Probabilistic Models for Preference Learning

Zoubin Ghahramani

Department of Engineering
University of Cambridge, UK

Machine Learning Department
Carnegie Mellon University, USA

zoubin@eng.cam.ac.uk
http://learning.eng.cam.ac.uk/zoubin/

Choice Models and Preference Learning Workshop
NIPS 2011

What is preference learning?

- Learning where the data is in the form of inequality relations:

- Surprisingly many machine learning problems fall into this class.

Instance Preference

Data is a set of preference relations measured on items or instances:

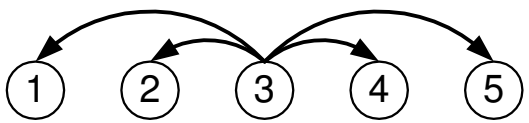
$$\mathcal{D} = \{(x \succ x')\} \qquad x \in \mathcal{X}, x' \in \mathcal{X}$$

"Coke" vs "Pepsi"

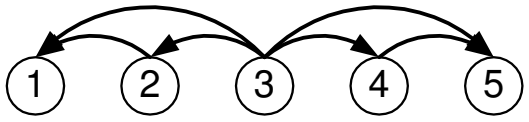
Label Preference or Label Ranking

items $\mathcal{X}$, labels $\mathcal{L}$,
$x \in \mathcal{X}$, $\ell, \ell' \in \mathcal{L}$

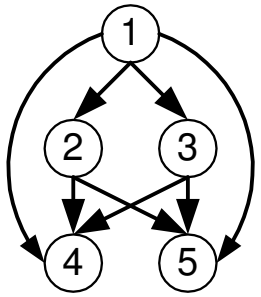
$$\mathcal{D} = \{(x, \{\ell \succ \ell'\})\}$$



(a) Classification



(b) Ordinal Regression



(c) Hierarchical multiclass setting

Preference Functions

I'm going to focus on instance preferences.

If the user exhibits a preference

$$x \succ x'$$

this suggests that he has a utility function (or preference function) f such that

$$f(x) > f(x')$$

perhaps corrupted by noise.

Many machine learning methods can be thought of as learning such preference functions from data.

(Thurstone, 1927; Elo 1978)

Learning Preference Functions

How do we learn preference functions from data $\mathcal{D} = \{(x_n \succ x'_n)_{n=1}^N\}$?

- Optimisation approach

$$\min_f \sum_n L_n(f) + \lambda R(f)$$

- Probabilistic approach

$$p(f|\mathcal{D}) = \frac{p(\mathcal{D}|f)p(f)}{p(\mathcal{D})}$$

Let's define a general and flexible prior $p(f)$ over preference functions.

Gaussian Processes (a mini tutorial)

A Gaussian process defines a distribution over functions, $p(f)$, where f is a function mapping some input space $\mathcal{X}$ to $\Re$.

$$f : \mathcal{X} \to \Re.$$

Notice that f can be an infinite-dimensional quantity (e.g. if $\mathcal{X} = \Re$)

Let $\mathbf{f} = (f(x_1), \ldots, f(x_n))$ be an n-dimensional vector of function values evaluated at n points $x_i \in \mathcal{X}$. Note $\mathbf{f}$ is a random variable.

Definition: $p(f)$ is a Gaussian process if for *any* finite subset $\{x_1, \ldots, x_n\} \subset \mathcal{X}$, the marginal distribution over that finite subset $p(\mathbf{f})$ has a multivariate Gaussian distribution.

Gaussian process covariance functions (kernels)

$p(f)$ is a Gaussian process if for *any* finite subset $\{x_1, \ldots, x_n\} \subset \mathcal{X}$, the marginal distribution over that finite subset $p(\mathbf{f})$ has a multivariate Gaussian distribution.

Gaussian processes (GPs) are parameterized by a mean function, $\mu(x)$, and a covariance function, or kernel, $K(x, x')$.

$$p(f(x), f(x')) = \mathsf{N}(\mu, \Sigma)$$

where

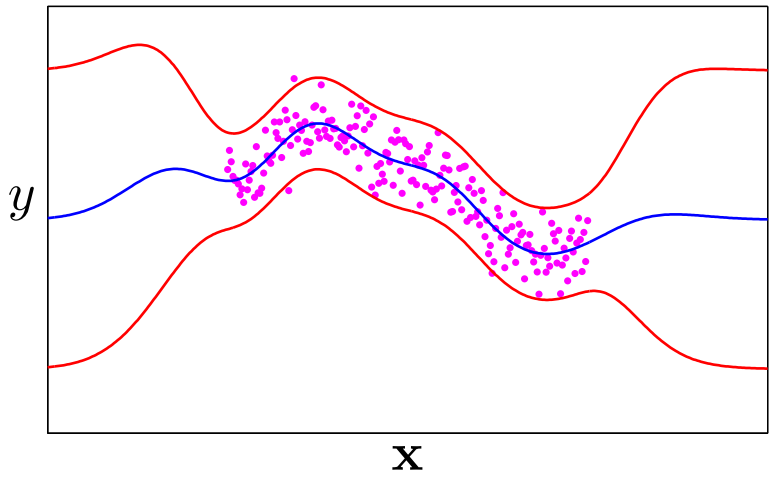
$$\mu = \begin{bmatrix} \mu(x) \\ \mu(x') \end{bmatrix} \quad \Sigma = \begin{bmatrix} K(x,x) & K(x,x') \\ K(x',x) & K(x',x') \end{bmatrix}$$

and similarly for $p(f(x_1), \ldots, f(x_n))$ where now μ is an $n \times 1$ vector and Σ is an $n \times n$ matrix.

