LECTURE 14

Kevin Murphy

3 November 2004
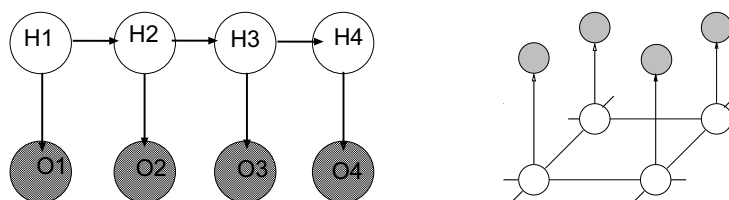
- Generative model defines $P(h, o) = P(h)P(o|h)$.

  - HMM (hidden Markov model)

  $$P(h, o) = [\prod_t P(h_t|h_{t-1})][\prod_t p(o_t|h_t)]$$

  - MRF (Markov random field)

  $$P(h, o) = [\frac{1}{Z} \prod_i \prod_{j \in N_i} \psi_{ij}(h_i, h_j)] \prod_i P(o_i|h_i)$$
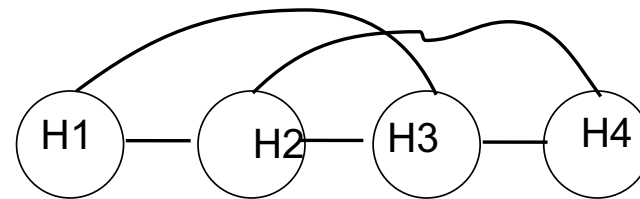

- Conditional model defines $P(h|o)$.

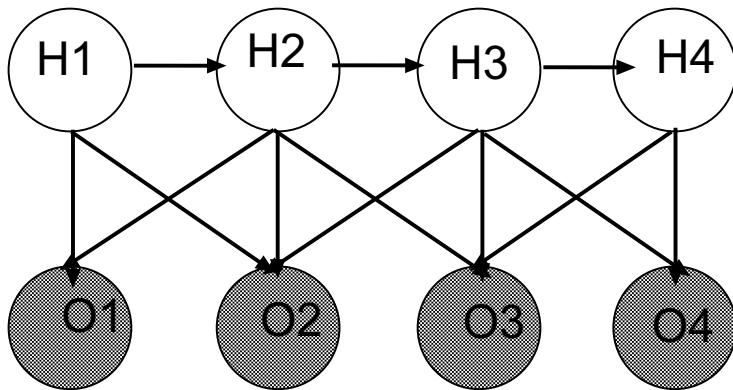 - CRF (conditional random field)

  $$P(h|o) = \frac{1}{Z(o)} \prod_i \prod_{j \in N_i} \psi_{ij}(h_i, h_j, o)$$

- Do not need to waste parameters modeling observed inputs o.

- Can use supervised machine learning methods to learn local evidence $P(h_i|o)$.

- Can incorporate arbitrary, nonlocal features of the input, without increasing complexity of inference.

- An MRF is

$$P(h) = \frac{1}{Z} \prod_c \psi_c(h_c)$$

- The clique potentials are often defined in terms of feature vectors

$$\psi_c(h_c) = \exp\left(\theta_c^T f_c(h_c)\right)$$

- By using indicator features, we can recover tabular potentials:

$$f_c(h_c) = [\delta(h_c, 1), \ldots, \delta(h_c, K)]$$

- A CRF is

$$P(h|v) = \frac{1}{Z(v)} \prod_c \psi_c(h_c, v)$$

- The (input-dependent) clique potentials are often defined in terms of feature vectors

$$\psi_c(h_c, v) = \exp\left(\theta_c^T f_c(h_c, v)\right)$$

- Example: logistic regression. Hidden node $H \in \{-1, +1\}$, cliques = edges = $\{(H, v_c)\}$:

$$P(h|v) = \frac{1}{Z(v)} \prod_c e^{\theta_c v_c h}$$

- Example application: entity extraction from text (logistic regression with correlation amongst the hidden labels/ discriminative version of an HMM)

ENTITY EXTRACTION

Closed set

U.S. states

He was born in Alabama…

The big Wyoming sky…

Regular set

U.S. phone numbers

Phone: (413) 545-1323

The CALD main office can be reached at 412-268-1299

Complex pattern

U.S. postal addresses

University of Arkansas
P.O. Box 140
Hope, AR 71802

Headquarters:
1128 Main Street, 4th Floor
Cincinnati, Ohio 45210

Ambiguous patterns, needing context and many sources of evidence

Person names

…was among the six houses sold by Hope Feldman that year.

