Human Motion Analysis
Lecture 11: Discriminative Prediction

Raquel Urtasun

TTI Chicago

May 17, 2010

Materials used for this lecture

- See Bishop's book for linear regression.
- See references for the rest of the class.

The problem of human pose estimation

- The goal is given an image **I** to estimate the 3D location and orientation of the body parts **y**.

Pose estimation

- **Generative approaches:** focus on modeling

$$p(\phi|\mathbf{I}) = \frac{p(\mathbf{I}|\phi)p(\phi)}{p(\mathbf{I})}$$

- **Discriminative approaches:** focus on modeling directly

$$p(\phi|\mathbf{I})$$

Today we will talk about discriminative approaches.

Notation

$\mathbf{y}$ — the pose

$\mathbf{I}$ — the image

$\mathbf{x}$ — the image representation

N — number of training samples

Learning paradigm for pose estimation

- We have a set of training examples sampled i.i.d. from the joint distribution $p(\mathbf{x}, \mathbf{y})$.
- Learn a mapping from image observations $\mathbf{x}$ to pose $\mathbf{y}$.
- Main difficulties
  - What's a good image representation?
  - What's a good image similarity measure that can compare images of different sizes?
  - High dimensional inputs and high dimensional structured outputs.
  - Potentially this requires a large number of training examples. Might have computational issues.
  - The mapping is inherently multimodal.

Ambiguities of pose estimation

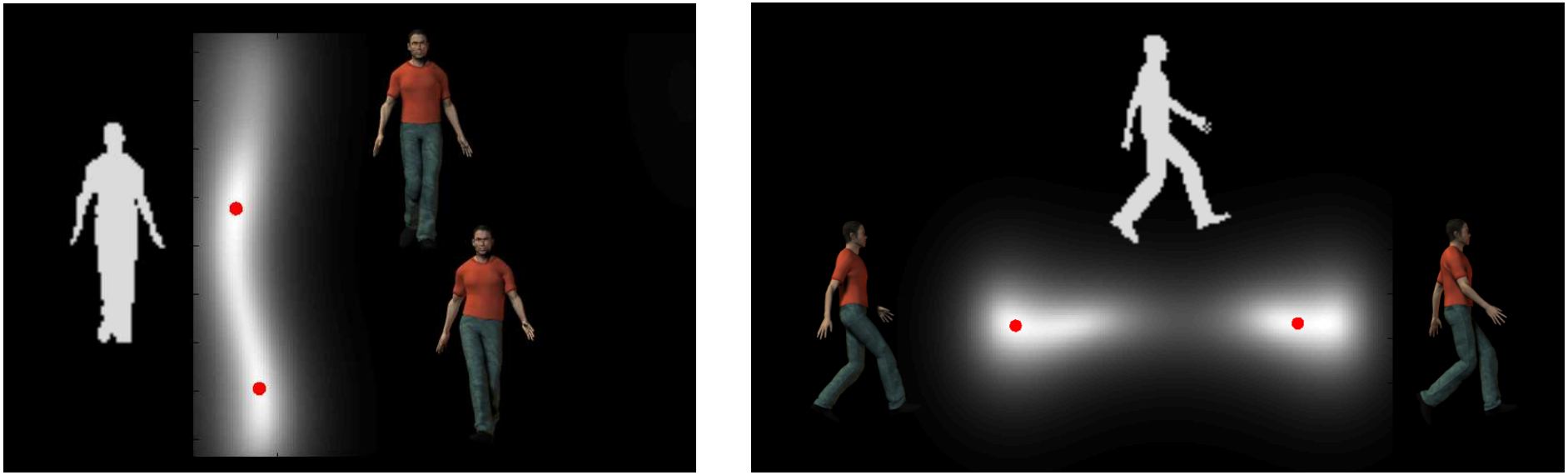


Figure: Illustration of ambiguities inherent to pose estimation from a single image (Ek 2009)

Contents of today's lecture?

We will look into discriminative approaches to pose estimation. We will focus on:

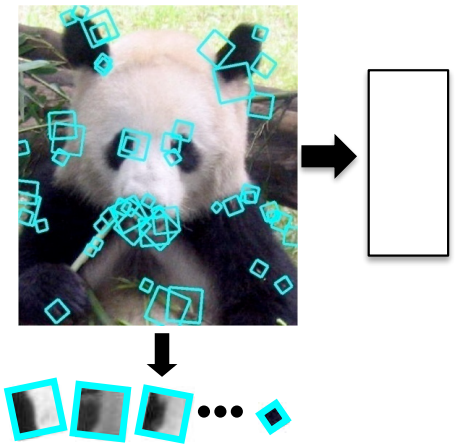
- NN approaches
- Regression
- Mixture of experts

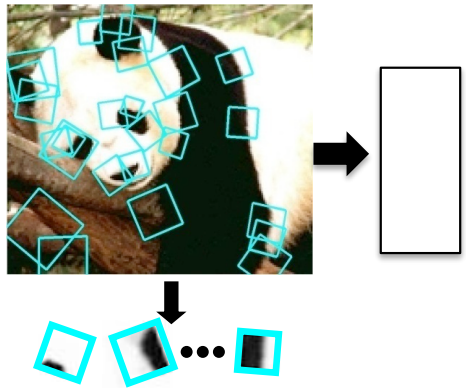
Next lecture

- Latent variable models for discriminative prediction.
- Structure prediction for discriminative prediction.
- Combination of discriminative and generative methods
- Activity recognition.

Feature types

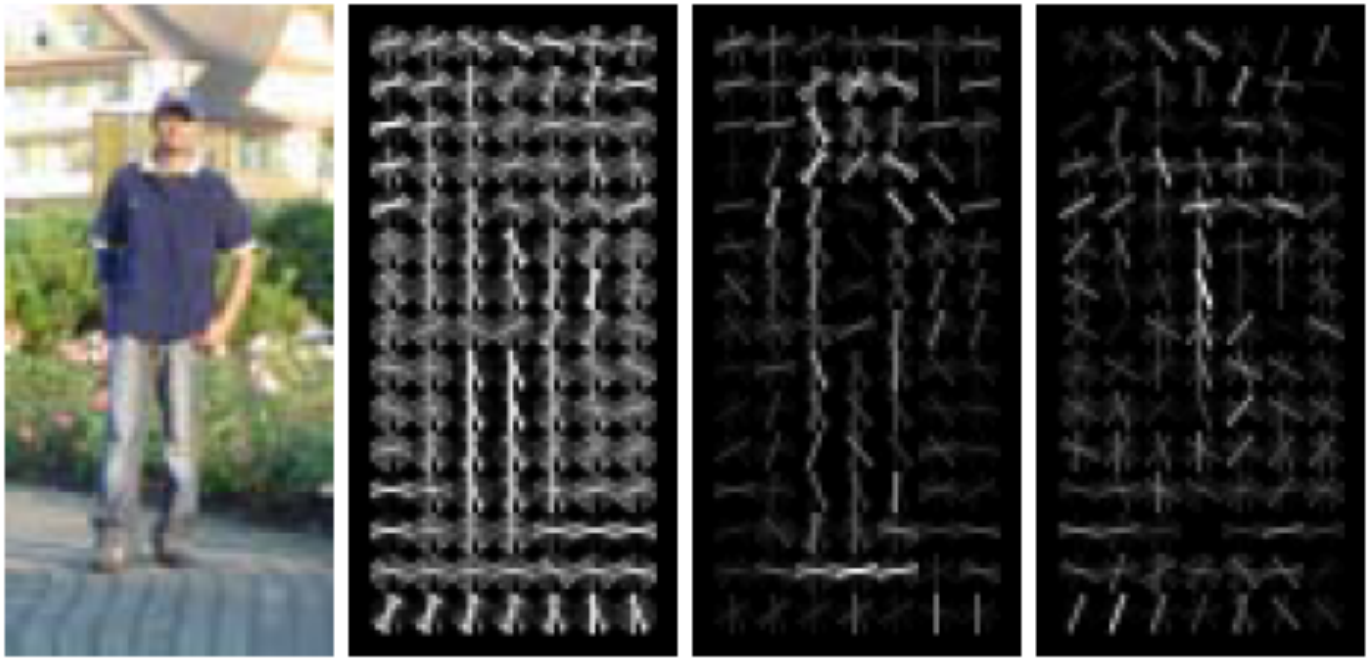
- Global vs local
- For local features: Interest points vs dense local features