Gaussian Processes for Nonlinear regression

GPs are often used for nonlinear regression:

You want to learn a function f with error bars from data $\mathcal{D} = \{\mathbf{X}, \mathbf{y}\}$



A Gaussian process defines a distribution over functions $p(f)$ which can be used for Bayesian regression:

$$p(f|\mathcal{D}) = \frac{p(f)p(\mathcal{D}|f)}{p(\mathcal{D})}$$

[Here we will use them for preference learning.]

Gaussian process covariance functions

Gaussian processes (GPs) are parameterized by a mean function, $\mu(x)$, and a covariance function, $K(x, x')$.

An example covariance function:

$$K(x_i, x_j) = v_0 \exp\left\{-\left(\frac{|x_i - x_j|}{r}\right)^{\alpha}\right\} + v_1 + v_2\,\delta_{ij}$$

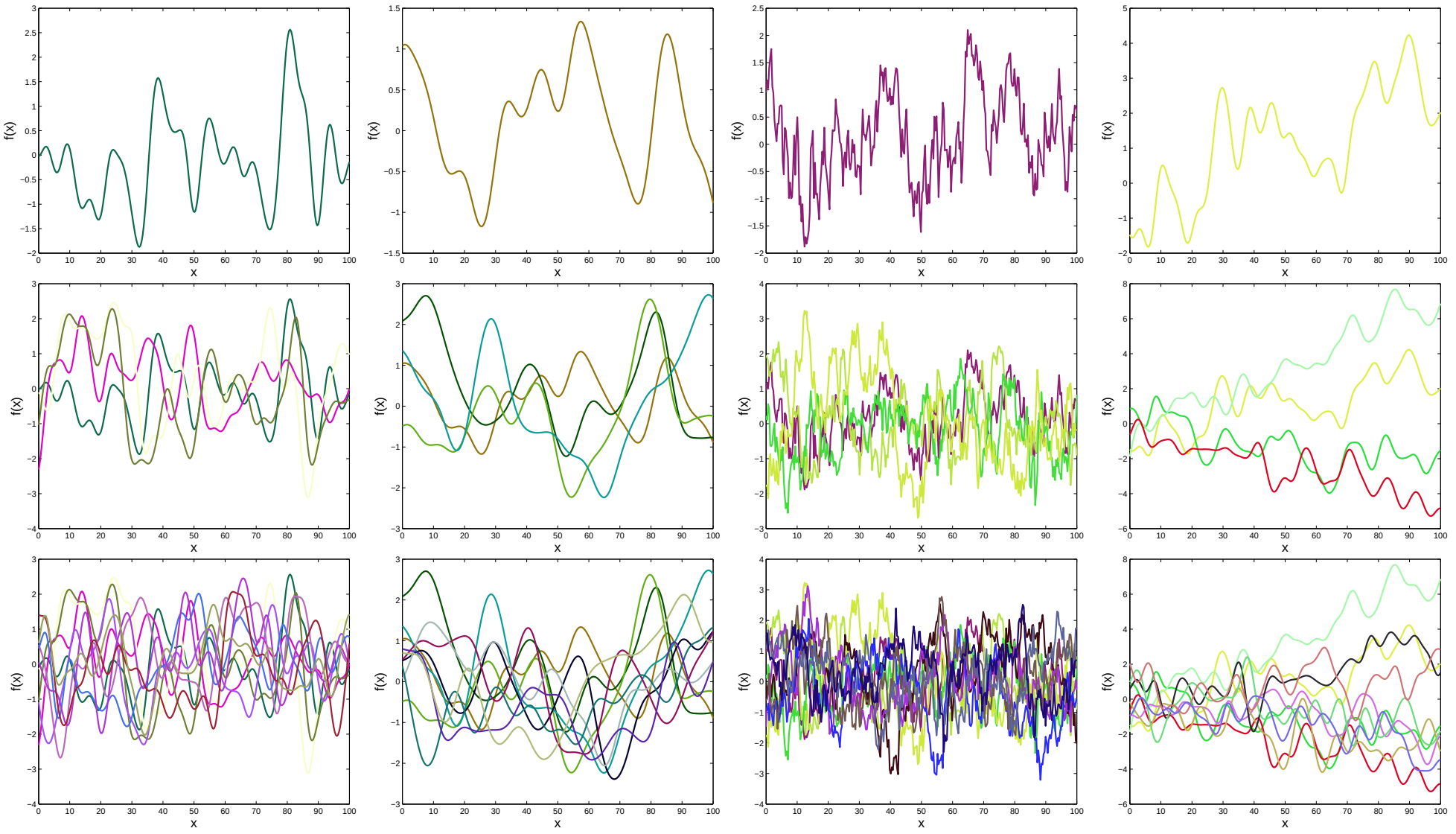
with parameters $(v_0, v_1, v_2, r, \alpha)$

These kernel parameters are interpretable and can be learned from data:

v_0	signal variance
v_1	variance of bias
v_2	noise variance
r	lengthscale
α	roughness

Once the mean and covariance functions are defined, everything else about GPs follows from the basic rules of probability applied to mutivariate Gaussians.

Samples from GPs with different $K(x, x')$



GP learning the kernel

Consider the covariance function K with hyperparameters $\boldsymbol{\theta} = (v_0, v_1, r_1, \ldots, r_d, \alpha)$:

$$K_{\boldsymbol{\theta}}(\mathbf{x}_i, \mathbf{x}_j) = v_0 \exp\left\{ -\sum_{d=1}^{D} \left(\frac{|x_i^{(d)} - x_j^{(d)}|}{r_d} \right)^{\alpha} \right\} + v_1$$

Given a data set $\mathcal{D} = (\mathbf{X}, \mathbf{y})$, how do we learn $\boldsymbol{\theta}$?