Pawel Opalinski, Software Engineer at WhizBang Labs.

EXAMPLE: FEATURES USED

Feature	Pattern
Capitalized	Xxxxx
Mixed Caps	XxXxxx
All Caps	XXXXX
Initial Cap	X….
Contains Digit	xxx5
All lowercase	xxxx
Initial	X
Punctuation	.,:;!(), etc
Period	.
Comma	,
Apostrophe	'
Dash	-
Preceded by HTML tag	

Character n-gram classifier says string is a person name (80% accurate)

In stopword list
(the, of, their, etc)

In honorific list
(Mr, Mrs, Dr, Sen, etc)

In person suffix list
(Jr, Sr, PhD, etc)

In name particle list
(de, la, van, der, etc)

In Census lastname list;
segmented by P(name)

In Census firstname list;
segmented by P(name)

In locations lists
(states, cities, countries)

In company name list
("J. C. Penny")

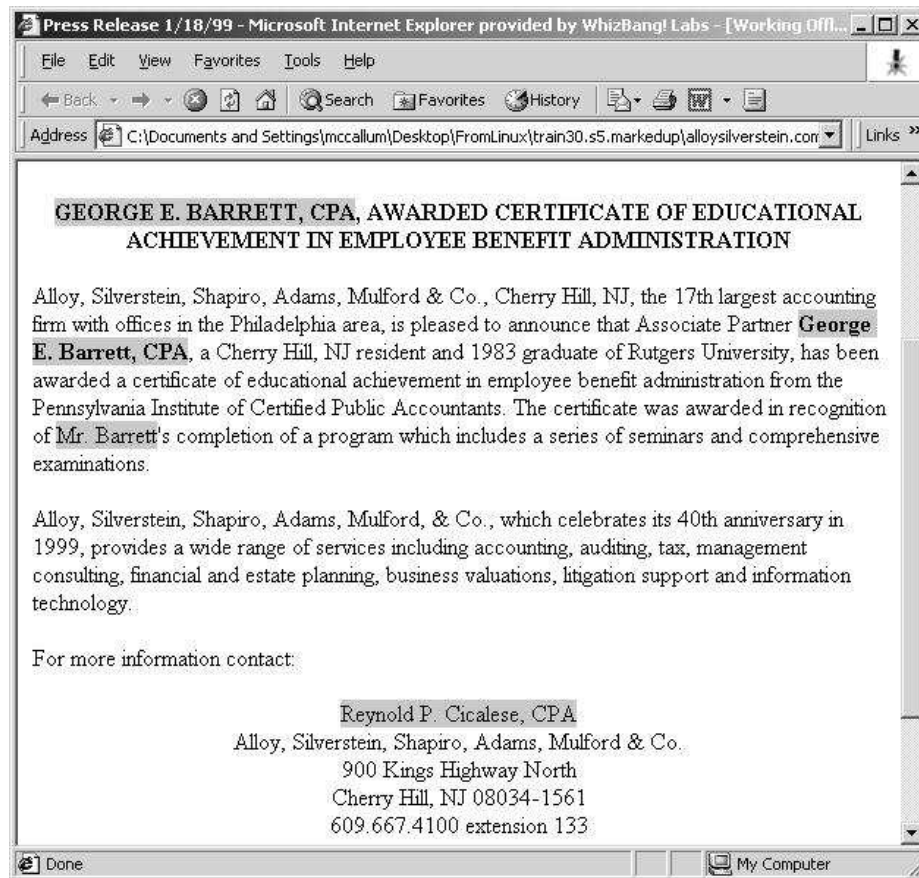In list of company suffixes
(Inc, & Associates, Foundation)

Hand-built FSM person-name extractor says yes, (prec/recall ~ 30/95)

Conjunctions of all previous feature pairs, evaluated at the current time step.

Conjunctions of all previous feature pairs, evaluated at current step and one step ahead.

