Substrate and Superstructure for Computational Intelligence

Gary Bradski, Intel Research
Background

• Background -- Research to enable AI
• Cognitive or “ML SQL” via MAP-REDUCE
• Substrate:
  – A Machine Learning “Basis Set”:
• Superstructure:
  – Embed in Hierarchy for global ⇔ local info flow
• Extension to Nano
Background Goal: ~Enable AI

What I do: Enable the next step of AI

OpenCV Function Areas


General Image Processing Functions

Image Pyramids

Segmentation

Geometric descriptors

Recognition

Features

Measures

Tracking

Utilities and Data Structures

Matrix Math

Fitting

Gary Bradski, garybradski@gmail.com
Machine Learning Library (MLL)
Growing subsection included in OpenCV above

**Bkgnd Goal:** ~Enable AI

**CLASSIFICATION / REGRESSION**
- CART
- Statistical Boosting
- MART
- Random Forests
- Stochastic Discrimination
- Logistic
- SVM
- K-NN

**CLUSTERING**
- K-Means
- Spectral Clustering
- Agglomerative Clustering
- LDA, SVD, Fisher Discriminate

**TUNING/VALIDATION**
- Cross validation
- Bootstrapping
- Sampling methods
Bkgn Goal: ~Enable AI

Probabilistic Network Library (PNL)
http://www.intel.com/technology/computing/pnl/

INFERENCE
Junction Tree
Most Probable Explanation
Loopy Belief Propagation
Out of clique inference in clique tree
Likelihood Weighted Particle Generation
Likelihood Weighted Data Dependent rule
Gibbs
Sum/Max product
Forward Sampling in a BayeNet

LEARNING
Greedy hill climbing with random restarts
Expectation Maximization (EM)
Structural EM
Maximum Likelihood MLE
Divide and conquer
Bayesian parameter learning
Greedy structure search

DISTRIBUTIONS
Tabular, Deterministic, Tree
Gaussian, Mixture of Gaussian

Figure 1: POSMDP for daily activity prompting
OUTLINE
ML or Cognitive SQL

- Background -- Research to enable AI
- Cognitive or “ML SQL” via MAP-REDUCE
- Substrate:
  - A Machine Learning “Basis Set”:
- Superstructure:
  - Embed in Hierarchy for global ⇔ local info flow
- Extension to Nano
Near Term AI Target: ML or “Cognitive SQL”

**SQL Today:**

Processors + platforms + SW are optimized to do this fast and well

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Age</td>
<td>Grade</td>
<td>Salary</td>
</tr>
<tr>
<td>Dave</td>
<td>37</td>
<td>7</td>
<td>$82,000</td>
</tr>
</tbody>
</table>

Gary Bradski, garybradski@gmail.com
ML or “Cognitive SQL”

**SQL Today:**

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Age</td>
<td>Grade</td>
<td>Salary</td>
<td>...</td>
</tr>
<tr>
<td>1</td>
<td>Dave</td>
<td>37</td>
<td>7</td>
<td>$82,000</td>
</tr>
<tr>
<td>2</td>
<td>...</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>...</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Cognitive SQL:**

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Age</td>
<td>Grade</td>
<td>Salary</td>
<td>...</td>
</tr>
<tr>
<td>1</td>
<td>Dave</td>
<td>37</td>
<td>7</td>
<td>$82,000</td>
</tr>
<tr>
<td>2</td>
<td>...</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>...</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**USER:**

“Give me all employees who make between $80K and $87K”

**SQL API**

Dave, 37, 7, $82,000, ...

**USER:**

“Find the variable that most describes compensation”

**ML API**

Google

Tenure with Company

Gary Bradski, garybradski@gmail.com
“Cognitive SQL” Server

Goals:
- Find machine learning spanning “basis”
- Subject to being fully distributed, fault tolerant.