The marginal likelihood is a function of $\boldsymbol{\theta}$

$$p(\mathbf{y}|\mathbf{X}, \boldsymbol{\theta}) = \mathcal{N}(\mathbf{0}, \mathbf{K}_{\boldsymbol{\theta}} + \sigma^2 \mathbf{I})$$

where its log is:

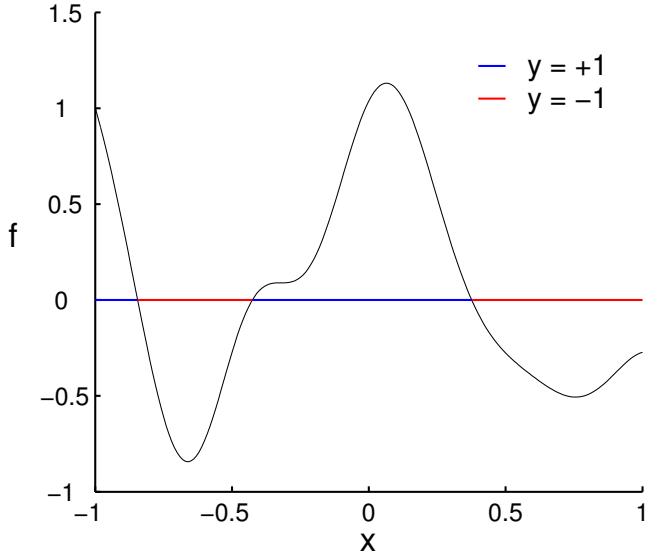
$$\ln p(\mathbf{y}|\mathbf{X}, \boldsymbol{\theta}) = -\frac{1}{2} \ln \det(\mathbf{K}_{\boldsymbol{\theta}} + \sigma^2 \mathbf{I}) - \frac{1}{2} \mathbf{y}^\top (\mathbf{K}_{\boldsymbol{\theta}} + \sigma^2 \mathbf{I})^{-1} \mathbf{y} + \text{const}$$

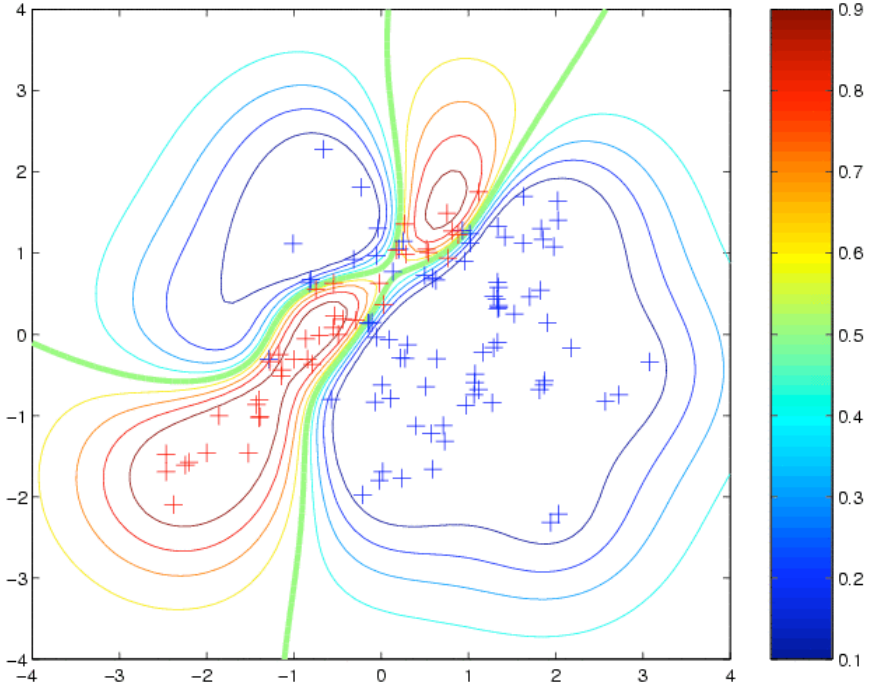
which can be optimized as a function of $\boldsymbol{\theta}$ and σ.

Alternatively, one can infer $\boldsymbol{\theta}$ using Bayesian methods, which is more costly but immune to overfitting.

Using Gaussian Processes for Classification

Binary classification problem: Given a data set $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$, with binary class labels $y_i \in \{-1, +1\}$, infer class label probabilities at new points.





There are many ways to relate function values $f_i = f(\mathbf{x}_i)$ to class probabilities:

$$p(y_i|f_i) = \begin{cases} \frac{1}{1+\exp(-y_i f_i)} & \text{sigmoid (logistic)} \\ \Phi(y_i f_i) & \text{cumulative normal (probit)} \\ \boldsymbol{H}(y_i f_i) & \text{threshold} \\ \epsilon + (1-2\epsilon)\boldsymbol{H}(y_i f_i) & \text{robust threshold} \end{cases}$$

Non-Gaussian likelihood, so we need to use approximate inference methods (Laplace, EP, MCMC).

Support Vector Machines

Consider soft-margin Support Vector Machines:

$$\min_{\mathbf{w}} \ \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_i (1 - y_i f_i)_+$$

where $()_+$ is the hinge loss and $f_i = f(\mathbf{x}_i) = \mathbf{w} \cdot \mathbf{x}_i + w_0$. Let's kernelize this:

$$\mathbf{x}_i \to \boldsymbol{\phi}(\mathbf{x}_i) = k(\cdot, \mathbf{x}_i), \qquad \mathbf{w} \to f(\cdot)$$

By reproducing property: $\langle k(\cdot, \mathbf{x}_i), f(\cdot)\rangle = f(\mathbf{x}_i).$

By representer theorem, solution: $f(\mathbf{x}) = \sum_i \alpha_i k(\mathbf{x}, \mathbf{x}_i)$

Defining $\mathbf{f} = (f_1, \dots f_N)^T$ note that $\mathbf{f} = \mathbf{K}\boldsymbol{\alpha}$, so $\boldsymbol{\alpha} = \mathbf{K}^{-1}\mathbf{f}$