All previous features, evaluated two steps ahead.

All previous features, evaluated one step behind.

Total number of features = ~200k

Mallet Software

Person name Extraction   *[McCallum 2001, unpublished]*

- MaxEnt models are a generalized version of exponential family models, and so they can be thought of as generative models which assign probability distribution to joint settings of the features $f_i(\mathbf{x})$.

- But they *are not* generative models of the original inputs $\mathbf{x}$, because the features may be very complicated, nonlinear functions.

- Futhermore, it may be possible to generate joint feature settings which do not correspond to *any* possible input $\mathbf{x}$.

- For example, what if our generative model of English spelling gives $f_{\texttt{ing}}(c_1, c_2, c_3) = 1$ and $f_{?\texttt{ed}}(c_1, c_2, c_3) = 1$?

Parameter learning in MRFs/CRFs

- A CRF is $P(h|v) = \frac{1}{Z(v)} \prod_c \psi_c(h_c, v)$

- The (input-dependent) clique potentials are often defined in terms of feature vectors $\psi_c(h_c, v) = \exp\left(\theta_c^T f_c(h_c, v)\right)$

- Assume fully labeled data, (h^m, v^m) pairs. Log-likelihood is

$$\ell = \sum_m \log P(h^m | v^m) = \sum_m \sum_c \theta_c^T f_c(h_c^m, v^m) - \log Z(v^m)$$

  where $Z(v^m) = \sum_h \prod_c \psi_c(h_c, v^m)$.

- Derivative of log-likelihood is

$$\frac{\partial \ell}{\partial \theta_c} = \sum_m f_c(h_c^m, v^m) - \sum_{h_c} P(h_c|v^m) f_c(h_c, v^m)$$

$$= \text{counts} - \text{expected counts}$$

$$Z_{v,\theta} = \sum_h \exp\left(\sum_c \theta_c^T f_c(h_c, v)\right)$$

$$\frac{\partial \log Z_{v,\theta}}{\partial \theta_c} = \frac{1}{Z_{v,\theta}} \frac{\partial}{\partial \theta_c} \sum_h \exp\left(\sum_c \theta_c^T f_c(h_c, v)\right)$$

$$= \frac{1}{Z_{v,\theta}} \sum_h \frac{\partial}{\partial \theta_c} \exp\left(\sum_c \theta_c^T f_c(h_c, v)\right)$$

$$= \frac{1}{Z_{v,\theta}} \sum_h \exp\left(\sum_c \theta_c^T f_c(h_c, v)\right) f_c(h_c, v)$$

$$= \sum_{h_{c'}} \sum_{h_c} P(h_{c'}, h_c | v, \theta) f_c(h_c, v)$$

$$= E_{h_c} f_c(h_c, v)$$

- Frequently we assumed tied parameters, to handle models of variable-size e.g., sequences of varying length, images of different size, web-data with multiple web-pages

$$\psi_c(h_c, v) = \exp(\theta^T f_c(h_c, v))$$

- In this case, we just sum the (expected) features over all cliques that share the same weight

$$\frac{\partial \ell}{\partial \theta_c} = \sum_m \left[\sum_c f_c(h_c^m, v^m) \right] - \left[\sum_c \sum_{h_c} P(h_c | v^m) f_c(h_c, v^m) \right]$$

- We can associate a weight with each type (class) of clique, to specify the tying pattern. This is called a *relational Markov network*.

- We usually put a $\mathcal{N}(\theta; 0, \sigma^2 I)$ prior on the weights to do "soft" feature selection.

- The penalized log-likelihood is

$$\ell = \sum_m \sum_c \theta_c^T f_c(h_c^m, v^m) - \log Z(v^m) - \frac{\theta^T \theta}{2\sigma^2} + C$$

- Derivative of penalized log-likelihood is

$$\frac{\partial \ell}{\partial \theta_c} = \sum_m f_c(h_c^m, v^m) - \sum_{h_c} P(h_c|v^m) f_c(h_c, v^m) - \frac{\theta}{\sigma^2}$$

- The prior variance σ^2 is usually set by cross-validation.