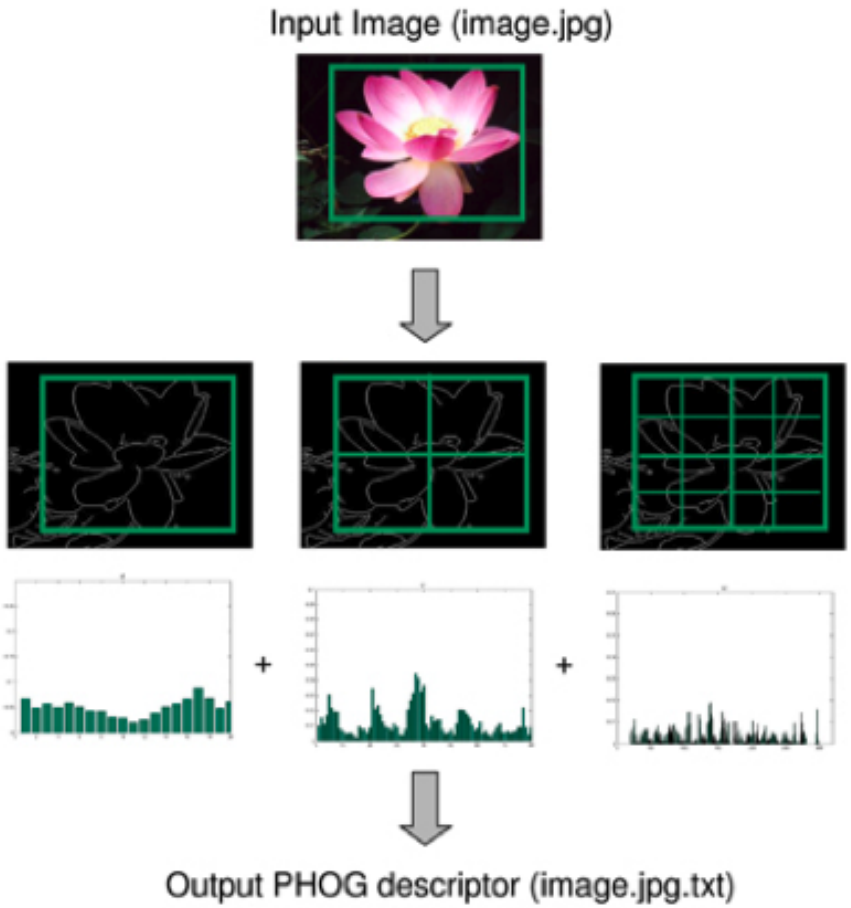
Global features: HOG

- Histogram of Oriented Gradients (Dalal and Triggs 05)



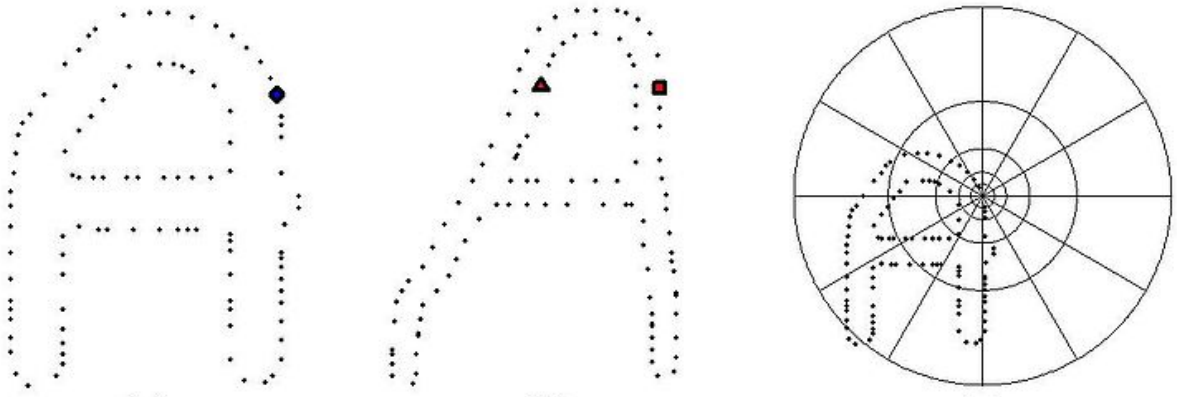
Global features: PHOG

- Pyramid of HOG (PHOG) due to (Bosch et al. 07)