Therefore the regularizer $\frac{1}{2}\|\mathbf{w}\|^2 \to \frac{1}{2}\|f\|^2_{\mathcal{H}} = \frac{1}{2}\langle f(\cdot), f(\cdot)\rangle_{\mathcal{H}} = \frac{1}{2}\boldsymbol{\alpha}^\top \mathbf{K}\boldsymbol{\alpha} = \frac{1}{2}\mathbf{f}^\top \mathbf{K}^{-1}\mathbf{f}$

So we can rewrite the kernelized SVM loss as:

$$\boxed{\min_{\mathbf{f}} \ \frac{1}{2}\mathbf{f}^\top \mathbf{K}^{-1}\mathbf{f} + C\sum_i (1 - y_i f_i)_+}$$

Support Vector Machines and Gaussian Processes

We can write the SVM loss as: $\min_{\mathbf{f}} \ \frac{1}{2}\mathbf{f}^\top \mathbf{K}^{-1}\mathbf{f} + C\sum_i (1 - y_i f_i)_+$

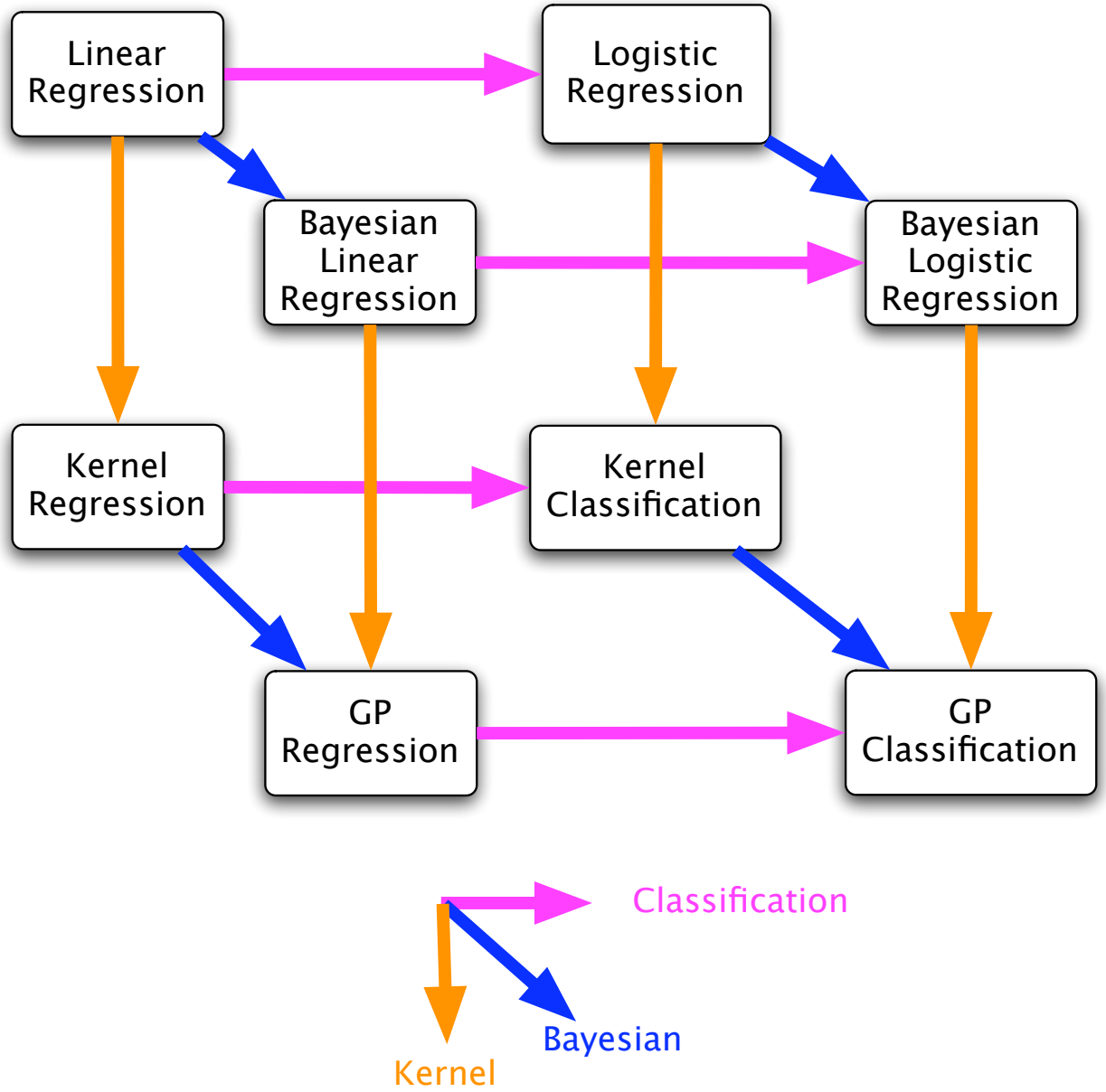
We can write the negative log of a GP likelihood as: $\frac{1}{2}\mathbf{f}^\top \mathbf{K}^{-1}\mathbf{f} - \sum_i \ln p(y_i|f_i) + c$

Equivalent? No.

With Gaussian processes we:

- Handle **uncertainty** in unknown function $\mathbf{f}$ by averaging, not minimization.
- Compute $p(y = +1|\mathbf{x}) \neq p(y = +1|\hat{\mathbf{f}}, \mathbf{x})$.
- Can **learn the kernel parameters** automatically from data, no matter how flexible we wish to make the kernel.
- Can **learn the regularization parameter** C without cross-validation.
- Can incorporate **interpretable** noise models and priors over functions, and can sample from prior to get intuitions about the model assumptions.
- We can combine **automatic feature selection** with learning using ARD.