- Derivative of penalized log-likelihood is

$$\frac{\partial \ell}{\partial \theta_c} = \sum_m f_c(h_c^m, v^m) - \sum_{h_c} P(h_c|v^m) f_c(h_c, v^m) - \frac{\theta}{\sigma^2}$$

- This can be passed to any gradient-based optimizer, e.g., conjugate gradient or BFGS. This will find the global optimum (since fully observed, convex problem).

- There is an alternative method called iterative scaling, but it is slower, more complex and less general.

- Learning requires computing $P(h_c|v^m)$ for every clique c, every training case m, and every iteration of the gradient algorithm, so it can be very slow.

- If inference is intractable, we can approximate $P(h_c|v^m)$.

- If we use loopy belief propagation, one can show (Wainwright, Jaakkola, Willsky, AISTATS 03) that for pairwise tabular MRFs, one possible local optimum is

$$\hat{\theta}_{s,j} = \log \tilde{P}(X_s = j), \quad \hat{\theta}_{st,jk} = \log \frac{\tilde{P}(X_s = j, X_t = k)}{\tilde{P}(X_s = j)\tilde{P}(X_t = k)}$$

  so we can set the parameters from the empirical distribution $\tilde{P}$ without running BP.

- If we run BP with these parameters, one possible fixed point is that the model marginals will match the empirical marginals!

- However, this may not match the behavior of the true MLEs when the local evidence changes.

- For more general models, one can use approximate inference to compute an approximate gradient, but this may not converge.
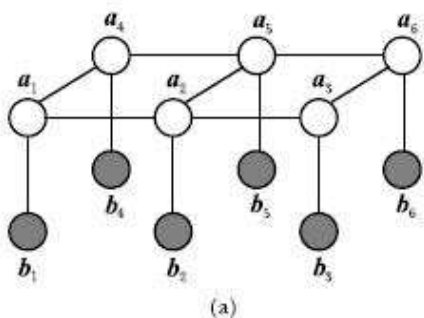
- Instead of doing approximate inference, we can change the objective function:

$$\ell(h^m|v^m) = \frac{1}{Z(v^m)} \prod_i \phi_i(h_i^m, v^m) \prod_{j \in N_i} \psi_{ij}(h_i^m, h_j^m)$$

$$\approx \prod_i \frac{1}{Z_i} \phi_i(h_i^m, v^m) \prod_{j \in N_i} \psi_{ij}(h_i^m, h_j^m)$$

$$= \prod_i P(h_i^m | h_{N_i}^m, v^m)$$

where $Z_i = \sum_{h_i} \phi_i(h_i, v^m) \prod_{j \in N_i} \psi_{ij}(h_i, h_j^m)$

(a)   (b)
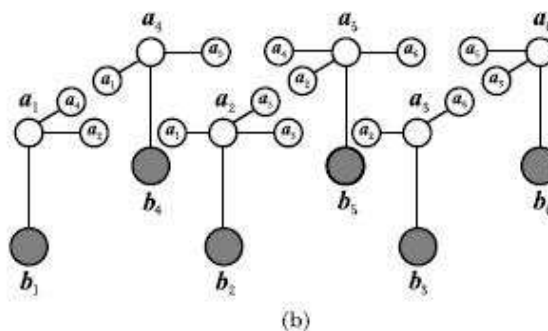
- Pseudo-likelihood learns to trust its hidden neighbors too much (since they are assumed known during learning), hence leading to over-smooth estimates at run time.

$$\ell(h^m|v^m) \approx \prod_i P(h_i^m|h_{N_i}^m, v^m)$$

- One hack is to regularize the pairwise interaction potential ψ_{ij}.

- "Discriminative Fields for Modeling Spatial Dependencies in Natural Images", Kumar & Herbert, NIPS 2003

- Goal: estimate $h_i \in \{-1, +1\}$ at each pixel i.

