Previously we defined probability distribution of form \( p(x) = \frac{1}{Z} e^{-\frac{d}{2} \phi(x)} \) exponential distribution.

We also briefly consider mixture distributions. These contain hidden variables:

**E.6. Mixture of Gaussians.**

\[
\begin{align*}
P(x | \mu_1, \sigma_1) &= \frac{1}{\sqrt{2\pi\sigma_1}} e^{-\frac{1}{2} (x-\mu_1)^2 / \sigma_1^2} \quad \text{model 1} \\
P(x | \mu_2, \sigma_2) &= \frac{1}{\sqrt{2\pi\sigma_2}} e^{-\frac{1}{2} (x-\mu_2)^2 / \sigma_2^2} \quad \text{model 2}
\end{align*}
\]

\( P_1 \) - data is generated by model 1
\( P_2 \) - data is generated by model 2, \( P_1, P_2 = 1 \).

\[
P(x) = P(x | \mu_1, \sigma_1) P_1 + P(x | \mu_2, \sigma_2) P_2
\]

Alternatively, define a hidden variable \( h \) indicating whether the data comes from model 1 or model 2.

\[
P(x | h) = \begin{cases} P(x | \mu_1, \sigma_1) & h = 0, \\ P(x | \mu_2, \sigma_2) & h = 1 \end{cases}
\]

\[
P(h) = P_1^h P_2^{1-h}
\]

\[
p(x | h) p(h) = P_1^h P_2^{1-h} \begin{cases} P(x | \mu_1, \sigma_1) & h = 0, \\ P(x | \mu_2, \sigma_2) & h = 1 \end{cases}
\]

formally, we can rewrite the distribution as

\[
p(x | h) p(h) = \frac{1}{Z} e^{-\frac{2}{2} \phi(x)} \quad \text{for } p(x)
\]

mixture \( p(x) \) graphically

Formally, we can rewrite the distribution as

\[
p(x | h) p(h) = \frac{1}{Z} e^{-\frac{2}{2} \phi(x)} \quad \text{for } p(x)
\]
How to extend this?  
- Define probability distribution on graphs. I'll describe models of this type used for vision processing  
  - these models can be used to describe objects (e.g., horses or images)

Similar models are used in speech recognition systems, modeling natural language, genomics, etc.

Graph \( G = (V, E) \), \( V \) set of nodes, \( E \) set of edges

Visually - two nodes \( \mu, \nu \in V \) are connected only if \( (\mu, \nu) \in E \).

\( \text{parent-child relationship} \)

\( \text{ch(v)} \) denotes the children of node \( v \).

Require graph to be separable so that

\( \text{ch(0)} \cap \text{ch(\mu)} = \emptyset \) (empty set)

Edges defined by the parent-child relationships

\( E.G. \)

Different graphs

Each node of the graph is assigned a state variable \( \langle Z_v : v \in V \rangle \).

Example: Leaf nodes correspond to edges in an image.

\( Z_v = \langle (x_0, y_0, \theta_0, s) \rangle \)

The next level nodes correspond to combinatorial edges

E.g., knight or

\( (x, y) \) position of center

& mean orientation & scale.
(3) Define a probability distribution over the graph structure:

\[ P \left( \{z_k\} \mid I \right) = \frac{1}{Z} \prod_{k} e^{-\frac{1}{2} \sum_{k} \alpha_k \Phi(\mathbf{z}_k, \mathbf{z}_{ch_k}) + \frac{1}{2} \sum_{\mathbf{i}, \mathbf{j} \in \mathbf{Z}_k} \Phi^{ij}_k (\mathbf{i}, \mathbf{j})} \]

Note: this can be a fixed graph structure, or non-fixed.

For non-fixed, allow two types of nodes:

(i) AND nodes \( \bigcirc \) composition

(ii) OR nodes \( \bigcirc \) logical OR like mixed causality

The combination of AND and OR nodes enables you to model great variability of shapes - e.g., etc.

Alternative types of models:

- Stochastic Context Free Grammars (SCFGs) modeling natural language
- Hidden Markov Models (HMMs) used to model speed
- Probability distribution defined on structured representations - grammars, graphs, etc.
What do you do with a model like this?

Two main tasks:

(i) Inference — given input I estimate the most probable state.

(ii) Learning — learn the parameters (known structure) or the parameters and structure. — harder.

How to do inference?

Good inference required as a pre-cursor to learning.

For models that are separable, the distribution becomes independent on different parts of the tree.

Express the model \( P(z) = \prod_{\nu} \prod_{\lambda} \frac{1}{Z} \phi(z_{\mu}, z_{\lambda}, \nu) \) Gibbs distribution

where \( E[Z] = \sum_{\nu, \lambda} \phi(z_{\mu}, z_{\lambda}, \nu) + \sum_{\nu, \lambda, \mu} \phi(z_{\mu}, z_{\lambda}, \nu, \lambda, \mu) \)

maximizing \( P(z) \) w.r.t. \( Z \) is equivalent to maximizing \( E \).

Exploit the graphical structure to express the energy function recursively.

\[ E_v(z_{\mu}, z_{\nu}, v) = \sum_{\mu} \sum_{\nu} \phi(z_{\mu}, z_{\nu}, v) + \sum_{\nu, \lambda} \phi(z_{\mu}, z_{\nu}, \lambda, \nu) \]

The full energy is obtained by evaluating \( E \) at the root node of the full graph.

