PMTK: Probabilistic Modeling Toolkit

Kevin Murphy
Matt Dunham

Dept. Computer Science
Univ. British Columbia
Canada

Presented at NIPS workshop on “Probabilistic programming languages”, Dec 2008

Outline

- Why yet another toolkit?
- What is PMTK?
- Examples

Why?

- I needed software for my forthcoming *undergraduate* textbook ("Machine learning: a probabilistic approach", MIT Press 2010)
- Book emphasizes *simple*, but widely used, probabilistic models and algorithms (linear and logistic regression, mixture models, HMMs, CRFs / Newton's method, stochastic gradient, EM, Gibbs sampling, etc)
- Want *readable*, but reasonably efficient, source code implementations of these models/ algorithms
- Existing toolkits inadequate
  - ML toolkits often not probabilistic
  - GM toolkits often not discriminative
  - Bayesian toolkits often not efficient

Generic ML toolkits

Name	Functionality	Language
PMTK	Probabilistic supervised learning (including kernel preprocessing), unsupervised density modeling	Matlab
Weka	Various supervised methods (dtrees, boosting, NN)	Java
Spider	Kernel-based supervised methods	Matlab
Netlab	NNs, mixtures, GPs	Matlab
Torch	NNs, mixtures, SVMs, HMMs, etc	C++
MLtools (Lawrence)	Various, including LLE, GPLVM, etc.	Matlab
Shogun	SVMs	C++

See www.mloss.org

Generic Bayesian toolkits

Name	Functionality	Language
PMTK	Exact conjugate analysis, MH & Gibbs, Dynamic programming	Matlab
(Open)BUGS	Gibbs sampling	Component Pascal
JAGS	Gibbs sampling	Java
HBC (Daume)	Collapsed Gibbs, emphasis on non-parametric Bayes	C?
Infer.net (Winn&Minka)	EP, VB, Gibbs	closed
Blaise (Bonawitz)	MH	Java
FBM (Neal)	MH for NNs, mixture models, Dirichlet difusion trees	C
VIBES (Winn)	VB	Java

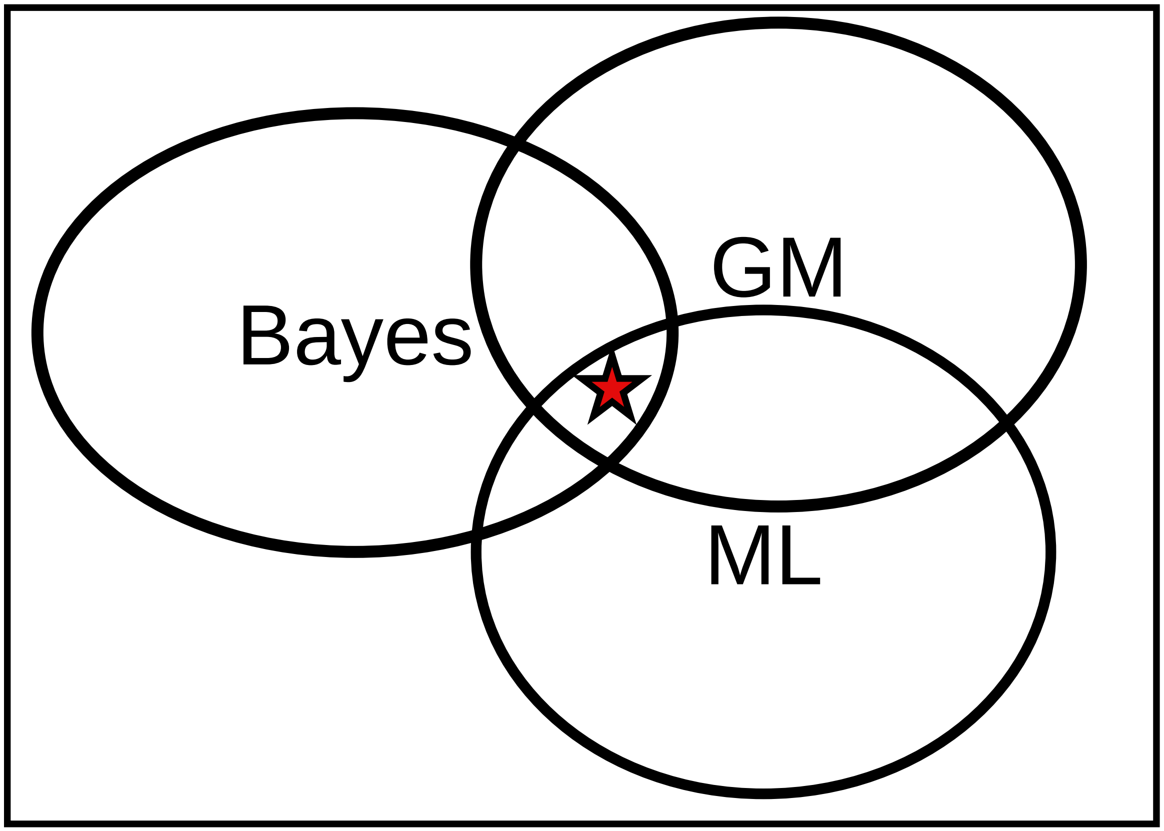
See "**Software for graphical models: a review**", Murphy, ISBA'07

Generic GM toolkits

Name	Functionality	Language
PMTK	DAGs, UGMs (Bayesian parameter & state estimation, model selection)	Matlab
BNT	DAGs (parameter & state estimation, model selection)	Matlab
PNL (Intel)	DAGs, UGMs (parameter & state estimation)	C++
Hugin, Netica	DAGs, parameter & state estimation	$
VIBES, BUGS, infer.net	Hierarchical Bayes	-
GMTK (Bilmes)	DAGs, especially DBNs	(C++)
gR	Wrapper to various R packages	R
Smile/Genie	DAGs and influence diagrams	C++
Alchemy	Markov logic nets	C++?

See "**Software for graphical models: a review**", Murphy, ISBA'07

The holy trinity



Why not BNT?