== Fits Google’s MAP-REDUCE => ML SQL

- Create hardware that will run this well.
- World model moves towards “Cognitive SQL”

Sensor Input Agents

Vision, Audio, Lexical Agents

Planning, Control Decision Agents

Fusion in a World Model
OUTLINE

Substrate: a “basis set”

• Background -- Research to enable AI
• Cognitive or “ML SQL” via MAP-REDUCE
• Substrate:
  – A Machine Learning “Basis Set”:
• Superstructure:
  – Embed in Hierarchy for global ↔ local info flow
• Extension to Nano
**Mini-Tutorial: Machine Learning Basis**

**Function Fitting**

- **INPUT** → \( f \) → **OUTPUT**

- Learn a model/function
- That maps input to output
- underfit
- just right
- overfit

**Example Uses of Prediction:**
- Insurance risk prediction
- Parameters that impact yields
- Gene classification by function
- Topics of a document . . .

Find a function that describes given data and predicts unknown data

Gary Bradski, garybradski@gmail.com
Binary Recursive Decision Trees
*Leo Breiman’s “CART”*

At Each Level:
• Find the variable (predictor) and its threshold.
  – That splits the data into 2 groups
  – With maximal purity within each group
• All variables/predictors are considered at every level.

Perfect purity, but…

Gary Bradski, garybradski@gmail.com

*Classification And Regression Tree*
Binary Recursive Decision Trees

Leo Breiman’s “CART”*

- **At Each Level:**
  - Find the variable (predictor) and its threshold.
    - That splits the data into 2 groups
    - With maximal purity within each group
  - All variables/predictors are considered at every level.

Prune to avoid over fitting using complexity cost measure
“Fully Distributed: Random Forests 1

Leo Breiman

Random Forests builds a “forest” of “J” trees, each of which will get one equally weighted vote for classification or clustering.

1) Data:

\[
\begin{align*}
X^1 &= X_1^1, X_2^1, X_3^1, X_4^1 \\
X^2 &= X_1^2, X_2^2, X_3^2, X_4^2 \\
&\quad \vdots \\
X^N &= X_1^N, X_2^N, X_3^N, X_4^N
\end{align*}
\]

2) Build trees all the way down:

3) At each split point:

\[
\begin{align*}
&\text{… only consider Sqrt(M) randomly chosen features} \\
&\quad \text{(here Sqrt(4) = 2)}
\end{align*}
\]

\[
\begin{align*}
X^1 &= X_2^1, X_3^1 \\
X^2 &= X_3^2, X_4^2 \\
&\quad \vdots \\
X^N &= X_1^N, X_3^N
\end{align*}
\]
Random Forests 2

1) A forest of 1.. J trees are built

2) Each Tree learns independently on a subset of the data

\( X^1 = X_1^1, X_2^1, X_3^1, X_4^1 \)

\( X^h = X_1^h, X_2^h, X_3^h, X_4^h \)

\( X^i = X_1^i, X_2^i, X_3^i, X_4^i \)

3) In use, each tree gets one vote

Accuracy by the central limit theorem

Fault tolerant

Distributed

Gary Bradski, garybradski@gmail.com
What we have now …

• Random Forests
  – Fits Google’s MAP-REDUCE parallelization
  – Distributed and fault tolerant
  – Gives us
    • Classification
    • Clustering
    • Capture of distributions in it’s leaves
    • Measure of variable’s importance
Mini-Tutorial: World Model Basis -- Bayesian Networks a.k.a Graphical Models

The graph is directed and acyclic.

A joint distribution, here \( p(B, E, A, J, M) \), is everything we can know about the problem factoring it decreases the number of parameters, here from 16 to 10.

Gary Bradski, garybradski@gmail.com
Computational Advantages of Bayesian Networks

- Bayesian Networks graphically express *conditional independence* of probability distributions.
- Independencies can be exploited for large computational savings.