Maximize \( E \) by dynamic programming (DP)
\[ F(Z_1, Z_2, Z_3, Z_4, Z_{12}, Z_{23}, Z_{24}) \]

\[ F_1(Z_1, Z_2, Z_{12}) \]

\[ + F_2(Z_3, Z_4, Z_{23}) \]

\[ + F_3(Z_{12}, Z_{23}, Z_{24}) \]

For each \( Z_{12} \), find max \( F_1(Z_1, Z_2, Z_{12}) \)

\text{ wrt. } Z_{12}

Call this \( \tilde{F}_1(Z_{12}) \)

For each \( Z_{23} \), find max \( F_2(Z_3, Z_4, Z_{23}) \)

\text{ wrt. } Z_{23}

Call this \( \tilde{F}_2(Z_{23}) \)

Then find solution by maximizing

\[ \tilde{F}_1(Z_{12}) + \tilde{F}_2(Z_{23}) + F_3(Z_{12}, Z_{23}, Z_{24}) \]

\text{ wrt. } Z_{12}, Z_{23}, Z_{24}

Polynomial time in no. of layers.

---

\[ Z_1 \rightarrow Z_2 \rightarrow Z_{23} \rightarrow Z_{1234} \rightarrow Z_{12345678} \]

**Basic Idea**

→ divide and conquer.

**For vision** → also like

"Constraint satisfaction"

find probable positions for legs of horse

"" ""

"torso of horse"

find consistent position for legs and torso
(6) Learning when the structure is known.

Optim: Training data \( \{ (I^\mu, z^\mu) : \mu = 1, \ldots, N \} \), complete knowledge.

\[
P(I, z | I^\mu) = \frac{1}{Z[I^\mu]} \exp \left( \alpha \cdot \phi(z) + \sum_{\mu'} \phi^\mu(I, D^\mu) \right)
\]

ML estimation: \( \hat{\alpha}_N = \arg \max \sum_{\mu} P(I^\mu, z^\mu | I). \)

This can be very difficult. (More sampling method)

- Summing over \( z \) is possible (by dynamic programming)
  but very difficult over images \( I \). (Don't think anybody knows how to do this)

An alternative:

Express the distribution as a probabilistic discriminative model:

\[
P(z | I, \alpha) = \frac{1}{Z[I, \alpha]} \exp \left( \alpha \cdot \phi(z) + \sum_{\mu'} \phi^\mu(I, D^\mu) \right)
\]

Use a discriminative method to learn \( \alpha \)

Which does not require knowledge of \( Z[I, \alpha] \).

Intuition: Find parameters \( \alpha \) so that

\[P(z | I^\mu, \alpha) \] is large for values of \( z \) which are close to the true value \( z^\mu \)

for \( \mu = 1, \ldots, N \).
(7) Structure Perceptron Algorithm:

1. Iterate over training data. $n = 1 .. N$
2. Use update to calculate $z^n = \arg \max (\alpha \cdot \phi(z) + \lambda \cdot \phi(I, D \theta))$
3. Update $\alpha^{n+1} = \alpha^n + \lambda \cdot (\phi(I^n, z^n) - \phi(I^n, z^n))$

Repeat

- Convergent, no guarantee of convergence, but works surprisingly well.

Alternative: Structure Max-Margin (generalization of SVM)

Formulate as a primal-dual optimization problem.

Define $\Delta(z^n, z^0) = 1, \text{if } |z^n - z^0| > \tau$

$= 0, \text{otherwise}$

Error measure $L(z^n, z) = \sum_{n \in V} \Delta(z^n, z^0)$

Form weights to minimize the criterion. Slack variables:

$\lambda \|d\|^2 + \lambda \|d\|^2 < C \sum_{n \in V} \xi^n$

s.t. $\langle d, \phi(z^n) + \lambda \cdot \phi(I, D \theta) \rangle$

like margin constraints

Reduce to a dual problem.

Can be solved but much more computation required than for standard SVM.
(2) What about learning the structure?

Sketch of recent work.

- Exploit the compositional structure.
  - Edges only \(-\) quantize to few possible values

Images containing object and varied background.

- Many edges \(-\) some due to object, some due to background.

Search image subregion to find triplet patterns that occur frequently and have little variation. e.g. \(\triangle\)

\(\Rightarrow\) Call these suspicious coincidences.

- Candidates for the bottom level structure of the graph
  - Highly correlated structures may be due to a hidden cause

Assemble a level 1 dictionary of structures that occur frequently, plus their variations.

\(\triangledown\) \(\downarrow\) \(\downarrow\) \(\downarrow\)

- Note: search for triplet combinations of these level 1 dictionary elements that frequently occur.

- Form dictionary at level 2

Continue with levels until you don't find any structures that frequently occur.

Result: Horse model \(\bigcirc\) biggest repetitive structure (highest stage) in the middle

Exploits compositional structure - few correlated substructures

- Find correlated subparts - and propose them as parts.
Overview:
Probability distribution on structured representations:
- graphs, grammars.

Requiring inference algorithms (Dynamic Programming in) the example

Requiring learning if structure is known:
- structure perception,
- structure max-margin.
- EM - difficult because of partition function.

Structured learning possible in some cases:
- exploit compositionality,
  exploit correlation,
  suspicious coincidences, complementary exclusion.

Other approaches to structured learning:
- analogy.
  - if you've learned the structure of a horse, then you can guess the structure of a yak.