- BNT (Bayes Net Toolbox) is a very popular* Matlab package that I wrote in grad school
- But it does not support
  - Bayesian parameter estimation
  - Undirected graphical models
  - Non-parametric models (GPs, DPs, kNN, etc)
  - L1 priors
  - Kernels
  - Etc.
- Also
  - It is written in Matlab's old object oriented system; the new version is much better (see later)
  - Various other design flaws

* About 120,000 visits up to 2002..

Lessons learned from BNT

- Packages in interpreted languages are much easier to use
- Matlab source code is fairly easy to read/ change
- Most users are students, although also used by ML researchers
- Most applications just need simple variants on existing models e.g.
  - HMM with robust observation model
  - Mixture models with feature selection
  - Supervised learning with missing data

Outline

- Why yet another toolkit?
- What is PMTK?
- Examples

Design philosophy

- Provide “reference implementation” for a set of widely used models and algorithms.
- Provide lightweight, uniform interface to a large collection of existing code.
- Separate models and algorithms.
- Models = likelihood + prior + transformer.
- Computation is manipulating probability distributions.

Matlab: Pros and Cons

- Well-suited to rapid prototyping (interpreted, dynamically typed)
- Portable
- Succinct syntax
- Large code base
- Popular in NIPS comm.
- Functional / Object Oriented / Imperative
- Excellent plotting / Visualization

- Can be slow for anything other than matrix-vector computations
- Can be expensive
- Not always backwards compatible

May port to Python in the future…

Matlab 2008a's OO system

```
classdef ExampleClass < superclass1 & superclass2
% An example class definition

   properties
      prop1;       % field for storing data within objects of this clas
      prop2;
      prop3;
   end

   methods

       function obj = ExampleClass(varargin)
       % class constructor
            obj.prop1 = 1;
       end

       function obj = operation1(varargin)
       % public class method
       end

       function obj = operation2(varargin)

       end

       function val = get.prop1(obj)
       % this method is called, whenever a user tries to access the prop1
       % property of this class
       end

       function obj = set.prop1(obj)
       % this method is called, whenever the user tries to set the prop1 property
       end
   end

   methods(Access = 'private')
   % private helper methods not part of the class interface go here.
   end
end
```

New Syntax

- Single File Definition
- Abstract Classes
- Visibility Control
- Static Methods
- Handle Classes (can implement pointers, eg shared parameters)
- Events
- Event driven property access
- Operator Overloading
- Meta Classes

Main classes in PMTK

- Probability distributions
- Inference engines
- Transformers

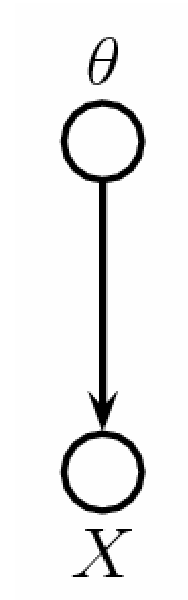
Methods for ProbDist

- Fit: PD * Data -> PD
- Predict: PD * Data * Query -> PD
- Sample: PD * Int -> Data
- Marginal: PD * Query -> PD
- Mean, mode, variance: PD -> real
- Plot: PD -> ()
- Etc.

Simple unconditional distributions

"Unsupervised learning"

Simple



Examples

- •Gaussian
- •Gamma
- •Student
- •Mvn
- •Wishart
- •Bernoulli
- •Beta
- • etc

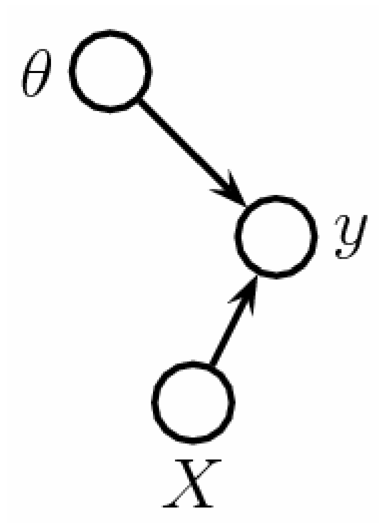
Fit $$\hat{\theta} = \arg\max_{\theta} p(\theta) \prod_{i=1}^{n} p(x_i|\theta)$$

Predict $$p(x|\theta)$$

Conditional distributions

"Supervised learning"

Conditional



Examples

- •Linear regression
- •Logistic regression
- •Probabilistic decision trees
- • etc

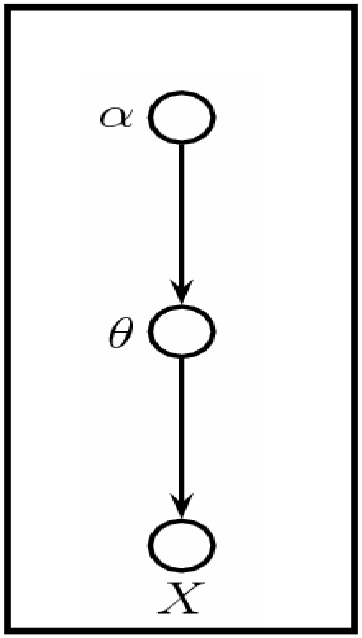
Priors: L1, L2, uniform, etc

Fit $\hat{\theta} = \arg\max_{\theta} p(\theta) \prod_{i=1}^{n} p(y_i | x_i, \theta)$

Predict $p(y|x, \theta)$ Plug-in rule

Compound distributions

“Bayesian unsupervised learning”