**EXAMPLE:**
Joint probability of 3 discrete variable (A,B,C) system with 5 possible values each:

\[
P(A,B,C) = 5 \times 5 \times 5 \text{ table:}
\]

- 125 parameters

But a graphical model factors the probabilities taking advantage of the independencies:

- 55 parameters
Causality and Bayesian Nets

One can also think of Bayesian Networks as a “Circuit Diagram” of Probability Models

- The Links indicate causal effect, not direction of information flow.
- Just as we can predict effects of changes on the circuit diagram, we can predict consequences of “operating” on our probability model diagram.
BayesNet Calc: Message Passing

Going down arrow, sum out parent

```
P(v_1) = 0.8, P(\neg v_1) = 0.2
P(v_2|v_1) = 0.4, P(\neg v_2|v_1) = 0.6
P(v_2|\neg v_1) = 0.9, P(\neg v_2|\neg v_1) = 0.1
```

```
P(v_2) = P(v_2|v_1) \pi_{V_2}(v_1) + P(v_2|\neg v_1) \pi_{V_2}(\neg v_1)
P(\neg v_2) = P(\neg v_2|v_1) \pi_{V_2}(v_1) + P(\neg v_2|\neg v_1) \pi_{V_2}(\neg v_1)
```

Specifically:
We have:
```
\pi_{V_2}(v_1) = P(v_1) = 0.8
\pi_{V_2}(\neg v_1) = P(\neg v_1) = 0.2
```

Local computation in V_2:
```
P(v_2) = P(v_2|v_1) \pi_{V_2}(v_1) + P(v_2|\neg v_1) \pi_{V_2}(\neg v_1)
```

Going up arrow, Bayes Law

```
P^*(v_1) = \frac{P(v_2|v_1) P(v_1)}{P(v_2)}
```

\[ P^*(v_1) = \frac{\alpha \lambda_{V_1}(v_1) V_1(v_1)}{P(v_2)} \]

\[ P^*(\neg v_1) = \frac{\alpha \lambda_{V_1}(\neg v_1) V_1(\neg v_1)}{P(v_2)} \]

\[ P(v_2) = P(v_2|v_1) P(v_1) + P(v_2|\neg v_1) P(\neg v_1) \]

with \( P^*(v_1) + P^*(\neg v_1) = 1 = 0.32\alpha + 0.18\alpha \)
\[ \Rightarrow \alpha = 1/P(v_2) = 2 \]

Bayes Law:
\[ P(A|B) = \frac{P(B|A) P(A)}{P(B)} \]

some figures from: Peter Lucas BN lecture course
Message Passing

**Generic Formula: Data Fusion**

\[
P^*(V_i) = P(V_i | E) = \alpha \cdot \text{diagnostic information for } V_i \cdot \text{causal information for } V_i = \alpha \cdot \pi(V_i) \cdot \lambda(V_i)
\]

where:
- \( E = E_{V_i}^+ \cup E_{V_i}^- \): evidence
- \( \alpha \): normalisation constant
- \( \pi(V_i) \): compound causal parameter
- \( \lambda(V_i) \): compound diagnostic parameter

Bayes Law:
\[
P(A | B) = \frac{P(B | A)P(A)}{P(B)}
\]

**Diagnostic message against arrow**

\[
\lambda_{V_i}^V(V_j) = P(V_i | V_j)
\]

**Causal message with arrow**

\[
\pi_{V_i}^V(V_j) = P(V_j)
\]

some figures from: Peter Lucas BN lecture course
Inference in general graphs

- BP is only guaranteed to be correct for trees
- A general graph should be converted to a junction tree, by clustering nodes
- Computationally complexity is exponential in size of the resulting clusters called “cliques” (NP-hard)
Junction tree*: BN → Junction Tree

Algorithm for turning a Bayesian Network with loops into a junction tree

1. “Moralize” the graph by connecting parents
2. Drop the arrows.
3. Triangulate (connect nodes if a loop of >3 exists)
4. Put in intersection variables

* Lauritzen 96
Global message passing, parallel, distributed version:

- Serially we collect to a root node and distribute back out. But …
- All nodes can simultaneously send messages out, if it has received the messages from all its parents
- “March in, March Out”

Stage 1.

Stage 2.
Data Association.
What if we have too many loops?

- It’s combinatoric

In general, tracking N objects gives us N! possible associations.