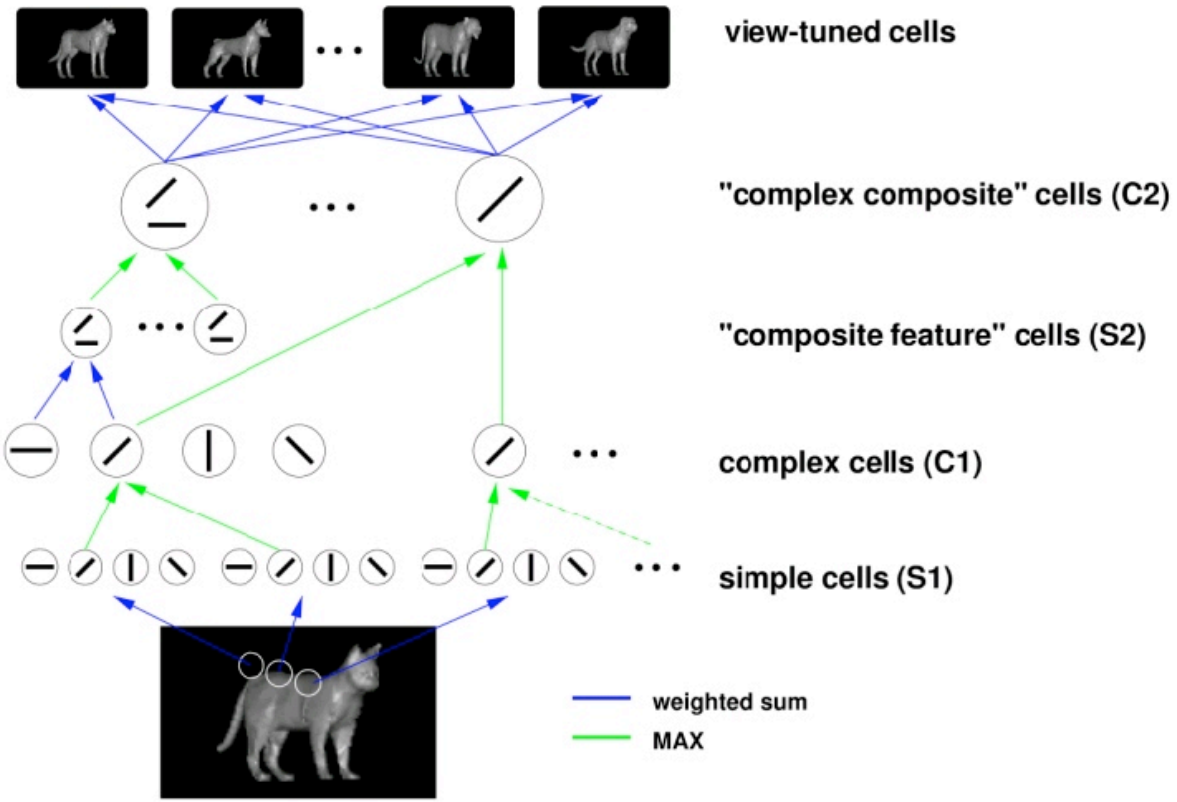
Global features: Shape context

- Shape context: distribution over relative positions on the contour (Belongie and Malik. 00)



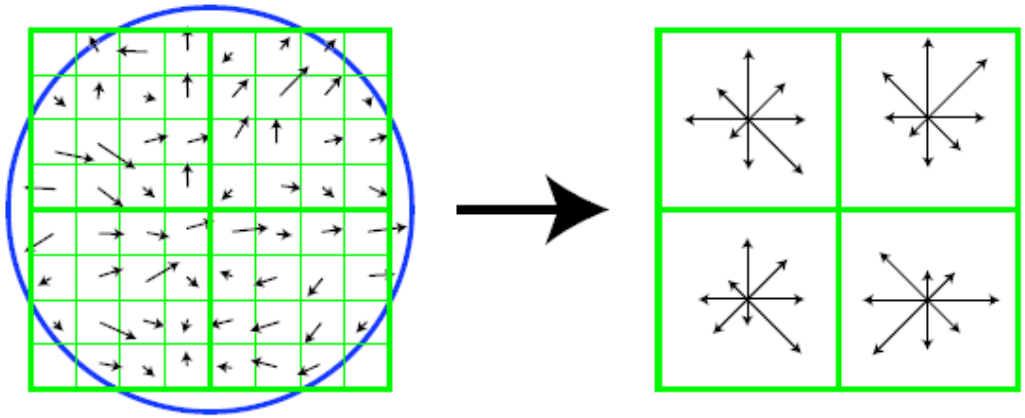
Other global features

- Gist: Widely used, particularly for scene understanding (Oliva and Torralba 06)
- HMAX (Poggio et al 99)



Local features: SIFT (Lowe 04)

- Interest points detected via differences of Gaussians at different scales.
- A descriptor is created by first computing the gradient magnitude and orientation at each image sample point in a region around the keypoint location.
- These are weighted by a Gaussian window, indicated by the overlaid circle.
- Accumulate the samples into orientation histograms over 16x16 subregions.
- This features are invariant to image translation, scaling, and rotation
- Partially invariant to illumination changes and local geometric distortion
- Used in conjunction with PCA to reduce dimensionality



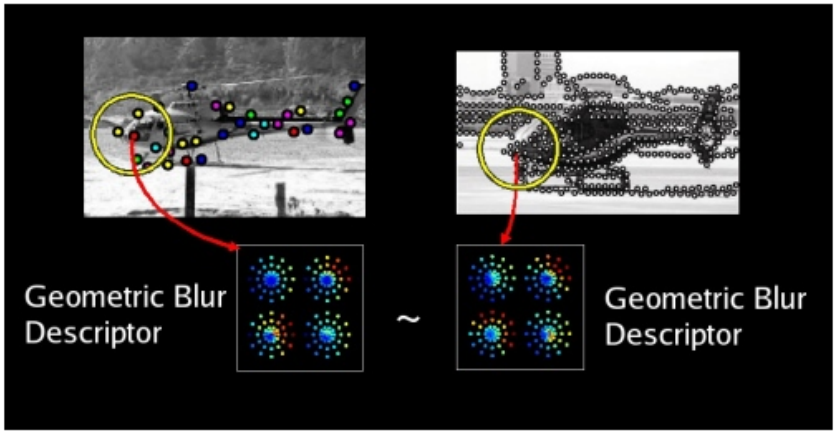
Local features: SURF

- Robust image detector and descriptor (Bay et al. in 06)
- Faster than SIFT to compute.
- It is used for real time applications
- SURF is based on sums of approximated 2D Haar wavelet responses
- Make use of integral images for efficient computation

Local features: Geometric blur (Berg et al. 05)

- It's simply an average over geometric transformations of a signal.

- It's useful operation for comparing two signals when some geometric distortion is expected
- The signal should be sparse in order for the geometric blur to produce a discriminative descriptor.
- e.g. the output of orientation tuned edge detectors after non-max suppression



What's next?

We now know how to compute global and local descriptors, but how can we compare two images of different sizes

- For global descriptors since they have the same size it's easy, compute similarity metrics
- Local descriptor is more complicated since the number of features is not necessary the same

Similarity between global descriptors

Compute kernels by using different similarity measures

- Euclidean distance between the high dimensional representations

$$d(\mathbf{x}, \mathbf{x}') = (\mathbf{x} - \mathbf{x}')^T(\mathbf{x} - \mathbf{x}')$$

- Mahalanobis: some dimensions are more important than other

$$d(\mathbf{x}, \mathbf{x}') = (\mathbf{x} - \mathbf{x}')^T\Sigma(\mathbf{x} - \mathbf{x}')$$

with Σ a PSD matrix.

Similarity between global descriptors

Compute kernels by using different similarity measures

- Euclidean distance between the high dimensional representations

$$d(\mathbf{x}, \mathbf{x}') = (\mathbf{x} - \mathbf{x}')^T(\mathbf{x} - \mathbf{x}')$$

- Mahalanobis: some dimensions are more important than other

$$d(\mathbf{x}, \mathbf{x}') = (\mathbf{x} - \mathbf{x}')^T\Sigma(\mathbf{x} - \mathbf{x}')$$

  with Σ a PSD matrix.

- Comparing Histograms, e.g., intersection kernels are well used

$$d(\mathbf{x}, \mathbf{x}') = \frac{\sum_{i=1}^{D} \min(\mathbf{x}_i, \mathbf{x}'_i)}{\sum_{i=1}^{D} \mathbf{x}_i \mathbf{x}'_i}$$

- RBF or polynomial kernels for non-linear relationships

Similarity between global descriptors

Compute kernels by using different similarity measures

- Euclidean distance between the high dimensional representations

$$d(\mathbf{x}, \mathbf{x}') = (\mathbf{x} - \mathbf{x}')^T(\mathbf{x} - \mathbf{x}')$$

- Mahalanobis: some dimensions are more important than other

$$d(\mathbf{x}, \mathbf{x}') = (\mathbf{x} - \mathbf{x}')^T\Sigma(\mathbf{x} - \mathbf{x}')$$

  with Σ a PSD matrix.