Examples

- Gauss_NormInvGamma
- Bernoulli_Beta
- Discrete_Dirichlet
- etc

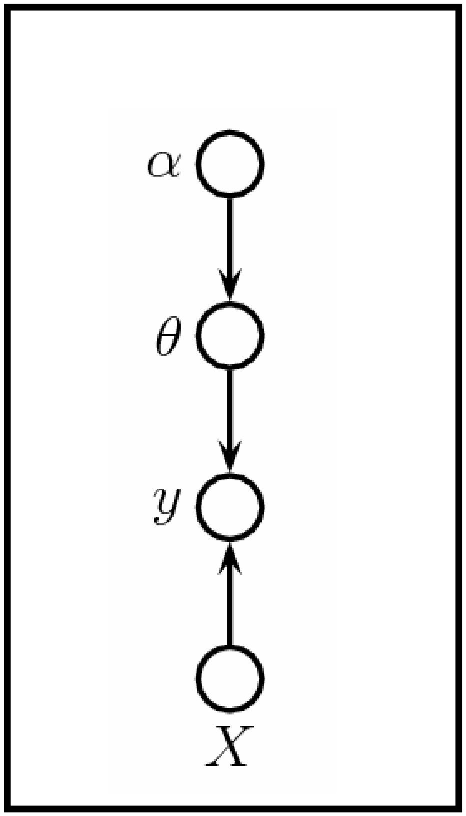
Update hyper-parameters

Fit $$p(\theta|\alpha') = p(\theta|X, \alpha) \propto p(X|\theta)p(\theta|\alpha)$$

Predict $$p(x|\alpha) = \int p(x|\theta)p(\theta|\alpha)d\theta$$

Conditional Compound

"Bayesian supervised learning"



Examples

- •Linreg_MvnInvGamma
- •Logreg_Mvn
- • etc

Fit $$p(\theta|\alpha') \propto p(Y|X,\theta)p(\theta|\alpha)$$

Predict $$p(y|x,\alpha) = \int p(y|x,\theta)p(\theta|\alpha)d\theta$$

Mixtures of simple distributions

"Unsupervised learning"

$$\begin{tikzcd} \theta \arrow[r] & z \arrow[d] \\ \theta \arrow[r] & X \end{tikzcd}$$

Examples

- Mix_Mvn
- Mix_Bernoulli
- etc

Fit $\hat{\theta} = \arg\max_{\theta} p(\theta) p(X|\theta)$ EM

Predict $p(z|x, \theta)$ Eg. Soft assignment in clustering, or PPCA projection

Mixtures of compound distributions

"Bayesian unsupervised learning"

$$\begin{tikzcd} \alpha \arrow[r] & \theta \arrow[r] & z \arrow[d] \\ \alpha \arrow[r] & \theta \arrow[r] & X \end{tikzcd}$$

Examples

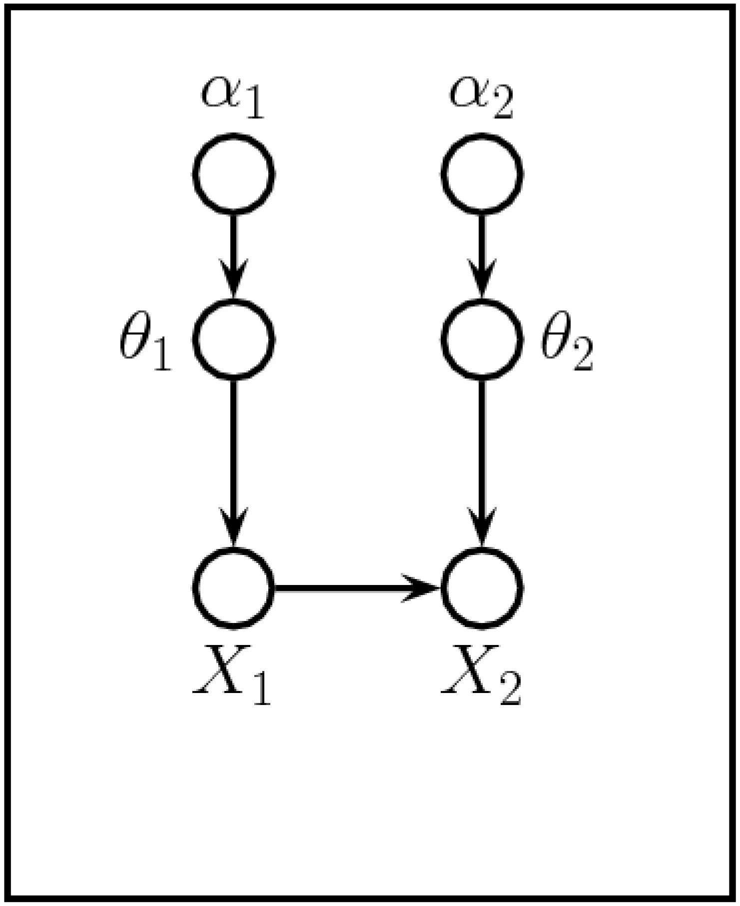
- Mix_Mvn_MvnInvWishart
- Mix_Bernoulli_Beta
- etc