To do a full Bayesian solution, we must maintain the posterior probabilities $p(A)$ of N! permutation matrices $A$.
Info Filter
Schumitch, Thrun, Bradski, Olukotun NIPS 2005

• Update using log probability (information) is additive (bio trick??)

\[
\begin{pmatrix}
0 \cdots 0 & c & 0 \cdots 0 \\
\vdots & \vdots & \vdots \\
0 \cdots 0 & c & 0 \cdots 0 \\
\vdots & \log \pi & \vdots \\
0 \cdots 0 & c & 0 \cdots 0 \\
\end{pmatrix}
\]

# Data tracks
# Types

\[
\Omega \leftarrow \Omega + \text{with } c = \log \frac{1 - \pi}{N - 1}
\]

• Which, since the info matrix is un-normalized, reduces to an \(O(1)\) update of one (object\(_i\), track\(_i\)) position in the matrix:

\[
\omega_{i,j} \leftarrow \omega_{i,j} + \log \pi - \log \frac{1 - \pi}{N - 1}
\]

• Can use what is effectively Loopy Belief to turn the info matrix

Loopy Belief Propagation: Ignore loops; Message pass in Parallel.
Movies on Northrop Grumman Data

Schum itch, Thrun, Brad ski, Olukotun “Information-Form Data Association Filter” NIPS 2005

Raw Data – Entities ground truth

Uncertainty – Sensor Resolution

~700 entities

Naïve tracker – Red is wrong

Information Filter – Blue is correct

Loopy belief (or Simulated annealing …) can unpack the information matrix.  

Gary Bradski, garybradski@gmail.com
Fault Tolerant: Loopy Belief Propagation

- Approximate inference for large Bayesian/Markov nets
- Iterative message passing algorithm
  - **LBP tolerates message dropping**
- Experiment: Probabilistic Relational Model
  - Uses LBP to solve a graphical model that classifies web pages
- Drop messages randomly and measure impact on accuracy
  - Fraction of incorrect beliefs <= 2%
  - 0 miss-classifications
  - LBP achieves high accuracy even under extreme message dropping
LBP is Redundant

- Many messages are **non-critical**

**Based on Belief Value**

**Based on Classification**

**Intelligent message dropping policies:**

#1: Drop message if sender and receiver’s belief values have not changed

#2: Drop message if receiver’s belief becomes strong

---

Slides: Donald Yeung, Xuanhua Li, and Vida Kianzad
Information Sciences Institute

Gary Bradski, garybradski@gmail.com
Intelligent Message Dropping Performance

- Policy #1: 2.3X speedup
- Policy #2: 3.5X speedup
- Policy #2 exhibits poor numerical convergence
- Policies #1 and #2: 0 miss-classifications

Accuracy-cost tradeoff

LOOPY:
- Distributed (completely ||)
- Fault tolerant (80% drop rate OK)

Stanford CMU MIT Berkeley

Slides: Donald Yeung, Xuanhua Li, and Vida Kianzad
Information Sciences Institute

Gary Bradski, garybradski@gmail.com
The Power of Generative Hierarchies

• By indexing entities in hierarchy
  – We can now generate a “World”:
  – World: $Global \leftrightarrow local$ information flow
Proposal

SUBSTRATE:
• Have distributed, fault tolerant” pieces.
• Fits with “MAP-REDUCE
• Use to capture spatial-temporal distributions
• Claim: These form a “basis function” for ML space

SUPERSTRUCTURE:
• Embed in hierarchical structure global ⇔ local influence
  – Use Inference for
    • Recognition
    • Generative simulation
  – Models are either a tree, or use form of Loopy
OUTLINE

Superstructure

- Background -- Research to enable AI
- Cognitive or “ML SQL” via MAP-REDUCE
- Substrate:
  - A Machine Learning “Basis Set”:
- Superstructure:
  - Embed in Hierarchy for global ↔ local info flow
- Extension to Nano
Your world model:
Local computation fails without hierarchical match
Robots Naturally Use World Models as we did in the Stanford Racing Robot

World model live, used to plan the path …

Robot  Robot’s world model

… fusing a network of sensors for action.
Our Own World Model has a Strange Geometry *

Same size things get smaller, we hardly notice…

Parallel lines meet at a point…

Works on a melon rind

The world is in your head

World Model Geometry for Biological Perception

Perception must be mapped to a space variant grid

Logarithmic in nature

Logarithmic in all directions

This is your beautiful house
Again: World model, global effects on local can be non-intuitive.