- Local-evidence defined in terms of features $f_i(v)$:

$$\phi_i(h_i, v; w) = \log \sigma(h_i w^T f_i(v))$$

- Image-dependent smoothing between neighboring labels

$$\psi_{ij}(h_i, h_j, v; \theta) = h_i h_h \theta^T g_{ij}(v)$$

- Inference= graph cuts, Learning= pseudo-likelihood
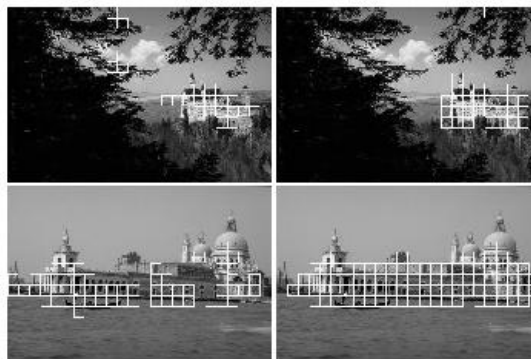
- We assumed $\psi_c(x_c) = \exp \theta_c^T f_c(x_c)$.

- Where do the features come from?

- McCallum (UAI 03) suggested a greedy feature induction scheme for 1D CRFs applied to text:

  - At each iteration, consider (in parallel) adding new atomic features (binary tests on the input) and conjunctions of existing features.
  - Evaluate quality of proposed candidates using the change in pseudo-likelihood.
  - Having chosen a set of features, add them and refit the weights using BFGS.
  - It learned features like $f_i(v) = \delta(v_i =' the', v_{i+1} =' of')$.

- Dietterich et al (ICML 04) suggested using boosting to solve a similar task.

- Let v be visible, h be always hidden, and s be desired output state (observed in training).

$$P(s^m|v^m) = \sum_h P(s^m, h|v^m)$$

$$= \sum_h \frac{1}{Z(v^m)} e^{\Psi(s^m, h, v^m)}$$

$$= \frac{\sum_h e^{\Psi(s^m, h, v^m)}}{\sum_h \sum_s e^{\Psi(s, h, v^m)}} \overset{\text{def}}{=} \frac{Z(s^m, v^m)}{Z(v^m)}$$

where

$$\Psi(s, h, v) = \sum_c \theta_c^T f_c(s_c, h_c, v)$$

- Log-likelihood

$$\log P(s^m|v^m) = \log Z(s^m, v^m) - \log Z(v^m)$$

- Derivative is

$$\frac{\partial \log P(s^m|v^m)}{\partial \theta_c} = E_h f_c(s^m, h, v^m) - E_h E_s f_c(s, h, v)$$

- Or we can use EM.

  - E-step: compute expected sufficient statistics.