Fit $p(\theta|\alpha') \propto p(\theta|\alpha)p(X|\theta)$ VBEM, Gibbs

Predict $p(z|x,\alpha)$ Eg. Soft assignment in clustering, or PPCA projection

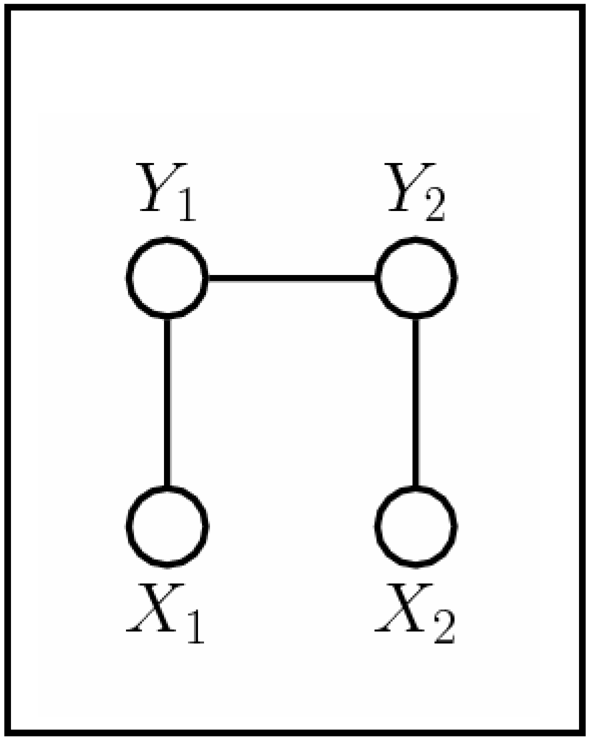
Directed graphical models

PMTK supports general DAGs, with arbitrary CPDs, just like BNT



Undirected graphical models

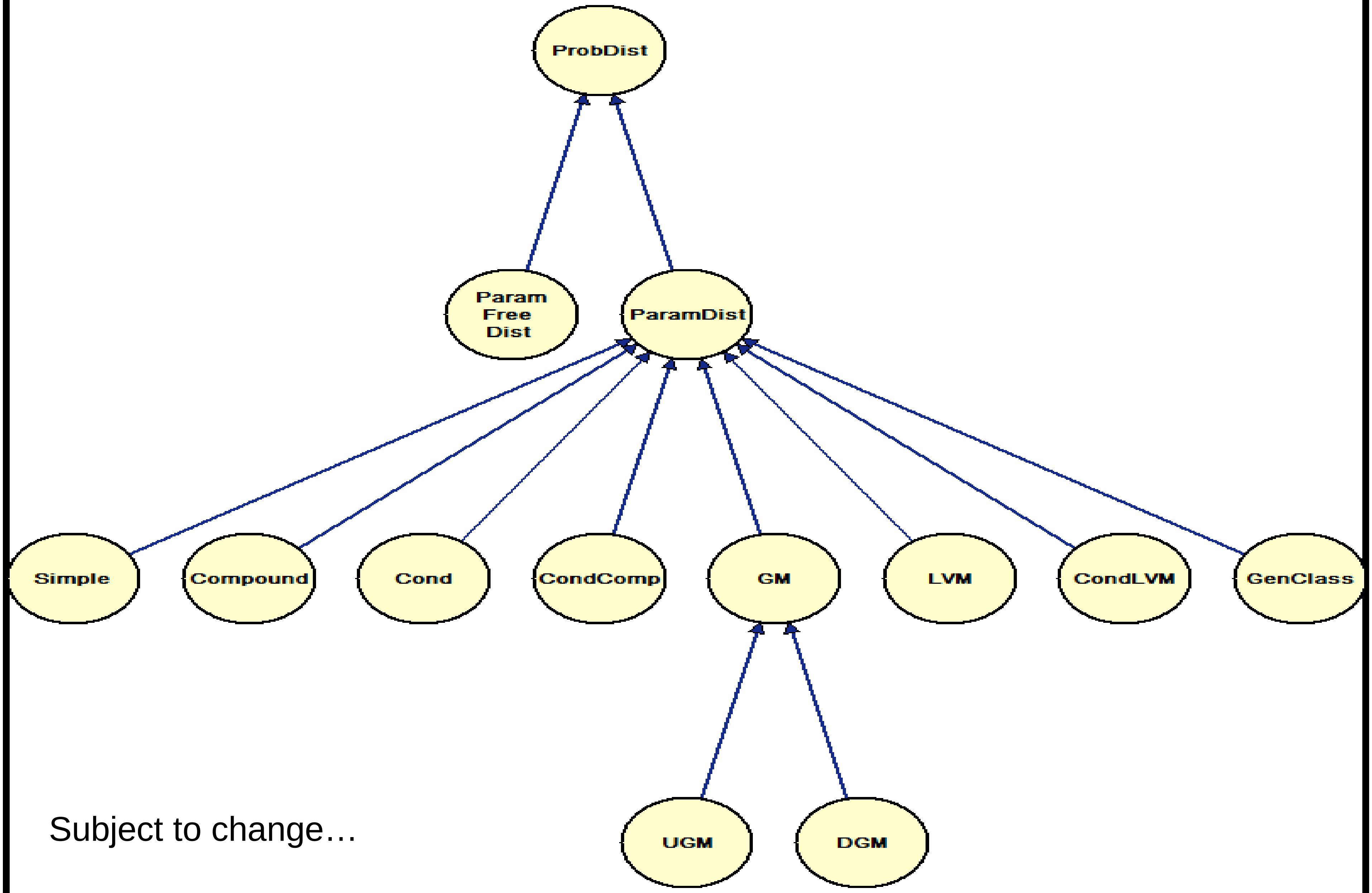
PMTK will support UGMs (including CRFs).
Currently only Gaussian MRFs are supported



Parameter free distributions

- Sometimes a function (such as predict or marginal) may return a distribution with no free parameters e.g.
  - Bag of samples
  - Distribution over paths in an HMM
  - Laplace approximation to a posterior
- Such distributions do not support ‘fit’ or ‘predict’, but do support sampling, plotting, etc.
- We avoid the term “non-parametric” distributions to prevent confusion with GPs, DPs, kNN, etc.

ProbDist class hierarchy



Subject to change…

Main classes in PMTK

- Probability distributions
- Inference engines
- Transformers

Inference Engines

- Exact inference for conjugate exponential compound distributions with no missing data
- Exact inference for multivariate Gaussians
- Laplace approximation
- MCMC: Gibbs and MH (user-specified proposal)
- For discrete-state GMs:
  - Dynamic programming (forwards-backwards / BP)
  - Exhaustive enumeration
  - Variable elimination
- Various optimization algorithms*
  - EM, L-BFGS, projected gradient, local search, etc.