A picture


Linear Regression
Logistic Regression
Bayesian Linear Regression
Bayesian Logistic Regression
Kernel Regression
Kernel Classification
GP Regression
GP Classification
Classification
Bayesian
Kernel

Matlab Demo: Gaussian Process Classification

`matlab/gpml-matlab/gpml-demo`

`demo_ep_2d`

`demo_gpr`

Some Comparisons

Table 1: Test errors and predictive accuracy (smaller is better) for the GP classifier, the support vector machine, the informative vector machine, and the sparse pseudo-input GP classifier.

Data set			**GPC**		**SVM**		**IVM**			**SPGPC**		
name	train:test	dim	err	nlp	err	#sv	err	nlp	M	err	nlp	M
synth	250:1000	2	0.097	*0.227*	0.098	98	0.096	*0.235*	150	**0.087**	*0.234*	4
crabs	80:120	5	0.039	*0.096*	0.168	67	0.066	*0.134*	60	**0.043**	*0.105*	10
banana	400:4900	2	0.105	*0.237*	0.106	151	**0.105**	*0.242*	200	0.107	*0.261*	20
breast-cancer	200:77	9	0.288	*0.558*	**0.277**	122	0.307	*0.691*	120	0.281	*0.557*	2
diabetes	468:300	8	0.231	*0.475*	**0.226**	271	0.230	*0.486*	400	0.230	*0.485*	2
flare-solar	666:400	9	0.346	*0.570*	**0.331**	556	0.340	*0.628*	550	0.338	*0.569*	3
german	700:300	20	0.230	*0.482*	0.247	461	0.290	*0.658*	450	**0.236**	*0.491*	4
heart	170:100	13	0.178	*0.423*	**0.166**	92	0.203	*0.455*	120	0.172	*0.414*	2
image	1300:1010	18	0.027	*0.078*	0.040	462	**0.028**	*0.082*	400	0.031	*0.087*	200
ringnorm	400:7000	20	0.016	*0.071*	0.016	157	0.016	*0.101*	100	**0.014**	*0.089*	2
splice	1000:2175	60	0.115	*0.281*	**0.102**	698	0.225	*0.403*	700	0.126	*0.306*	200
thyroid	140:75	5	0.043	*0.093*	0.056	61	0.041	*0.120*	40	**0.037**	*0.128*	6
titanic	150:2051	3	0.221	*0.514*	**0.223**	118	0.242	*0.578*	100	0.231	*0.520*	2
twonorm	400:7000	20	0.031	*0.085*	0.027	220	0.031	*0.085*	300	**0.026**	*0.086*	2
waveform	400:4600	21	0.100	*0.229*	0.107	148	0.100	*0.232*	250	**0.099**	*0.228*	10

From (Naish-Guzman and Holden, 2008), using exactly same kernels.

Gaussian Processes for Preference Learning

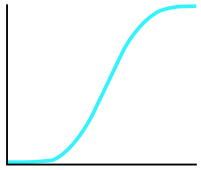
$$P(f|\mathcal{D}) = \frac{P(f)P(\mathcal{D}|f)}{P(\mathcal{D})}$$

The preference function likelihood can be written:

$$P_{\text{noise-free}}(v_n \succ u_n | f(v_n), f(u_n)) = \begin{cases} 1 & \text{if } f(v_n) > f(u_n) \\ 0 & \text{otherwise} \end{cases}$$

Using a probit (cumulative normal, Φ) instead of step function encodes preference judgement noise:

$$P(v_n \succ u_n | f(v_n), f(u_n), \sigma) = \Phi\left(\frac{f(v_n) - f(u_n)}{\sqrt{2}\sigma}\right)$$



$$P(\mathcal{D}|f, \sigma) = \prod_{n=1}^{N} P(v_n \succ u_n | f(v_n), f(u_n), \sigma)$$

(Chu and Ghahramani, Preference Learning with Gaussian Processes, ICML 2005)

see also poster by Ferenc Huszar in this workshop

ANSI C Source code: `http://www.gatsby.ucl.ac.uk/~chuwei/plgp.htm`

MAP Inference and Laplace Approximation

$$f_{\mathrm{MAP}} = \arg\max_f P(f|\mathcal{D})$$

$$S(f) = \ln P(f|\mathcal{D}) + c = -\sum_n \ln \Phi(z_n) + \frac{1}{2}\mathbf{f}^\top \Sigma^{-1}\mathbf{f}$$

where $z_n = (f(v_n) - f(u_n))/(\sqrt{2}\sigma)$. $S(f)$ is *convex*.

Laplace approximation to marginal likelihood:

$$P(\mathcal{D}) = \exp(-S(f_{\mathrm{MAP}}))\ |I + \Sigma\Lambda_{\mathrm{MAP}}|^{-1/2}$$

The marginal likelihood, $P(\mathcal{D})$, is used for model selection and hyper-parameter (i.e. kernel) learning.

Prediction

Once you learn about the latent preference function, prediction is straightforward:

$$\begin{aligned} P(r \succ s|\mathcal{D}) &= \int P(r \succ s|f(r), f(s))\ P(f(r), f(s)|\mathcal{D})\ df(r)df(s) \\ &= \Phi\left(\frac{\mu_r - \mu_s}{\sigma^*}\right) \end{aligned}$$