  - M-step: maximize expected complete-data log-likelihood using standard techniques for fully observed MRFs (eg IPF or gradient).
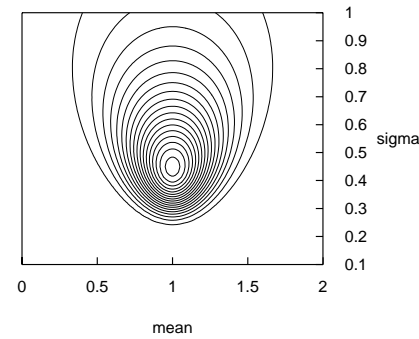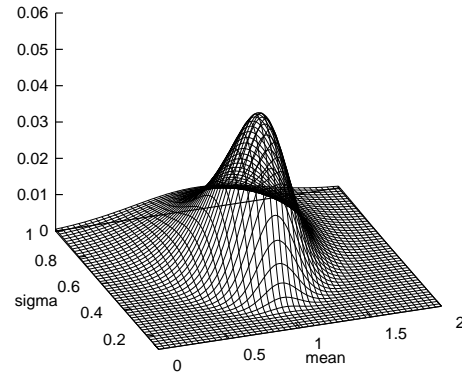
- Log-likelihood $\ell(\mu, \sigma) = \sum_n \log P(x_n | \mu, \sigma)$
- MLE for mean: $\hat{\mu}_{ML} = \frac{1}{N} \sum_n x_n$
- MLE for variance: $\hat{\sigma^2}_{ML} = \frac{1}{N} S$, where $S = \sum_n (x_n - \overline{x})^2$
- $\hat{\sigma^2}_{ML}$ is biased: $E_{X_{1:n} \sim \mathcal{N}(\mu, \sigma^2)} \hat{\sigma^2}_{ML}(X_{1:n}) = \frac{N-1}{N} \sigma^2$
- So we use $\hat{\sigma}^2_{N-1} = \frac{1}{N-1} S$
- Unbiased is not enough, e.g $E_{X_{1:n} \sim \mathcal{N}(\mu, \sigma^2)} \tilde{\mu}(X_{1:n}) = E X_1 = \mu$
- Also need consistency: $E(\hat{\theta} - \theta)^2 \to 0$.
  $\hat{\mu}$, $\hat{\sigma}^2_N$ and $\hat{\sigma}^2_{N-1}$ are all consistent.
- e.g., for data below, $N = 5$, $\overline{x} = 1.0$, $S = 1.0$, $\sigma_N = 1/\sqrt{(5)} = 0.45$, $\sigma_{N-1} = 1/\sqrt{(4)} = 0.5$.
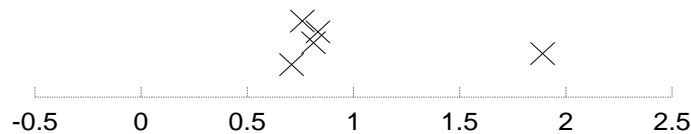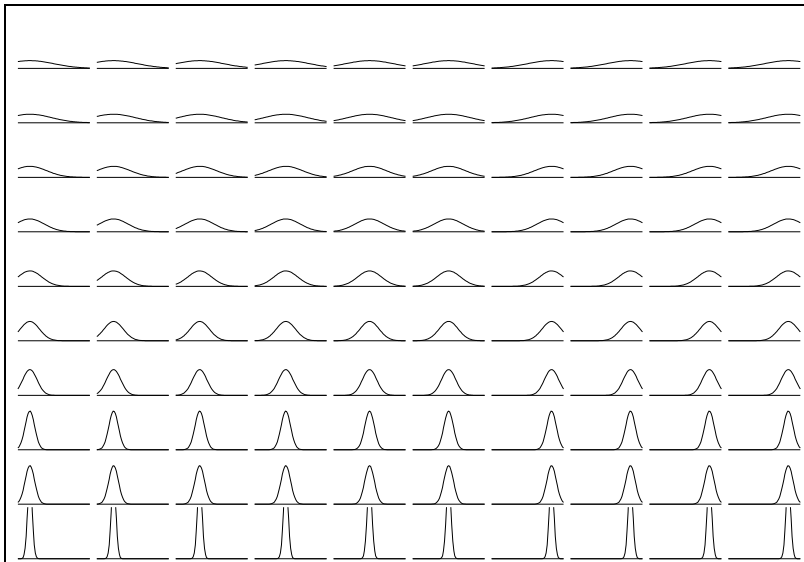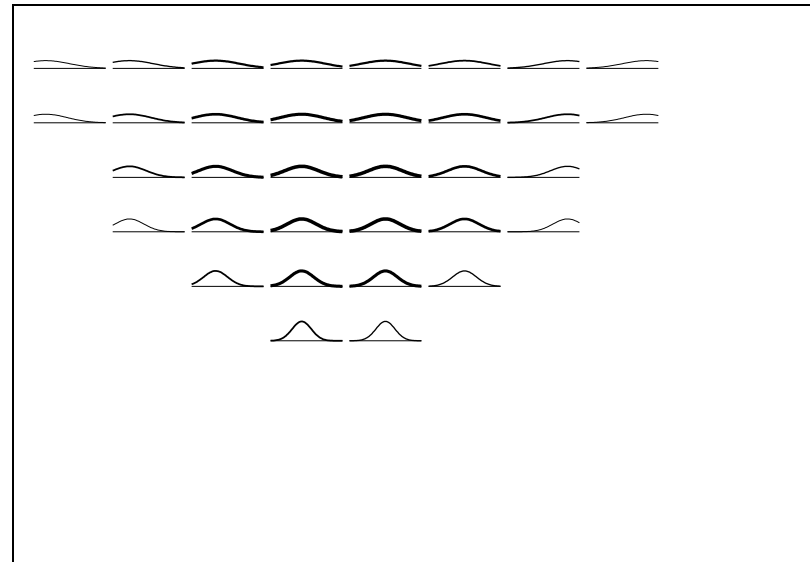
We plot likelihood $\ell(\mu, \sigma)$ for σ vs μ.
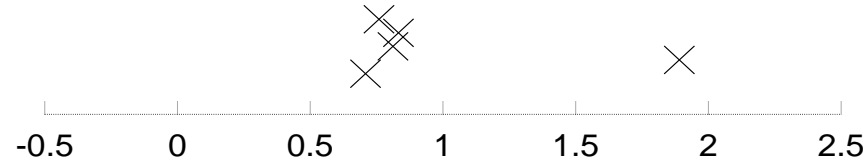
Prior