* Mostly written by Mark Schmidt

Transformers

- We rarely build probability models directly on raw data. Instead we apply deterministic preprocessing operations. This can be considered part of the model, and chosen by eg. CV.
- PMTK supports various transformers
  – Standardize
  – PCA
  – Basis function expansion (“kernels”)

$$\phi(\mathbf{x}) = [K(\mathbf{x}, \mathbf{x}_1), \ldots, K(\mathbf{x}, \mathbf{x}_n)]$$

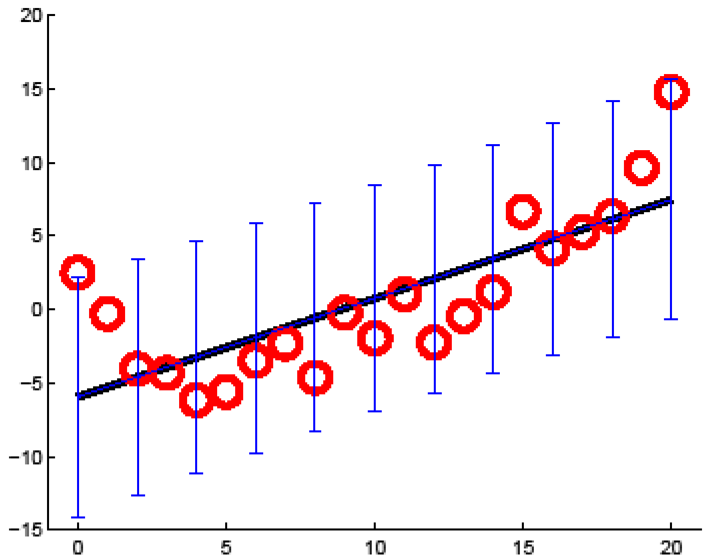
- Transformers can be chained together*

* Idea borrowed from “Spider” toolbox

Outline

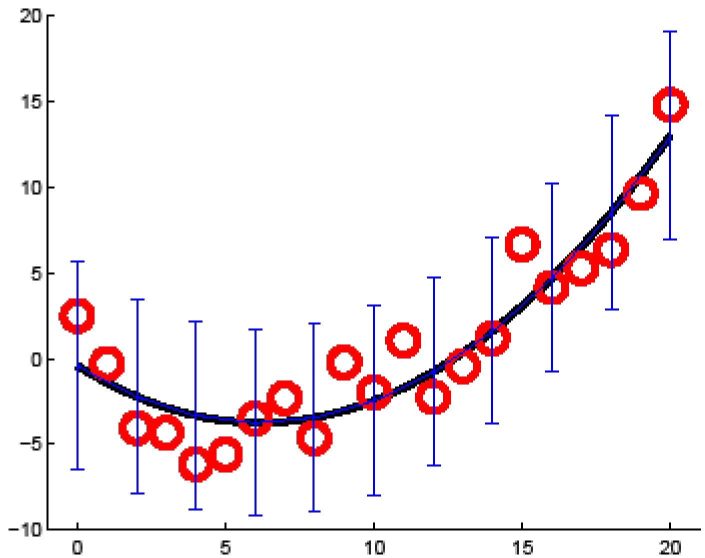
- Why yet another toolkit?
- What is PMTK?
- Examples

Vanilla linear regression



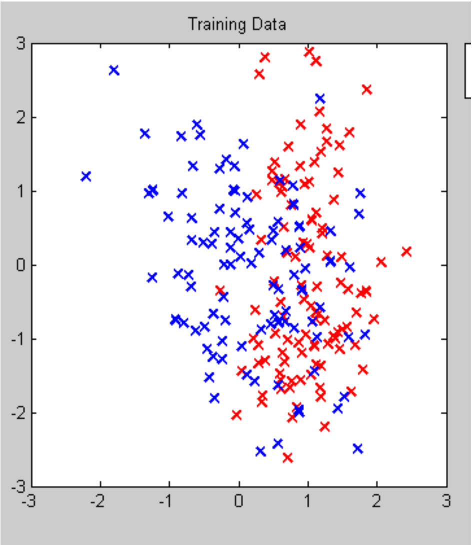

```
model = LinregDist;
model = fit(model, 'X', xtrain, 'y', ytrain);
ypred = predict(model, xtest);
mu = mean(ypred); sigma = sqrt(var(ypred));
errorbar(xtest, mu, sigma);
```

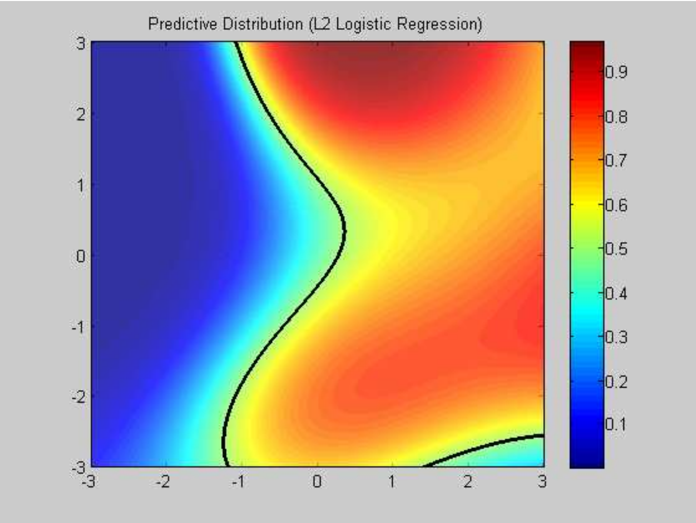
Polynomial linear regression

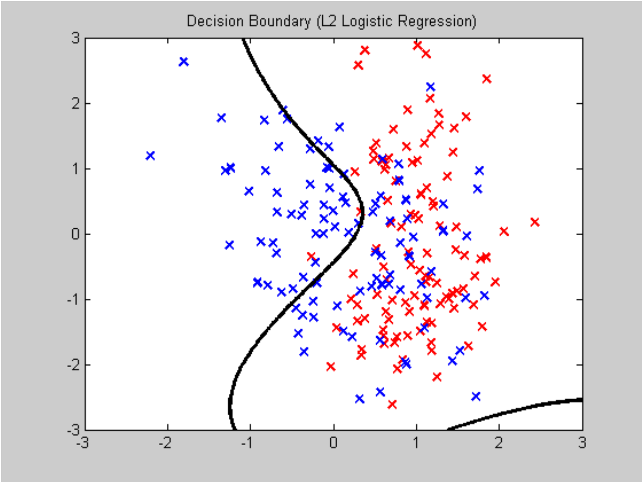



```
T =  ChainTransformer({RescaleTransformer, ...
   PolyBasisTransformer(2)});
model = LinregDist('transformer', T);
...
```