**Diffused global properties like lighting best communicated in hierarchy**

Which square is darker?
Biology Constructs World Models Hierarchically

The following sample of technologies do the same:

2. Yann LeCun, Convolutional Neural Networks for handwriting recognition are found at http://yann.lecun.com/exdb/fenet/index.html
11. Redwood Neuroscience Institute http://www.rni.org/
18. Pietro Perona’s vision group works on hierarchical feature models of objects http://www.vision.caltech.edu/html-files/publications.html
22. Peter Bock, developer of ALISA and PI of ALISA group http://www.seas.gwu.edu/~pbock/
Hierarchical Convolutional Neural Networks

Yann LeCun

Broke all the HIPs code (Human Interaction Proofs) from Yahoo, MSN, E-Bay …
Hierarchical

Shimon Ullman

Fragment Based Hierarchy

• Top down and bottom up hierarchy

http://www.wisdom.weizmann.ac.il/~vision/research.html
See also Perona’s group work on hierarchical feature models of objects http://www.vision.caltech.edu/html-files/publications.html
Hierarchical

HMAX

Maximilian Riesenhuber and Tomaso Poggio

In object recognition hierarchy

Basic building blocks

Modulated by attention
Hierarchical

Joijic and Frey

- Scene description as hierarchy of sprites
Hierarchical spatial-temporal memory

- Modular hierarchical spatial-temporal memory

Hierarchy

Module

Layer 3 cells store sequences under a context and generate inferences by combining feedback with input according to equation 6.

Layer 5-6 cells combine decisions based on current input with feedback from higher levels to make predictions according to equation 3.
Peter Bock’s ALISA
An explicit hierarchical Cognitive Model

Figure 1: The ALISA symbolic cognitive hierarchy

Gary Bradski, garybradski@gmail.com
Hierarchical

ALISA Sensor Fusion and Geometry Classification
Hierarchical

ALISA Labeling 2 Scenes

- Sky
- Water
- Evergreen
- Grass
ALISA Labeling Parts/whole
From just a few examples
Recap

• **Substrate**
  – Showed distributed, fault tolerant methods
    • Basis for classification, clustering, representing distributions..

• **Superstructure**
  – Embed in hierarchy to propagate global ↔ local info

The world is simulated in its global context
OUTLINE

Nano Electronics Considerations

• Background -- Research to enable AI
• Cognitive or “ML SQL” via MAP-REDUCE
• Substrate:
  – A Machine Learning “Basis Set”:
• Superstructure:
  – Embed in Hierarchy for global ↔ local info flow
• Extension to Nano
Example Idea: Uses *distributed state, random connections, competition and noise inverse to match*

**TESMECOR** (Temporal Episodic and Semantic Memory using Combinatorial Representations),
Gerard J. Rinkus “A Neural Model of Episodic and Semantic Spatiotemporal Memory” CogSci 2004 (rinkus@comcast.net)

Architecture randomly selects winners, mismatch generates noise

Pattern A presents at Layer 1
F-vector starts

- L1 cells send activation up (forward) to L2 cells.
- We’re assuming that this is a brand new network, i.e., all weights = 0

**NOTE:** For purposes of keeping the figures readable we divide L2 into two zones, left right. The three CMs on the left will hold the internal representation of the L1 pattern at $T = 1$. The three on the right will hold the representation for $T = 2$. In the model’s actual operation, ALL CMs are used on EVERY time slice.
There is complete mismatch between expected and actual input (really, there is no expected input on this very first time slice of the very first episode experienced.

Winners are selected at random and weights increased, forming an IR, IR_{IA}, for pattern A.