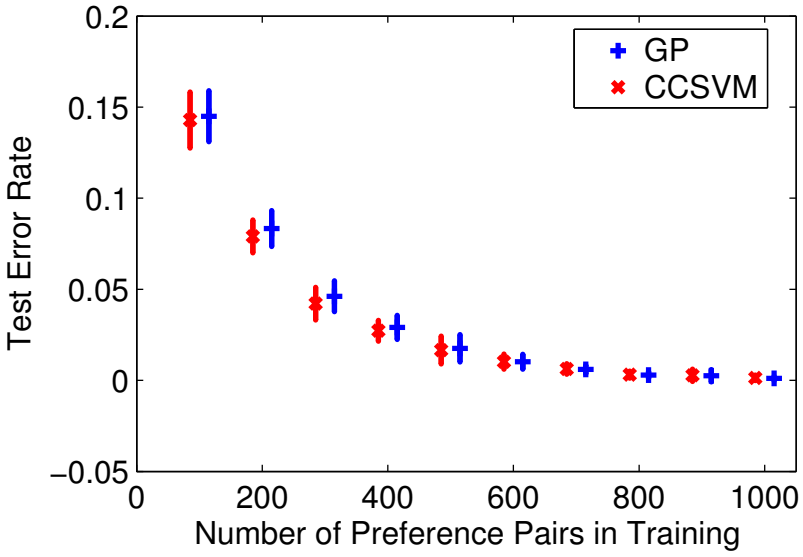
Results

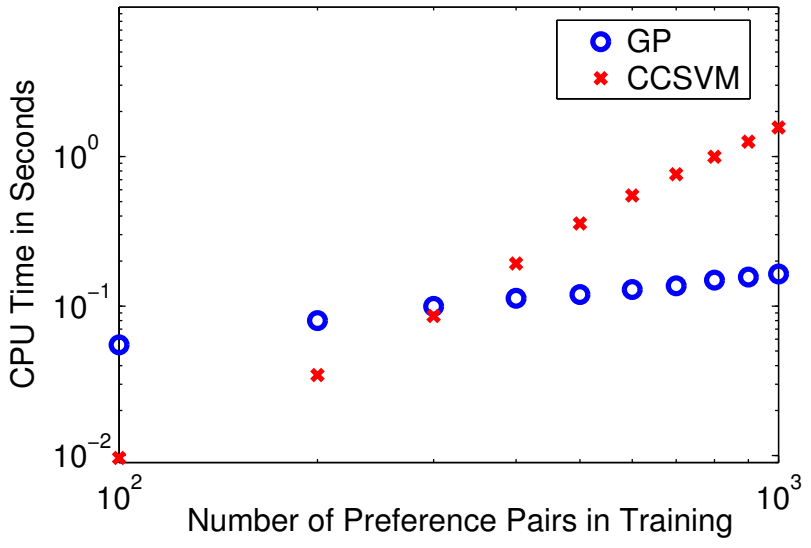
			Error Rate (%)		
Dataset	m	d	CC-SVM	GP	GPARD
Pyrimidines	100	27	16.01±2.29	**14.43±2.02**	16.56±1.76
Triazines	300	60	20.37±1.32⋆	**17.78±0.97**	18.26±1.45
MachineCpu	500	6	15.31±1.26⋆	**12.12±1.49**	12.86±0.71
BostonHouse	700	13	13.30±1.08	12.85±0.46	**10.44±0.64**
Abalone	1000	10	18.76±0.35⋆	**17.29±0.38**	17.35±0.38

CC-SVM: Har-Peled et al (2002)

GP: our method

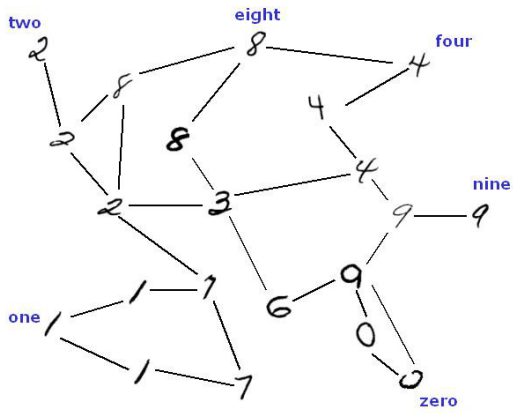
GP-ARD: using automatic relevance determination (ARD) kernels





Semi-supervised and Active Preference Learning

Graph-based semi-supervised learning: form a graph G connecting nearby points; compute the graph combinatorial Laplacian, L; use this as a likelihood for the GP.



$$P(G|\mathbf{f}) \propto \exp\left\{-\frac{1}{2}\mathbf{f}^T L \mathbf{f}\right\}$$

Intuition: for an unweighted graph this term encourages the function values to be similar at connected nodes

$$P(G|\mathbf{f}) \propto \exp\left\{-\frac{1}{2}\sum_{(i\sim j)}(f_i - f_j)^2\right\}$$

Active Learning: evaluate expected information gain (entropy reduction) from querying $u_k \overset{?}{\succ} v_k$; pick the best query

Chu, W. and Ghahramani, Z. (2005b) Extensions of Gaussian processes for ranking: semi-supervised and active learning, *NIPS Workshop on Learning to Rank.*