- Comparing Histograms, e.g., intersection kernels are well used

$$d(\mathbf{x}, \mathbf{x}') = \frac{\sum_{i=1}^{D} \min(\mathbf{x}_i, \mathbf{x}'_i)}{\sum_{i=1}^{D} \mathbf{x}_i \mathbf{x}'_i}$$

- RBF or polynomial kernels for non-linear relationships

Similarity between local descriptors

More complicated since different number of local features per image.
Ways of solving this

- Bag of Words
- Feature matching
- Pyramid match kernel
- Spatial pyramid

Bag of words

- Compute local features either densely or detect keypoints
- Compute a dictionary of features: either by clustering (e.g., K-means) or by learning dictionaries (e.g., mutual information, sparse coding, deep belief networks).
- Compute histograms of the local features where the bins are the codewords
- Use any distance measure between histograms.

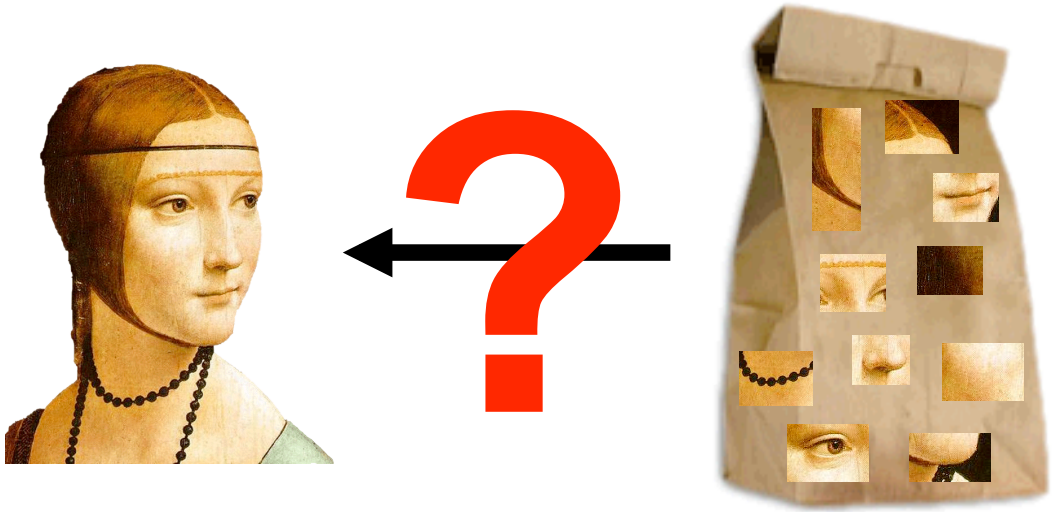


Figure: BOW illustration (Fei-Fei et al. ICCV09 tutorial)

Problems of Bag of Words

- All have equal probability for bag-of-words methods
- Location information is important
- BoW + location still doesnt give correspondence

Figure: BOW illustration (Fei-Fei et al. ICCV09 tutorial)

Matching the features

- Compare sets by computing a partial matching between their features.
- Robust to clutter, segmentation errors, occlusion.
- Very expensive computationally.

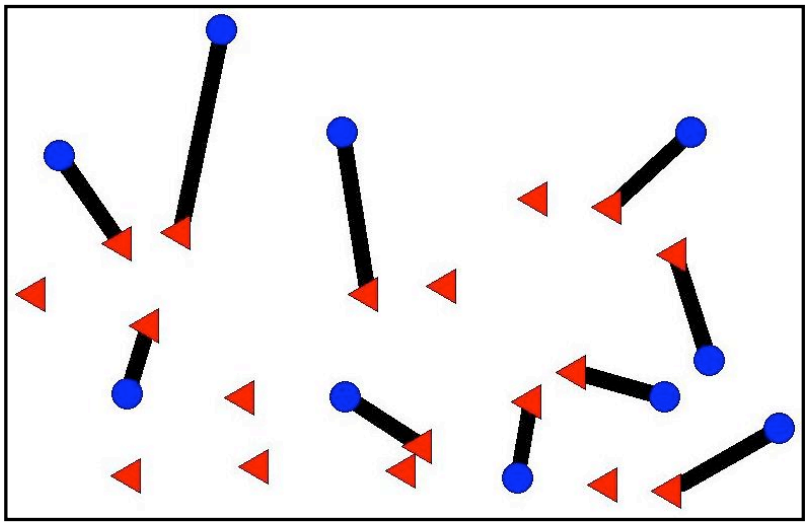


Figure: Feature matching (Grauman et al. 05)

Pyramid Match Kernel

PMK measures similarity of a partial matching between two sets:

- Place multi-dimensional, multi-resolution grid over point sets.
- Points matched at finest resolution where they fall into same grid cell.
- Approximate similarity between matched points with worst case similarity at a given level

$$\mathbf{K} = \sum_{i=0}^{L} w_i N_i$$

- N_i is the difference in histogram intersections across levels counts number of new pairs matched

$$N_i = \mathcal{I}(H_i(\mathbf{X}), H_i(\mathbf{Y})) - \mathcal{I}(H_{i-1}(\mathbf{X}), H_{i-1}(\mathbf{Y}))$$