**NOTE:** The notation, “[A],” as in IR_{IA}, means that this is an episode-initial instance of state A, i.e., it is state A occurring on the first time slice of an episode.
• At $T = 2$, there is again complete mismatch at the CMs for $T = 2$.
• Again, this is because the weight matrices, both the feedforward (F-matrix) and the horizontal (H-matrix) are naïve (all weights = 0).

NOTE: The gray shading of the cells of $IR_{IA}$ indicates that their activation level is fading since they were active at $T = 1$ and now we are at $T = 2$.

Context by transients:
Temporal decay
• Winners are chosen at random, forming a new IR, $\text{IR}_{AB}$, for pattern B. The horizontal and forward weights shown (depicted with solid lines) are increased.
• Now on another trial we present pattern A again at T = 1.
• This time, there is a perfect match at Layer 2 (see note below). Therefore, the same IR that was learned on the first trial, IR_{IA}, is reactivated.

Note: The model can detect that it is a perfect match at Layer 2 in this case because we assume that the amount of activity (the total number of cells active at L1 on any time slice) is normalized. So, in this case for example, if an L2 cell knows it is receiving four large inputs and if it knows that the model’s parameters are set so that L1 patterns will always have exactly four active cells, it then knows that it is in a perfect match condition.
• But now, an unexpected pattern, C, occurs at $T = 2$.
• The cells of $IR_{AB}$ receive the maximum possible support (three out of three), via the H-matrix, from L2 cells that were active at $T = 1$.
• But they only receive 50% (two out of four) of the maximum, via the F-matrix, from currently active L1 cells.
• Note that the grayness of the L2 cells at $T = 2$ is used to connote partial activation due to partial match. As mentioned earlier, we also use grayness to connote decaying activation, e.g., in the L2 cells for $T = 1$. 
• This amount of mismatch leads to a significant amount of noise being added to the winner selection process. Let’s assume that this amount of noise leads, probabilistically, to a new winner being chosen in one out of the three CMs for $T = 2$. We’ve arbitrarily chosen the lower right-hand CM to have the new winner.

• Thus we have a new IR, $\text{IR}_{AC}$, for the novel pattern, C. The newly increased weights are shown as dotted lines.

• Even though the new IR, $\text{IR}_{AC}$, is unique, it does have overlap with $\text{IR}_{AB}$. This overlap reflects the overall spatiotemporal similarity between this event, i.e., ‘state C following state A’, and the previously experienced (and learned) event, ‘state B following state A’. In both cases, state A was the predecessor and state C has two out of four features in common with state B.
• Now, on another trial, a new pattern, D, having only 2 out of 4 features (i.e., active L1 cells) in common with A, presents at L1.
• So only 50% of the total possible input from L1 occurs at the cells of IR_A. Hence, the gray shading.
• For concreteness, assume that that amount of mismatch causes new winners to be chosen in two out of three CMs, resulting in a new IR, IR_{ID}. The dotted lines denote F-weights that are increased.
• Pattern B, a familiar spatial pattern, presents again at $T = 2$. But it is in a novel temporal context; it has never occurred following pattern D before.
• $IR_{AB}$ receives full support (i.e., 4 out of 4) from the F-vector, but only 1 out 3 from the H-vector.
For concreteness, assume that this level of mismatch leads to new winners in 2 out of 3 CMs, resulting in a new IR, IR$_{DB}$, for state B. New learning shown with dotted lines.
In Hierarchy

By:
1. Embedding in a hierarchy
2. Each higher layer stays persistent at longer time scales

Any temporal pattern may be learned.

Longer Temporal Scale
Large spatial extent
Summary

SUBSTRATE:
• A step towards pragmatic AI using
  – learning algorithms that have Google’s MapReduce distributed capability
  – Use them to capture spatial-temporal distributions

SUPERSTRUCTURE:
• Embed in hierarchy for Global ⇔ local

NANO:
• Back to “neural nets”? But predicated on distributed states, convergence under random connections and use of noise.
BACKUP
Jtree Inference

Belief Network

Graphical Transformation

Join Tree Structure

1. Initialization
2. Observation entry

Inconsistent Join Tree

Propagation

Consistent Join Tree

1. Marginalization
2. Normalization

\[ P(V / e) \]

