from the start node a_0 to a node a_m is $\sum_{i=1}^{m} \log\{\frac{P(y_i|on)}{P(y_i|off)}\} + \sum_{i=1}^{m} \psi(a_i, a_{i+1})$.

A* requires a heuristic. An admissible heuristic must provide an upper bound of the cost of the remaining $N-m$ arcs. This can be given by $(N-m)\{\max_y \log\{\frac{P(y|on)}{P(y|off)} + max_{a_{(i+1)}}\psi(a_i, a_{i+1})\}\}$. But it is very very unlikely that the cost of the remaining arcs will be as big as this. *If we assume that the probability distributions accurately describe the data*, then the cost of the remained in more likely to be close to the expected cost $(N-m)\{KL(P(.|on)|P(.|off)) - H(P_{\Delta G})\}$ (where we take expectations with respect to $P(y|on)$ and $P_{\Delta G} = \psi(.,.)$, $KL(.)$ is the Kullback-Leibler divergence and H is the entropy). This follows from the law of large numbers – the average $\frac{1}{N}\sum_i x_i \mapsto \sum_x xP(x)$ for large N if the x_i are identically independently distributed from $P(x)$. The worst case cost is clearly far too conservative and will waste time exploring arc sequences which have very little chance of being the correct path. But the expected cost is not suitable either as a heuristic because it is certainly not an upper bound and can lead to large errors.

Yuille and Coughlan analyzed this problem by putting a probability distribution over the set of problem instances – using the distributions $P(.|on), P(.|off)$ and $P_{\Delta G}$. They showed that you can define a hueristic so that the A* algorithm is guaranteed to converge in $O(N)$ time to a path sequence which has (with high probability) only a small number of errors. This analysis is complicated and requires using techniques from large deviation theory to put bounds on the probability of rare events. Yuille and Coughlan also proved that the probability of detecting the true path directly – i.e. the Bayes risk – depended only on a single *order parameter* which is a function of the distributions. Detection may be impossible because although each false path (i.e. sequence of arcs that do not correspond to a road) has low expected score, there are an exponential number of them – so there is a non-zero chance that one of them has a higher score than the true road.

The point is that computer vision methods – particularly Bayesian formulations – make assumptions about the nature of the data. If we have a Bayes formulation $P(I|W)P(W)$ then this induces a distribution $P(I) = \sum_W P(I|W)P(W)$. When we analyse performance of algorithms to estimate $W^* = \arg\max P(W|I)$ we should take into account that the data is drawn from $P(I)$ – we should not have to be concerned with performance of the algorithm for "worst case data" which many have infinitesimally small $P(I)$. Of course, if we can obtain convergence results for any input I this is clearly preferable – but it seems impossible for real world problems as complex as those required to solve vision. In practice, it is rarely possible to perform this type of analysis using a distribution $P(I)$ (Yuille and Coughlan, Pearl and Karp, related work by Geman, Amit, Bouchard). So you reduce to computational evaluation – running the algorithms on representative data and obtaining the results.

5 Information Concept: Conditional Entropy and Mutual Information

Geman and Jedynak proposed an algorithm for finding the best path which is based on concepts from information theory (but it can be interpreted as A* approximately). This algorithm proceeds be performing the filter tests in an order which maximizes the expected gain in information. These tests are performed by searching along the arcs. In short, performing a test Y_k corresponds to exploring the corresponding arc.

Suppose they have performed tests $Y_1, ..., Y_k$ and obtained results $b_k = (y_1, ..., y_k)$. Then they choose the next test Y_{k+1} to minimize the conditional entropy of X:

$$H(X|b_k, Y_{k+1}) = -\sum_{y_{k+1}} P(Y_{k+1} = y_{k+1}|b_k)\{\sum_X P(X|b_k, Y_{k+1} = y_{k+1}) \log P(X|b_k, Y_{k+1} = y_{k+1}). \quad (3)$$

Here $P(X|b_k, Y_{k+1} = y_{k+1})$ is the probability distribution on the road path X conditioned on the observations b_k and on the (unknown) value of the proposed observation Y_{k+1}. The distribution $P(Y_{k+1} = y_{k+1}|b_k)$ is probability that the observation Y_{k+1} takes value y_{k+1} conditioned on the measurements b_k. Intuitively, the measurements b_k determine a conditional distribution $P(X|b_k)$ on the possible road configurations X and this determines a distribution $P(Y_{k+1} = y_{k+1})$. In short, $P(Y_{k+1} = y_{k+1}|b_k) = \sum_X P(Y_{k+1} = y_{k+1}|X)P(X|b_k)$. As can be shown (Yuille

and Coughlan) these computations are straightforward to compute by exploiting the tree structure – and results in an algorithm very similar to A*.

Minimizing the conditional entropy is a search strategy used in many algorithms for searching visual scenes. There is even some evidence ("Mr. Chips" – Legge *et al* that humans use this strategy when reading text – you do not necessarily read each letter if you can predict it from previous context but move your eyes to minimize the conditional entropy).

Selecting the test which minimizing the conditional entropy is also equivalent to the strategy of selecting the test which maximizes the mutual information between the test Y_{k+1} and the road configuration X. This follows from the standard identity $I(Y_{k+1} : X|b_k) = H(X|b_k) - H(X|b_k, Y_{k+1})$ where $I(Y_{k+1} : X|b_k)$ is the mutual information between Y_{k+1} and X. Hence minimizing $H(X|b_k, Y_{k+1})$ with respect to Y_{k+1} is equivalent to maximizing $I(Y_{k+1} : X|b_k)$ with respect to Y_{k+1}.

Workers in probabilistic medical expert systems (e.g., Lauritzen and Spiegelhalter) often use the "maximize mutual information" strategy to determine which tests to use, or to determine if a test is worth doing. For example, X could denote whether or not a patient has a disease. Y_{k+1} is a test that can be performed, perhaps requiring the use of expensive technology and with a possible risk to the patient's health. A medical expert system can evaluate the information provided by the test Y_{k+1} about X, taking into account the (probabilistic) knowledge $P(X|b_k)$ about state X obtained from the previous tests b_k. The system can evaluate the mutual information and determine whether the test is worth doing.