Pyramid Match Kernel

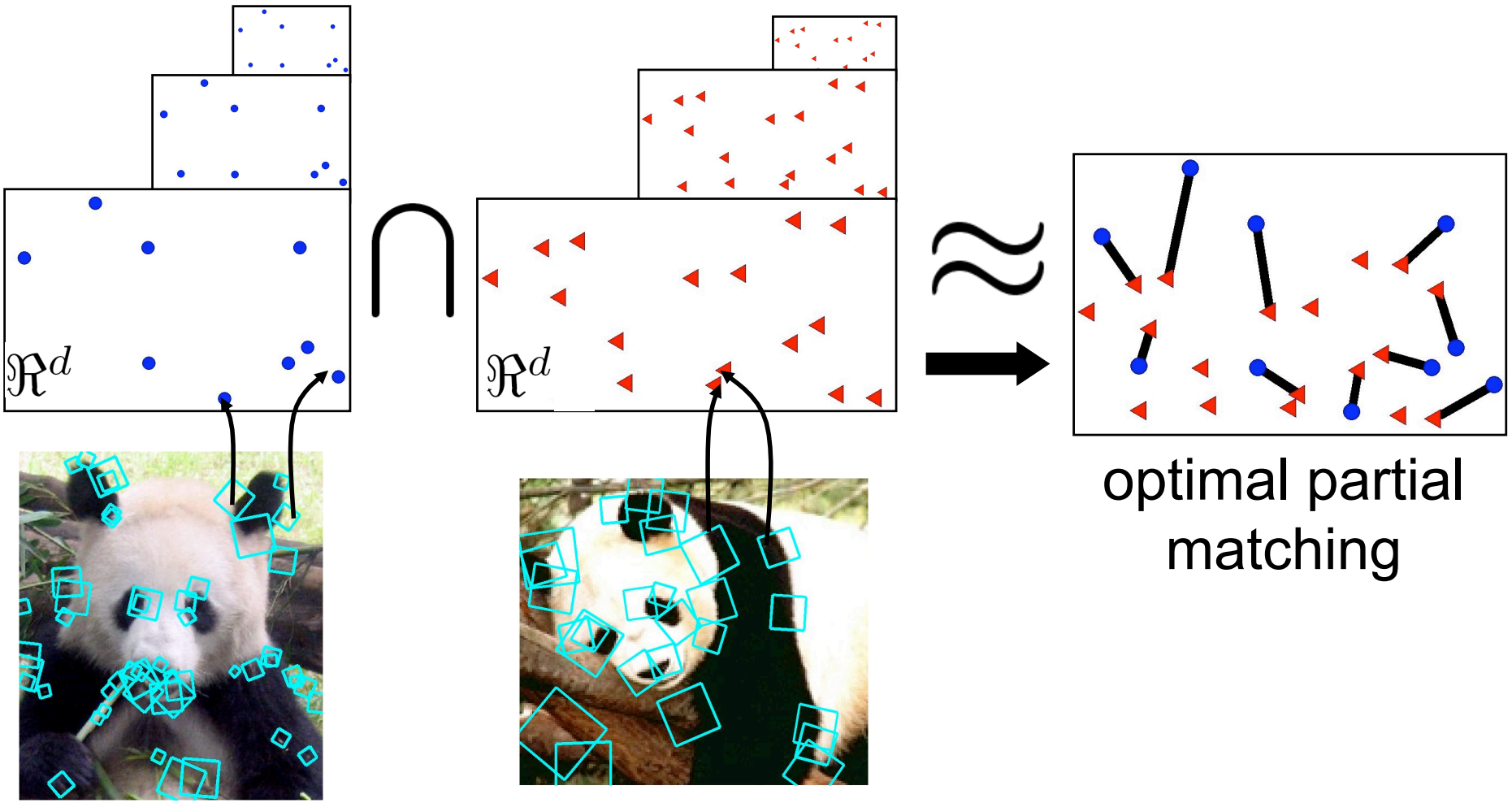


Figure: Pyramid match kernel (Grauman et al. 05)

Spatial pyramid

- Introduce spatial information by adding a pyramid.

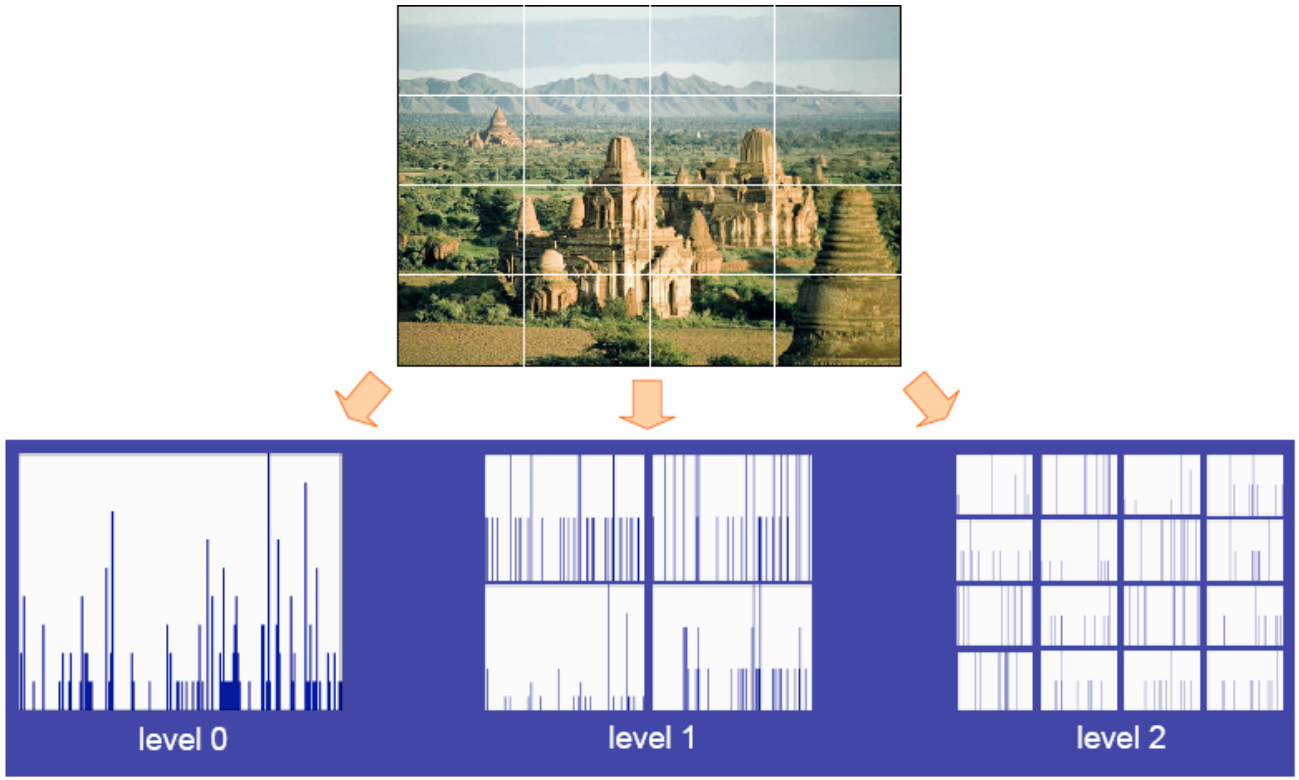


Figure: Lazebnik et al. 2006

Combining similarities: Multiple kernel learning

- A simple way to combine multiple cues and similarities is a multiple kernel learning

$$\mathbf{K} = \sum_i \alpha_i \mathbf{K}_i$$

- This has typically been addressed in a max margin framework.
- More efficient learning can be achieved with GPs.

What's next?

We have shown

- how to compute image representations
- how to compute similarities between images of different sizes

We now describe how to map from image observations to pose

- NN
- Regression
- Mixture of experts
- Structure prediction

Nearest Neighbors

- One simply searches in a database the example that it's close to the query under some metric.

Advantages:

- Simple to implement
- One car do metric learning to learn similarities

Disadvantages:

- Generalization: Amount of training data required is very large
- Then computing NN might be very slow.

Algorithms for computing the Nearest Neighbors I

Linear search

- Compute the distance from the query point to every other point in the database, keeping track of the "best so far"
- This runs in $\mathcal{O}(Nd)$ with N the number of points and d the dimensionality of the query.

Space partitioning

- Several algorithms have been developed.
- The simplest is kd-trees, which iteratively bisects the search space into two regions containing half of the points of the parent region.
- In this algorithm queries are performed traversing the tree.

Algorithms for computing the Nearest Neighbors II

Locality sensitive hashing (LSH)

- Way to compute approximate NN.
- Is a technique for grouping points in space into 'buckets' based on some distance metric operating on the points.
- Points that are close to each other under the chosen metric are mapped to the same bucket with high probability.
- Typically a function h is chosen such that
  - If $d(p,q) \leq R$ then $h(p) = h(q)$ with probability at least P_1.
  - If $d(p,q) \geq R$ then $h(p) = h(q)$ with probability at most P_2.
- A family of hash functions is interesting when $P_1 > P_2$

Efficient NN for pose estimation

- First NN approach due to Athitsos et al. 03
- Learn hash functions such as similar poses fall into the same bucket (Shakhnarovich et al 03).
- Improve results using locally weighted regression taking into account the modes of the distribution.