Image from
Cecil Huang

Gary Bradski, garybradski@gmail.com
Junction Tree Properties

An undirected graph whose vertices (clusters are sets of variables with three properties:

1. **Singly connected property** (only one path)
2. **Potential property** (all variables are represented)
3. **Running intersection property** (variable in 2 nodes implies that all nodes on the path have the variable)

\[ p(a, b, c, d, e) = \frac{1}{Z} \psi(a, b, c)\psi(c, d)\psi(c, e) \]
Global message passing: Two pass

- Select one clique as the root
- Two pass message passing: first collect evidence, then distribute evidence.
Junction Tree Calculation Details 1

- On each edge of the clique tree, we place a potential $\phi_S$ over the variables in the intersection of the two adjacent cliques it joins.
- These intersections are called separator sets and are themselves cliques (fully connected in the underlying graph) although of course they are no longer maximal.
- Now our representation of the joint probability is defined as:

\[ p(X) = \frac{\prod_C \psi_C(x_C)}{\prod_S \phi_S(x_S)} \]

where the normalizer is absorbed into a special separator $\phi_\emptyset$.
Junction Tree Calculation Details 2

• Potential
  – \(U\), the space of \(U\) (subset of the set of all nodes/vertices \(V\)) is the Cartesian product of the state sets of the nodes of \(U\)
  – A discrete potential on \(U\) is a mapping from \(U\) to the non-negative real numbers \(R_0\).
  – Each clique and separator in the junction tree has a potential (actually marginalized joint distribution on the nodes in the clique/separator)

• Propagation/message passing between two adjacent cliques \(C_1, C_2\) (\(S_0\) is their separator)
  – Marginalize \(C_1\)’s potential to get new potential for \(S_0\)
    \[
    \phi^{*}_{S_0} = \sum_{C_1 \backslash S_0} \phi_{C_1}
    \]
  – Update \(C_2\)’s potential
    \[
    \phi^{*}_{C_2} = \frac{\phi^{*}_{S_0}}{\phi_{S_0}}
    \]
  – Update \(S_0\)’s potential to its new potential
Message Passing General

• BayesNet forms a tree
  – Pearl’s algorithm is Message Passing first out and then back in from a given node
• Not a tree (has loops)
  – Turn loops into cliques until net is a tree, then use Pearl’s algorithm
• Cliques turn out to be too big
  – Exact computation is exponential in size of largest cliques
  – Use approximation algorithms (many)
What are Trees Good For?

Learn trees of every variable in terms of every others. Find which ones are important.

Dependency Networks == “Collaborative Filtering”
Link everything to everything else by importance of connection

Amazon.com Uses this to determine product preferences

Gary Bradski, garybradski@gmail.com
Machine Learning Libraries

Bkgnd Goal: ~Enable AI

http://www.intel.com/technology/computing/pnl/

Modeless

Unsupervised

Supervised

Key:
- Optimized
- Implemented
- Not implemented

- K-NN
- Radial Basis
  - Assoc. Net.
  - Random Forests
  - Adaptive Filters
- PCA
- ART
- Dependency Nets
- Spectral clustering
- Agglomerative clustering
- K-means

- SVM
- Multi-Layer Perceptron
  - Logistic Regression
- MART
- CART
- Decision trees
- Boosted decision trees
- HMM
- BayesNets: Classification
- Naïve Bayes
- Spectral clustering
- BayesNets: Parameter fitting
- Inference
- Tree distributions
- Gaussian Fitting

Statistical Learning Library:
MLL

Bayesian Networks Library:
PNL

Physical Models
Influence diagrams
Kalman Filter
Diagnostic Bayesnet
The Vision Sensors Project Road Segmentations into the Model

Segmentation Raw Image

Projection to world model
Fuse Information Into a Generative World Model that Simulates the System Itself

* From Steve Lehar: [http://cns-alumni.bu.edu/~slehar/Lehar.html](http://cns-alumni.bu.edu/~slehar/Lehar.html)

Gary Bradski, garybradski@gmail.com