Semi-supervised and Active Preference Learning

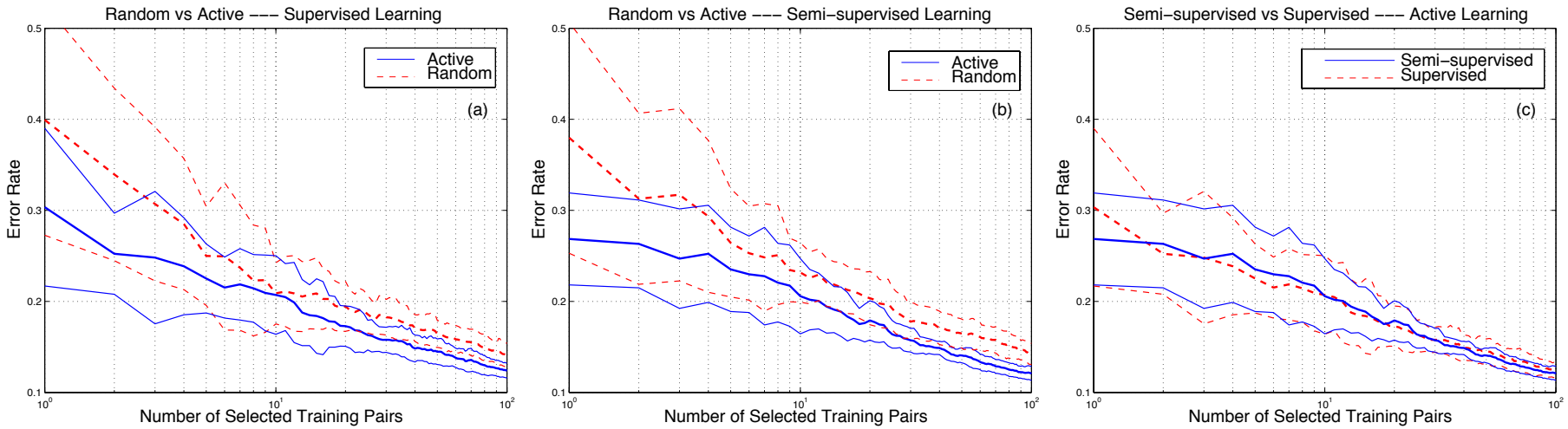


Figure 2: Performance of active learning on Boston Housing dataset. The error rate is the percent of incorrect preference predictions on 127765 pairs. The curves present the average over 20 trials along with standard deviation.

Unsupervised Preference Learning: Choice Models

Elimination by Aspects (EBA) model. The probability of choosing $x_i \succ x_j$ is:

$$p_{ij} = \frac{\sum_k w_k \; f_{ik}(1 - f_{jk})}{\sum_k w_k \; f_{ik}(1 - f_{jk}) + \sum_k w_k \; f_{jk}(1 - f_{ik})}$$

where f_{ik} is an indicator for whether item i has "aspect" or latent feature k.

Allowing some noise in preference data:

$$P(x_i \succ x_j | F, \mathbf{w}, \epsilon) = (1 - \epsilon) \; p_{ij} + \epsilon \; p_{ij}$$

Infinite sparse binary feature matrix drawn from an Indian Buffet Process:

$$F \sim \text{IBP}$$

Weights $\mathbf{w} = (w_1, w_2, \ldots)$ drawn from Gamma distribution.

Model learns how many latent features are needed to model the preference data.

Görür, Jäkel, and Rasmussen. A choice model with infinitely many latent features. ICML 2006

Clustering Users Based on Preferences

Consider data where users, indexed by u express preferences for items.
Let $p_{uij} = 1$ denote the observation that user u prefers item i to item j.

Our goal is to *cluster users on the basis of such preference data*.

We use x_{ui} to denote the utility or value of item i to user u.
Preferences are expressed as inequalities in utility:

$$p_{uij} = 1 \quad \text{iff} \quad x_{ui} > x_{uj}.$$

and utilities are dot products of a preference pattern for users and items:

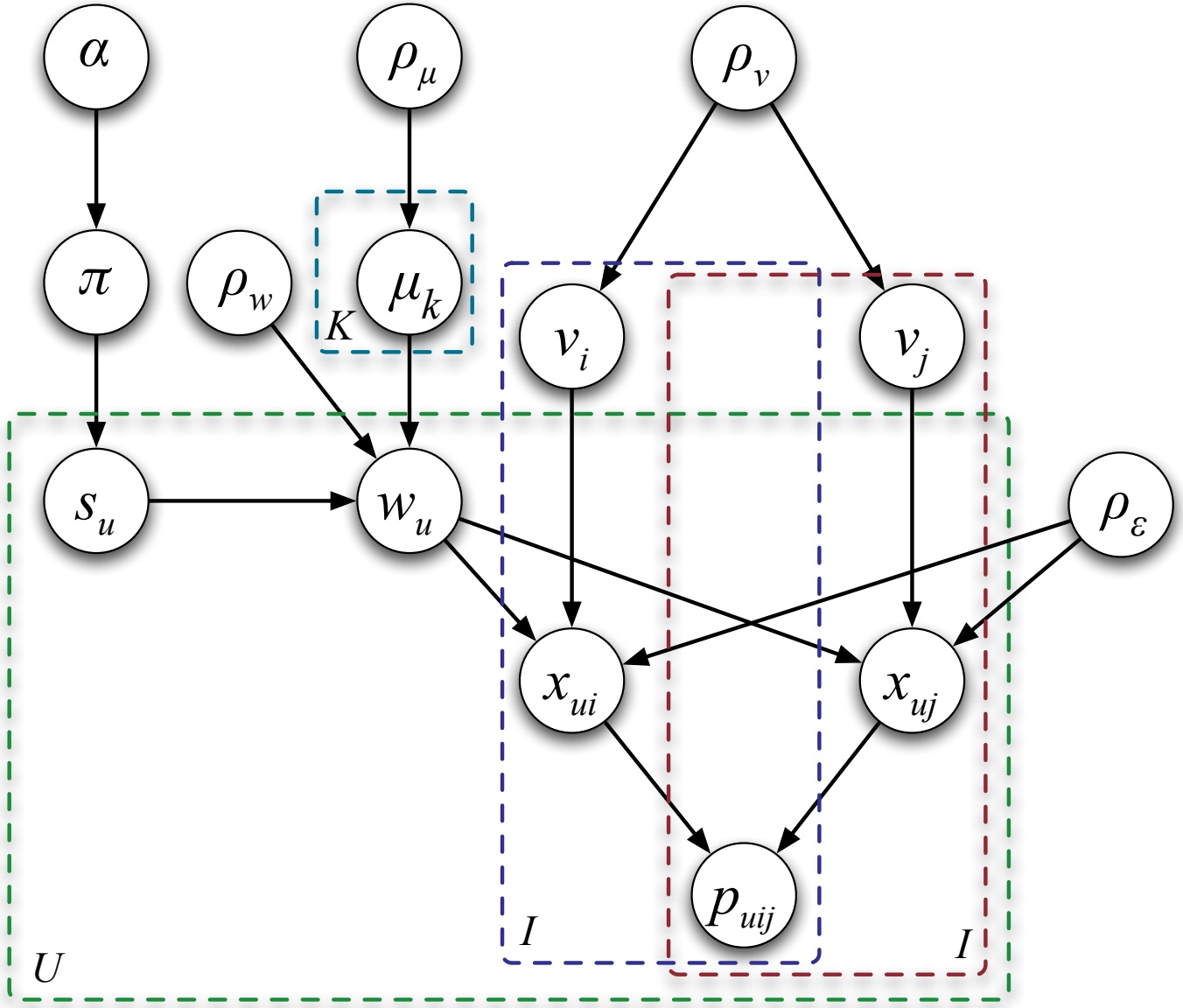
$$x_{ui} = \mathbf{w}_u^\top \mathbf{v}_i + \epsilon_{ui}$$