Posterior

Maybe this data would be better fit by

$$p(x|\mu_1, \sigma_1, \pi_1, \mu_2, \sigma_2, \pi_2) =$$

$$= \frac{\pi_1}{\sqrt{(2\pi)}\sigma_1} \exp(-\frac{(x-\mu_1)^2}{2\sigma_1^2}) + \frac{\pi_2}{\sqrt{(2\pi)}\sigma_2} \exp(-\frac{(x-\mu_2)^2}{2\sigma_2^2})$$
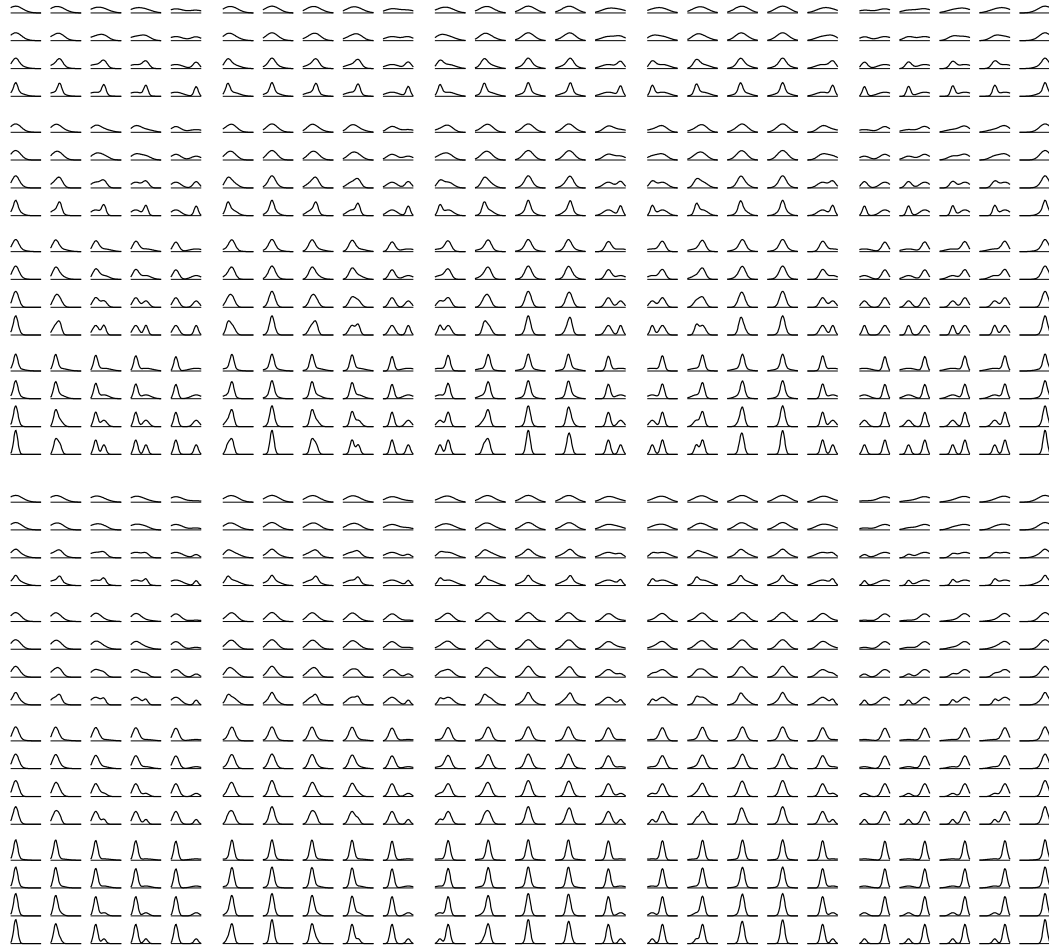
where $\pi_1 + \pi_2 = 1$.