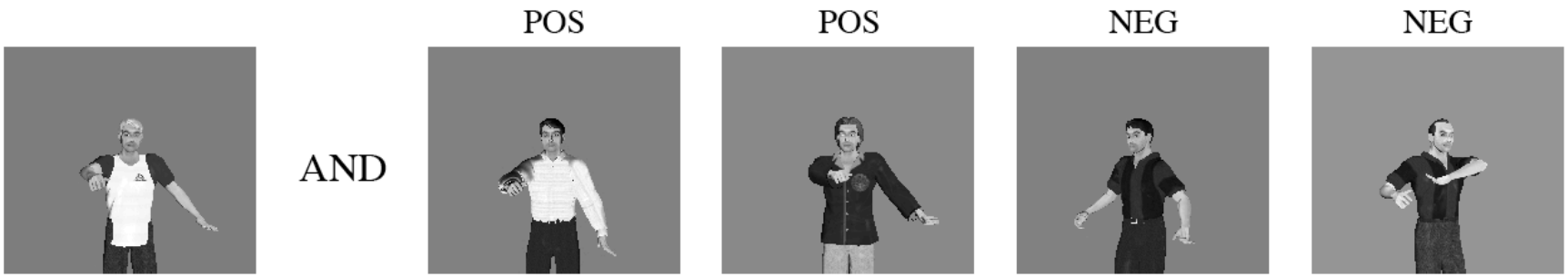
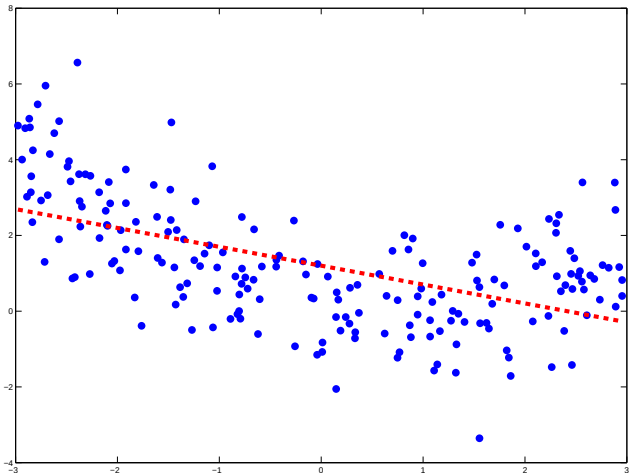


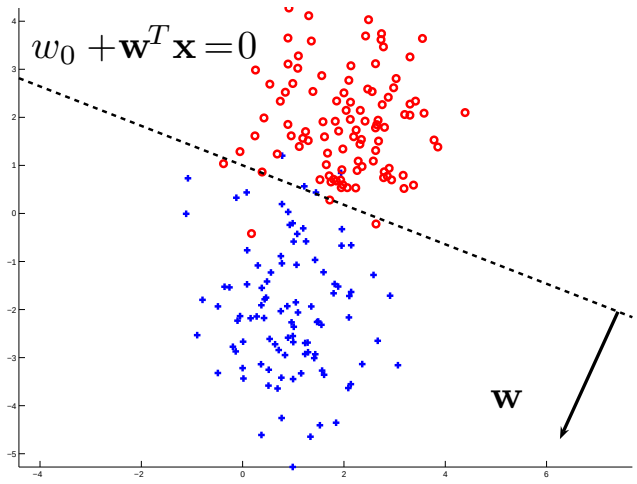
Figure: PSH (Shakhnarovich et al. 05)

PSH results

Figure: PSH results (Shakhnarovich et al. 05)

Classification vs Regression





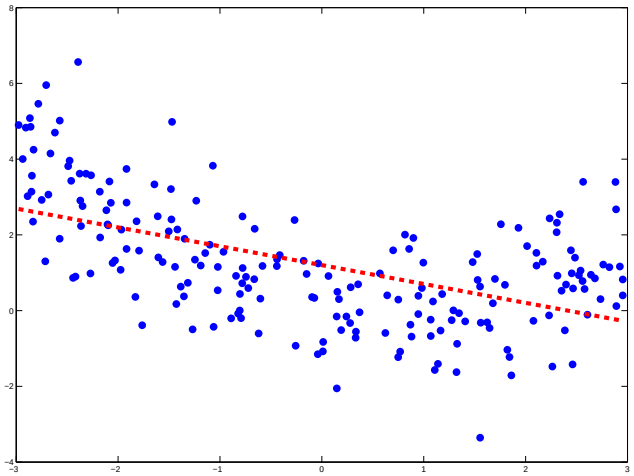
- In regression $y \in \Re$.
- In classification $y \in \{-1, 1\}$.

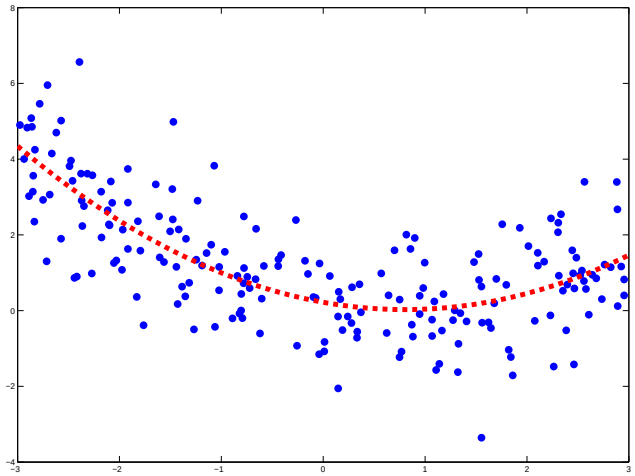
Statistical view of regression

- We will now explicitly model the randomness in the data:

$$y = f(\mathbf{x}; \mathbf{w}) + \nu$$

where the *noise* ν accounts for everything not captured by the mapping f.





Linear regression

- The simplest linear model for regression is one that involves a linear combination of the input variables

$$\mathbf{f}(\mathbf{x}, \mathbf{w}) = w_0 + w_1 x_1 + \cdots + w_D x_D$$

- We can extend this simple model to have linear combination of non linear functions

$$\mathbf{f}(\mathbf{x}, \mathbf{w}) = w_0 + \sum_{j=1}^{M} w_j \phi_j(\mathbf{x}) = \mathbf{w}^T \boldsymbol{\phi}(\mathbf{x})$$

  where $\mathbf{w} = (w_0, \cdots, w_{M-1})^T$, $\boldsymbol{\phi} = (\phi_0, \cdots, \phi_{M-1})$ and $\phi_0 = 1$ encodes the bias.

- If ϕ is non linear, we allow to have a non linear function of the input $\mathbf{x}$.
- Since it's linear in $\mathbf{w}$ learning is simplified.