where $\mathbf{w}_u \in \mathcal{R}^M$, $\mathbf{v}_i \in \mathcal{R}^M$, and users cluster according to a CRP:

$$\mathbf{w}_u \sim N(\mu_{s_u}, \rho_w^{-1} I), \qquad \{s_u\} \sim CRP(\alpha)$$

Has this been done before?

Clustering Users Based on Preferences



Summary

- Many problems in preference learning can be viewed as problems of inferring latent preference or utility functions.
- We can apply probabilistic modelling approaches to these problems.
- For *supervised* preference learning we can model preference functions using Gaussian processes.
- For *unsupervised* preference learning, latent feature models (such as the IBP) and clustering models (such as the CRP) might be useful.

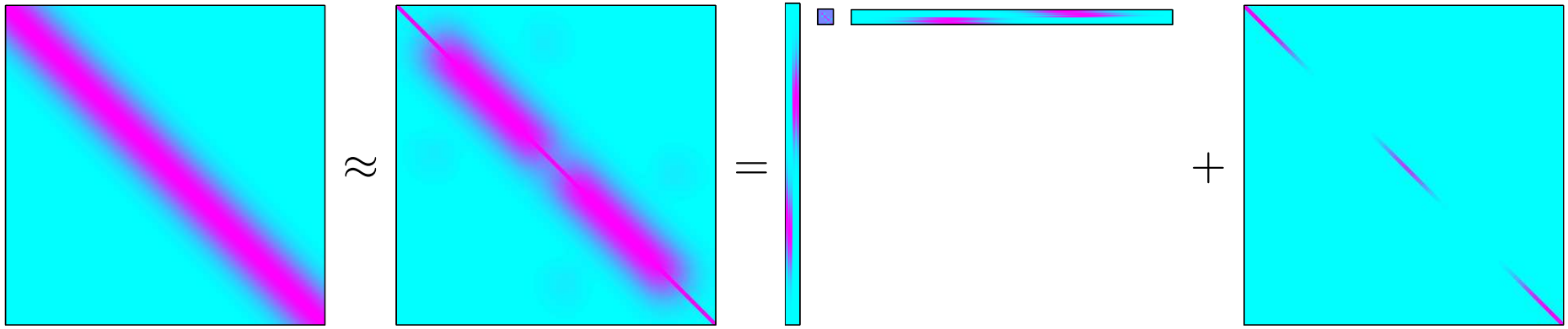
Appendix

Sparse Approximations: Speeding up GP learning

(Snelson and Ghahramani, 2006a, 2006b; Naish-Guzman and Holden 2008)

We can approximate GP through $M < N$ inducing points $\bar{\mathbf{f}}$ to obtain this Sparse Pseudo-input Gaussian process (SPGP) prior: $p(\mathbf{f}) = \int \mathrm{d}\bar{\mathbf{f}} \prod_n p(f_n|\bar{\mathbf{f}})\, p(\bar{\mathbf{f}})$

GP prior $\mathcal{N}(\mathbf{0}, \mathbf{K}_N) \approx p(\mathbf{f}) = \mathcal{N}(\mathbf{0}, \mathbf{K}_{NM}\mathbf{K}_M^{-1}\mathbf{K}_{MN} + \mathbf{\Lambda})$ SPGP prior



- SPGP covariance inverted in $\mathcal{O}(M^2N) \ll \mathcal{O}(N^3) \Rightarrow$ much faster
- SPGP = GP with non-stationary covariance parameterized by $\bar{\mathbf{X}}$
- Given data $\{\mathbf{X}, \mathbf{y}\}$ with noise σ^2, predictive mean and variance can be computed in $\mathcal{O}(M)$ and $\mathcal{O}(M^2)$ per test case respectively

Builds on a large lit on sparse GPs (see Quiñonero Candela and Rasmussen, 2006).

References

- Chu, W. and Ghahramani, Z. (2005) Preference Learning with Gaussian Processes (ICML-2005).
- Chu, W. and Ghahramani, Z. (2005b) Extensions of Gaussian processes for ranking: semi-supervised and active learning, Workshop Learning to Rank at (NIPS-18):29-34.
- Görür, D. Jäkel, F. and Rasmussen, C.E. (2006) A choice model with infinitely many latent features (ICML-2006).
- Quiñonero-Candela, J. and Rasmussen, C.E. (2005) A unifying view of sparse approximate Gaussian process regression. *Journal of Machine Learning Research* **6**:1959.
- Naish-Guzman, A. and Holden, S. (2008) The generalized FITC approximation. Advances in Neural Information Processing Systems 20:1057–1064.
- Rasmussen, C.E. and Williams, C.K.I. (2006) Gaussian Processes for Machine Learning. MIT Press.
- Snelson, E. and Ghahramani, Z. (2006a) Sparse Gaussian Processes using Pseudo-Inputs. In **Advances in Neural Information Processing Systems 18** (NIPS-2005).
- More information and code at: `http://www.gaussianprocess.org/`