Kernel logistic regression

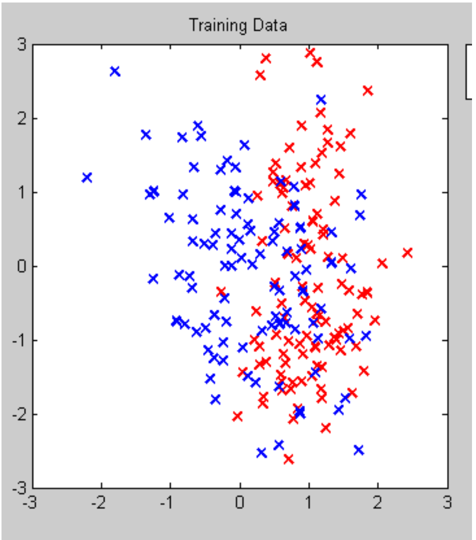


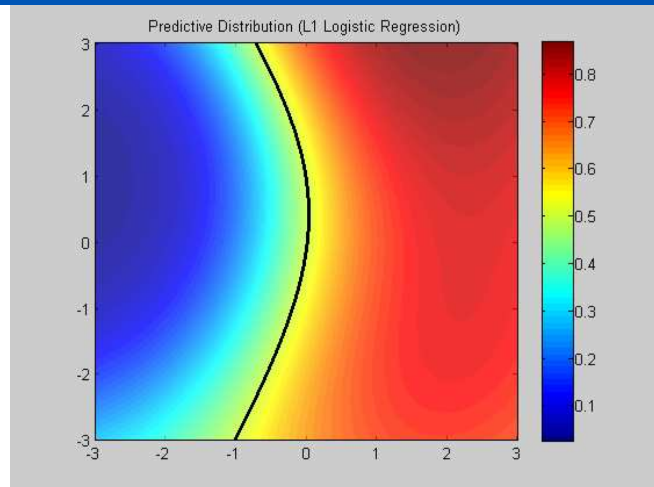


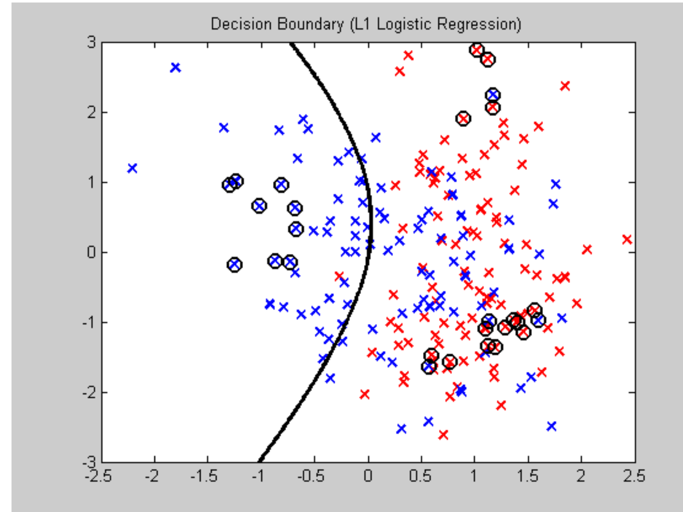



```
T = chainTransformer({standardizeTransformer,...
    kernelTransformer('rbf',sigma)} );
model = logregDist('nclasses',2, 'transformer', T);
model = fit(model,'prior','l2','lambda',lambda,...
   'X',Xtrain,'y',ytrain,'optMethod','lbfgs');
[X1grid, X2grid] = meshgrid(-3:0.02:3,-3:0.02:3);
pred = predict(model,'X',[X1grid(:),X2grid(:)]);
pgrid = reshape(pred.probs(:,1),nr,nc); surf(pgrid);
```

Sparse Kernel logreg ("RVM")

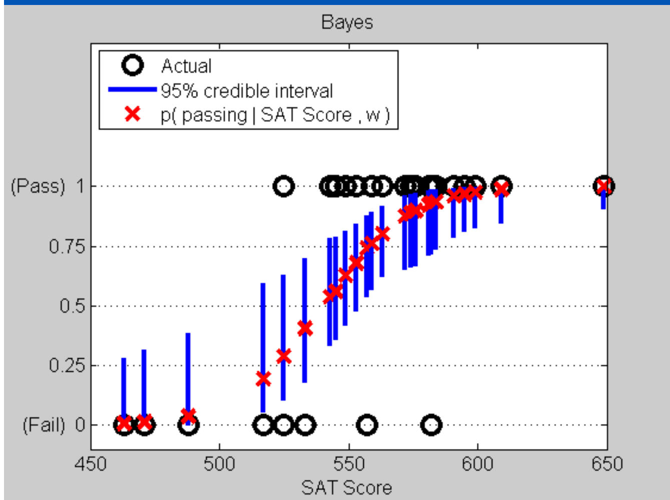


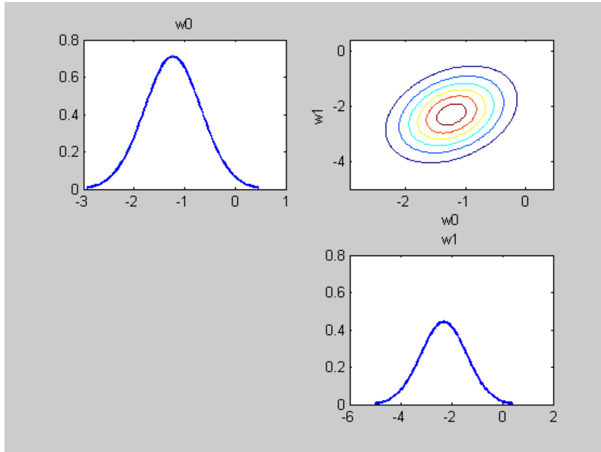


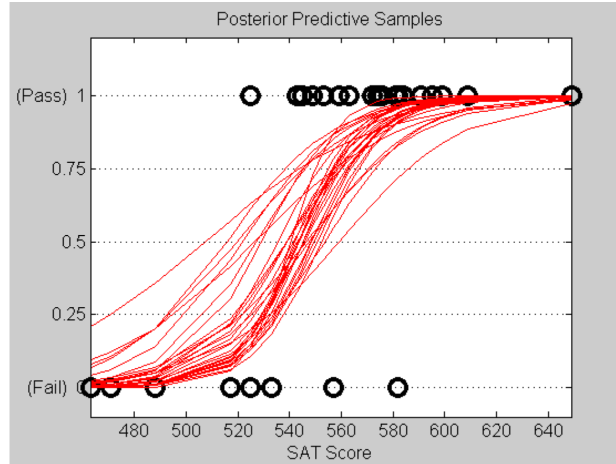



```
...
model = fit(model,'prior','l1','lambda',lambda,...
    'X',Xtrain,'y',ytrain,'optMethod','projGrad');
...
```