PRIOR HYPOTHESIS SPACE FOR MOG2
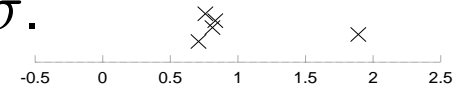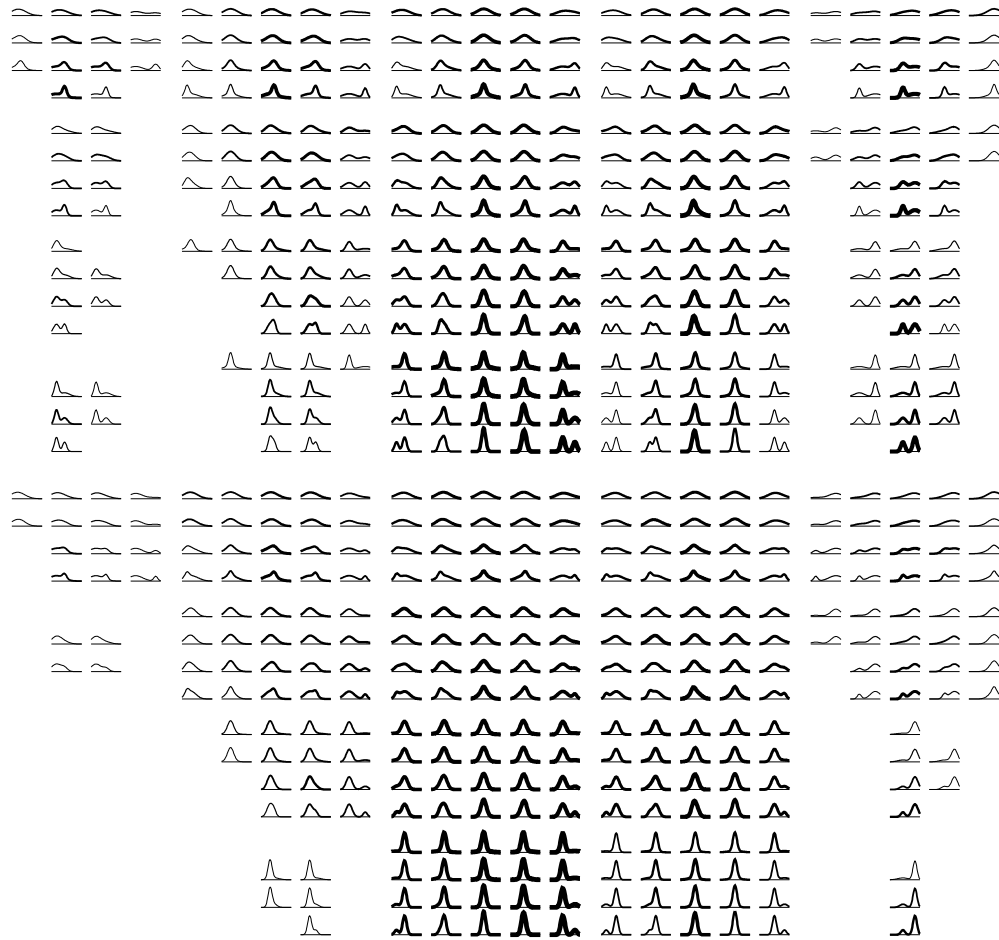
Top-half: $\pi_1 = 0.6$, bottom-half $\pi_1 = 0.8$. We plot μ vs σ.

POSTERIOR HYPOTHESIS SPACE FOR MOG2

Top-half: $\pi_1 = 0.6$, bottom-half $\pi_1 = 0.8$. We plot μ vs σ.

MODEL SELECTION

- Maximum likelihood always picks the most complex model.

- Instead we should pick the most probable model $P(M|D) \propto P(D|M)$, where $P(D|M)$ is the *marginal* likelihood:

$$P(D|M) = \int_{\mu,\sigma} P(D|\mu, \sigma, M)P(\mu, \sigma|M)$$

- The integral over parameters penalizes overly complex models (Bayesian Occam's razor).

- This can be used for model selection (Bayesian version of hypothesis testing).

- Examples of model selection: number of clusters in K-means, order in K-th order Markov model, structure learning...