Types of basis functions

- **Polynomial basis:** problem of tis basis is that changes in one of the basis affect globally.

$$\phi_j(x) = x^j$$

- **Gaussian** basis functions: are local

$$\phi_j(x) = \exp\{-\frac{(x-\mu_j)^2}{2s^2}\}$$

- **Sigmoidal** basis functions

$$\phi_j(x) = \sigma\left(\frac{x-\mu_j}{s}\right) \qquad \sigma(a) = \frac{1}{1+\exp(-a)}$$

where σ is the logistic sigmoid function.

- **Hyperbolic tangent**

$$tanh(a) = 2\sigma(a) - 1$$

- **Fourier basis:** leads to an expansion in sinusoidal functions.
- **Wavelets basis:** which are localized in space and frequency.

Examples of basis functions

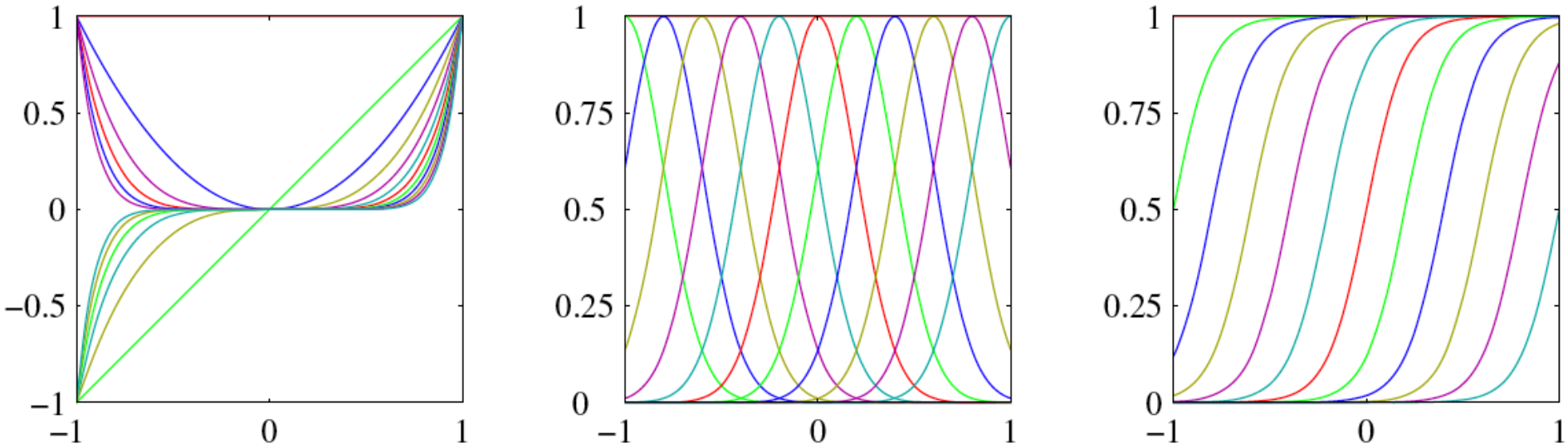


Figure: Examples of basis functions, (left) polynomials , (center) Gaussians and (right) sigmoidal (Bishop book).

Noisy observations

- We assume that the target is given by a deterministic function $\mathbf{f}(\mathbf{x}, \mathbf{w})$ and i.i.d. Gaussian noise ν

$$y = f(\mathbf{x}; \mathbf{w}) + \nu$$

- We can write the likelihood as

$$p(y|\mathbf{x}, \mathbf{w}, \beta) = \mathcal{N}(y|\mathbf{f}(\mathbf{x}, \mathbf{w}), \beta^{-1})$$

- Note that if we assume a squared loss function, then the optimal prediction, for a new value of x, will be given by the conditional mean of the target variable

$$E_{p(y|\mathbf{x})}[y|\mathbf{x}] = \int y p(y|\mathbf{x}) dy = \mathbf{f}(\mathbf{x}, \mathbf{w})$$

- This implies that $p(y|\mathbf{x})$ is unimodal.

Learning from training data

- Consider we have training data $\mathbf{X} = \{\mathbf{x}_1, \cdots, \mathbf{x}_N\}$ with corresponding target values $\mathbf{y} = [y_1, \cdots, y_N]$ i. i. d. sampled from $p(\mathbf{x}, \mathbf{y})$, we can write

$$p(\mathbf{y}|\mathbf{X}, \mathbf{w}, \beta) = \prod_{n=1}^{N} \mathcal{N}(y_n|\mathbf{w}^T \phi(\mathbf{x}_n), \beta^{-1})$$

- We can learn the model by minimizing the minus log likelihood $E_D(\mathbf{w}) = -\ln p(\mathbf{y}|\mathbf{X})$

$$\mathbf{w}^* = \underset{\mathbf{w}}{\operatorname{argmin}} \frac{1}{N} \sum_{i=1}^{N} (\mathbf{w}^T \phi(\mathbf{x}_i) - y_i)^2$$

- The maximum likelihood is the least squares solution.
- Necessary condition is that the derivative with respect to $\mathbf{w}$ must be zero.

Least squares estimation

- The derivative can be computed as

$$\nabla \ln p(\mathbf{y}|\mathbf{w}, \beta) = \sum_{i=1}^{N} \{y_n - \mathbf{w}^T \phi(\mathbf{x}_n)\} \phi(\mathbf{x}_n)^T$$

- Setting the gradient to zero this gives

$$0 = \sum_{n=1}^{N} y_n \phi(\mathbf{x}_n)^T - \mathbf{w}^T \left(\sum_{n=1}^{N} \phi(\mathbf{x}_n) \phi(\mathbf{x}_n)^T \right)$$

- Solving this we obtain

$$\mathbf{w}_{ML} = (\mathbf{\Phi}^T \mathbf{\Phi})^{-1} \mathbf{\Phi} \mathbf{y}$$

with $\mathbf{\Phi}^\dagger = (\mathbf{\Phi}^T \mathbf{\Phi})^{-1} \mathbf{\Phi}$ the Moore-Penrose pseudo-inverse and

$$\mathbf{\Phi} = \begin{bmatrix} \phi_0(\mathbf{x}_1) & \phi_1(\mathbf{x}_1) & \cdots & \phi_d(\mathbf{x}_1) \\ \vdots & & \vdots & \\ \phi_0(\mathbf{x}_N) & \phi_1(\mathbf{x}_N) & \cdots & \phi_d(\mathbf{x}_N) \end{bmatrix},$$

Regularized least squares

- Add a regularization term to an error function in order to control over-fitting

$$E_D(\mathbf{w}) + \lambda E_W(\mathbf{w})$$

  with λ the regularization coefficient
- Multiple types of regularization.
- **Ridge regression** uses an L_2 which is a Gaussian prior

$$E_w(\mathbf{w}) = \frac{1}{2}\mathbf{w}^T\mathbf{w}$$

- The solution to this problem can also be obtain in closed form

$$\mathbf{w} = (\lambda\mathbf{I} + \mathbf{\Phi}^T\mathbf{\Phi})^{-1}\mathbf{\Phi}\mathbf{y}$$

More general regularizers

- A more general regularizer takes the form

$$\frac{1}{2}\sum_{i=1}^{N}(\mathbf{w}^T\phi(\mathbf{x}_i) - y_i)^2 + \frac{\lambda}{2}\sum_{j=1}^{M}||w_j||_q^q$$