Bayesian logistic regression in 1D

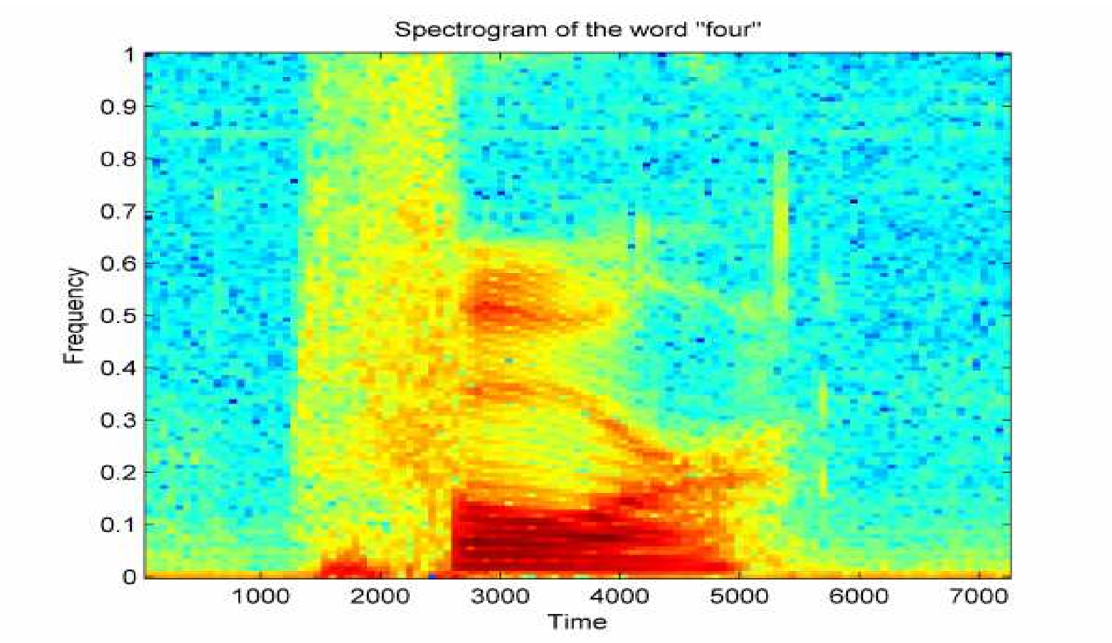


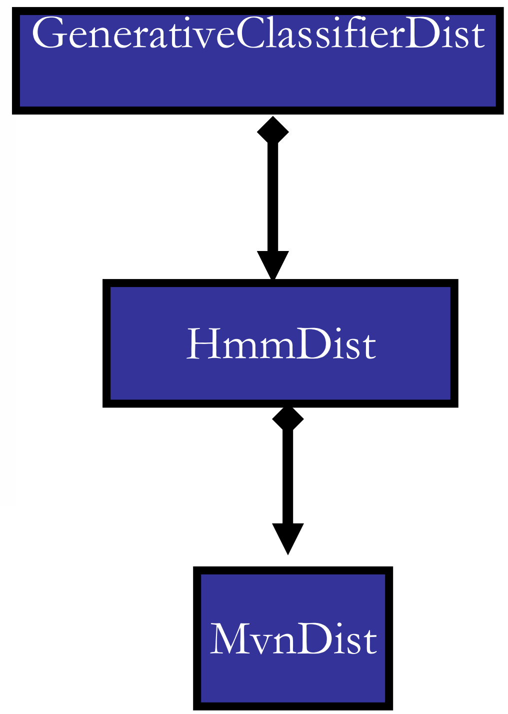


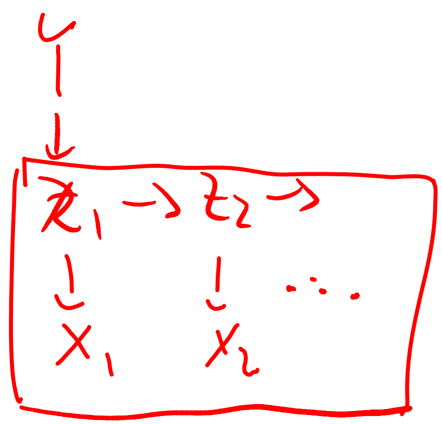



```
T = chainTransformer({standardizeTransformer,...
     addOnesTransformer});
model = Logreg_MvnDist('nclasses',2,'transformer',T,..
    'prior',MvnDist(zeros(d,1), 1e-3*eye(d)));
model = fit(model,'X',X,'y',y,'method','laplace');
subplot(2,2,1); plot(marginal(model.muDist,1)); ...
[pred] = predict(model,'X',X,'method','mc',...
    'nsamples',100);
```

HMM for isolated word classification







HMM classifier in PMTK

Sequence Classification

```
load data45;
nstates = 5;
```

```
%Initial Guesses for EM
pi0 = [1,0,0,0,0];
transmat0 = normalize(diag(ones(nstates,1)) + diag(ones(nstates-1,1),1),2);
```

```
condDensity = HmmDist('nstates',5,'observationModel',MvnDist());
```

```
model = GenerativeClassifierDist('classConditionals',condDensity,'nclasses',2,'classSupport',4:5);
```

```
trainingData = {train4,train5};
trainingLabels = [4,5];
fitOptions = {'transitionMatrix0',transmat0,'pi0',pi0};
```

```
model = fit(model,'observations',trainingData,'labels',trainingLabels,'fitOptions',fitOptions);
```

```
pred = predict(model,test45);
yhat = mode(pred);
nerrors = sum(yhat ~= labels');
display(nerrors);
```

A peek under the hood

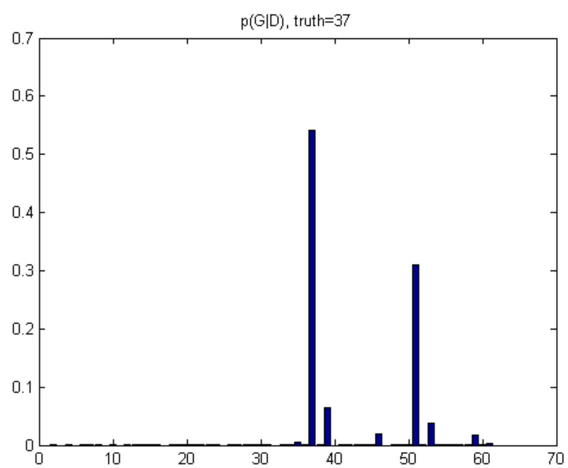
- Part of the fit method for HmmDist

```
% E step
for j=1:nseq
  trellis = predict(model, seq{j});
  for t=1:length(seq{j})-1
    essTrans = essTrans +  marginal(trellis,t,t+1);
  end
...
end
% M step
model.transDensity = fit(model.transDensity,...
   'suffStat',essTrans)
...
```

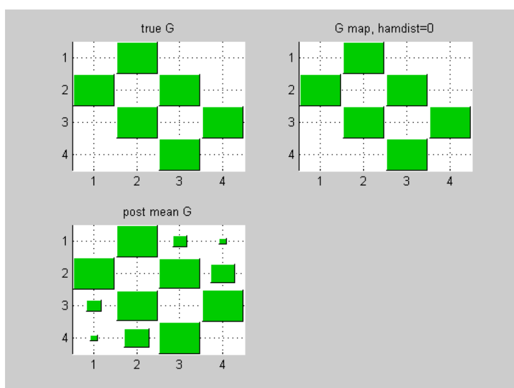
Model selection

- ModelSet.fit computes posterior over models.
- ModelSet.predict does Bayes model averaging.
- Can approximate marginal likelihood using BIC, or CV score.
- Can approximate posterior by single best model.

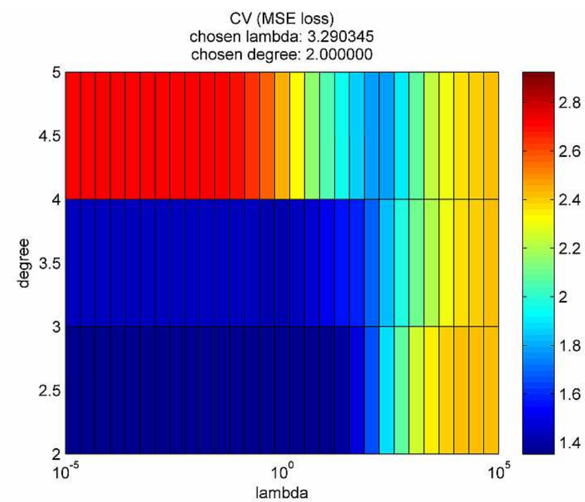
P(G|D)



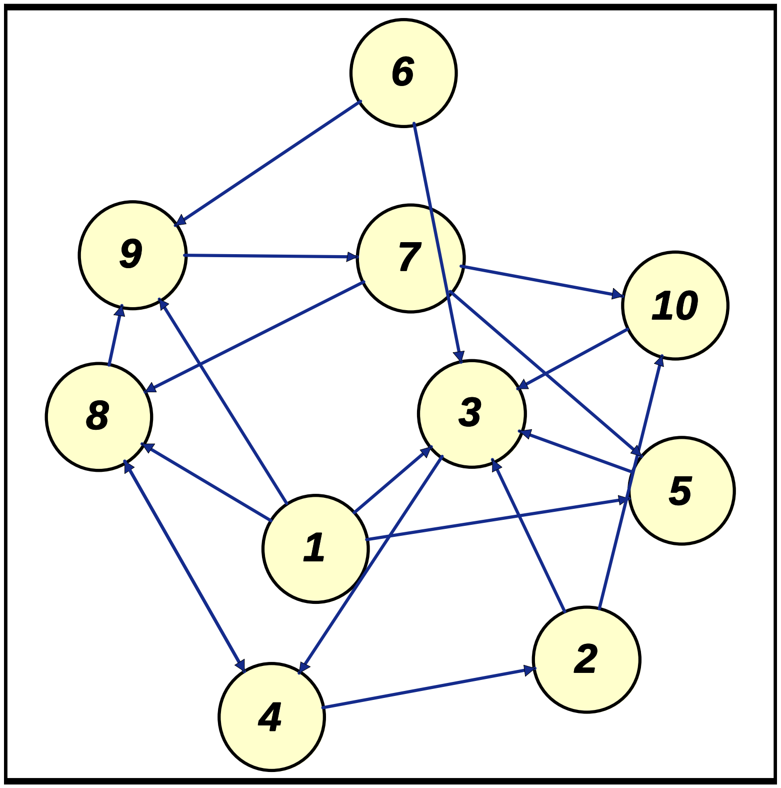
P(G_{ij}=1|D)



CV(σ,λ)



Graph structure visualization




```
G=rand(10,10)>0.8;figure;Graphlayout(G)
```

Conclusions

- PMTK strives to strike the right balance between simplicity, generality and efficiency.
- It combines elements from ML, GM and Bayesian communities.
- It provides a unified conceptual framework to data modeling, which is particularly useful for teaching.
- The source code is on pmtk.googlecode.com
- Email me if you want to use and/or develop it.