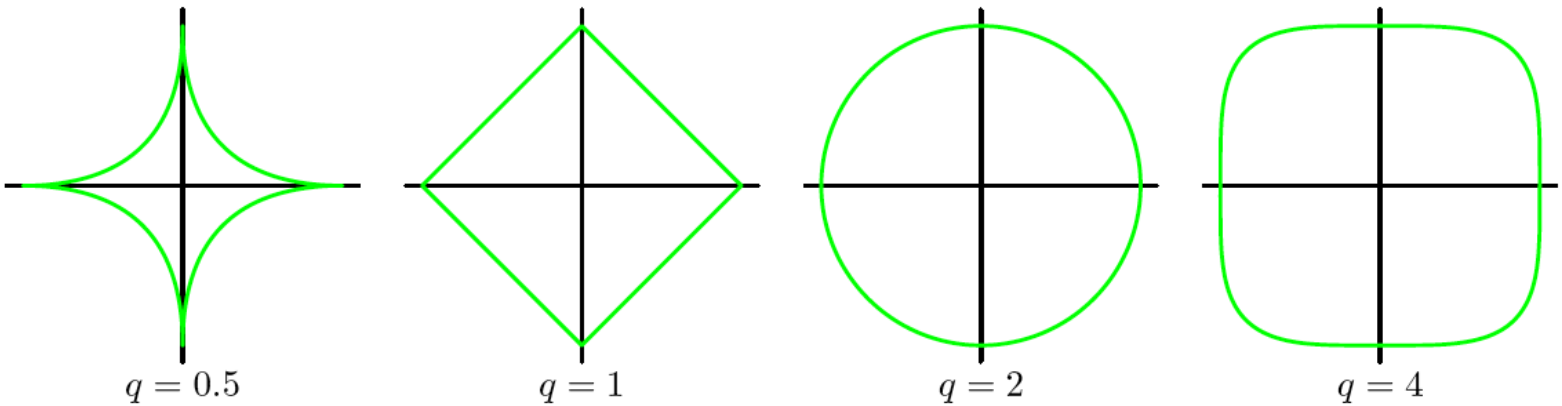


Figure: Contours of the regularization term for various values of q (Bishop book).

Why does the lasso result in sparse solutions?

- When $q = 1$ we have the **Lasso** which enforces sparsity

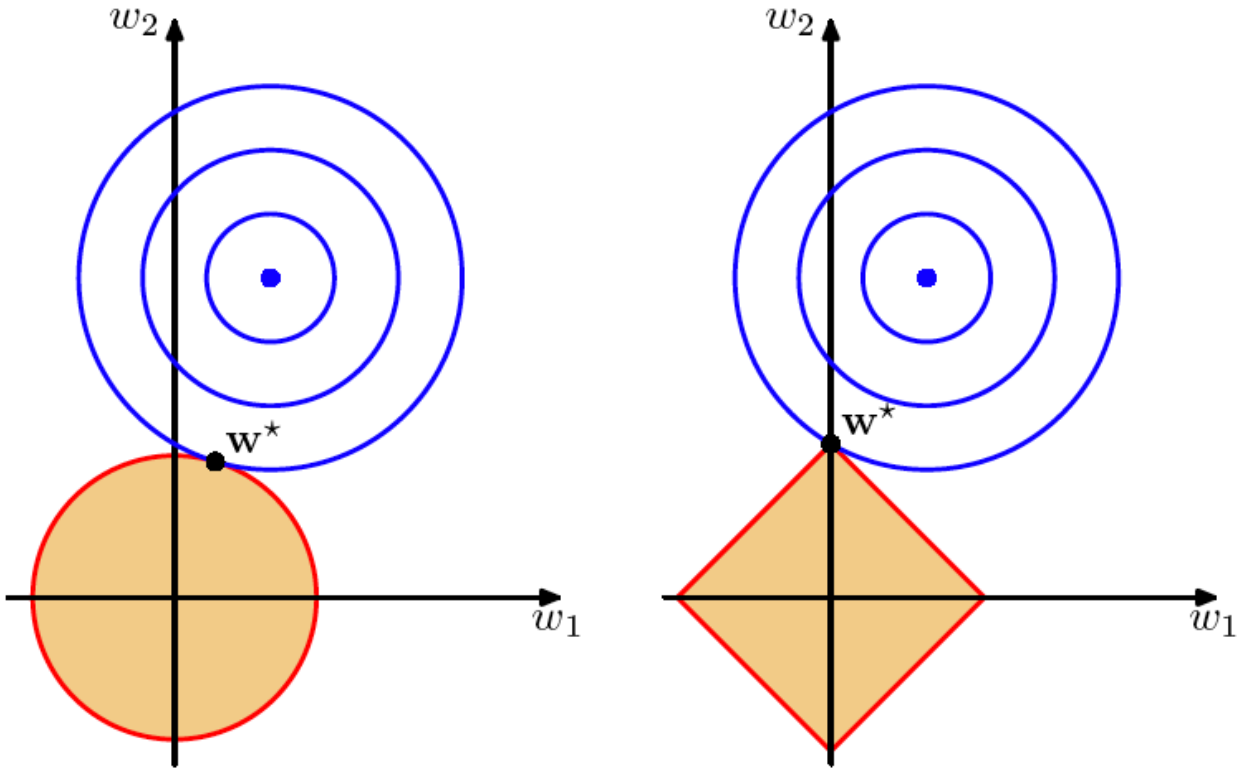


Figure: Contours of the unregularized error function (blue) along with the constraint region for (left) the quadratic regularizer $q = 2$ (right) the lasso regularizer $q = 1$ (Bishop book).

Prediction in these models

- The posterior mean can be computed as

$$f(\mathbf{x}, \mathbf{w}) = \mathbf{w}^T \phi(\mathbf{x})$$

- Equivalently the **Representer theorem**

$$f(\mathbf{x}, \boldsymbol{\alpha}) = \sum_{i=1}^{N} \alpha_i k(\mathbf{x}_i, \mathbf{x})$$

Gaussian Process (GP)

Prior for Functions

- Probability Distribution over Functions
- Functions are infinite dimensional.
  - Prior distribution over *instantiations* of the function: finite dimensional objects.
  - Can prove by induction that GP is 'consistent'.
- Mean and Covariance Functions
- Instead of mean and covariance matrix, GP is defined by mean function and covariance function.
  - Mean function often taken to be zero or constant.
  - Covariance function must be *positive definite*.
  - Class of valid covariance functions is the same as the class of *Mercer kernels*.

Gaussian Processes II

Zero mean Gaussian Process

- A (zero mean) Gaussian process likelihood is of the form

$$p\left(\mathbf{y}|\mathbf{X}\right) = N\left(\mathbf{y}|\mathbf{0}, \mathbf{K}\right),$$

  where $\mathbf{K}$ is the covariance function or *kernel*.
- The *linear kernel* with noise has the form

$$\mathbf{K} = \mathbf{X}\mathbf{X}^{\mathrm{T}} + \sigma^2\mathbf{I}$$

- Priors over non-linear functions are also possible.
  - To see what functions look like, we can sample from the prior process.

Covariance Samples

`demCovFuncSample`

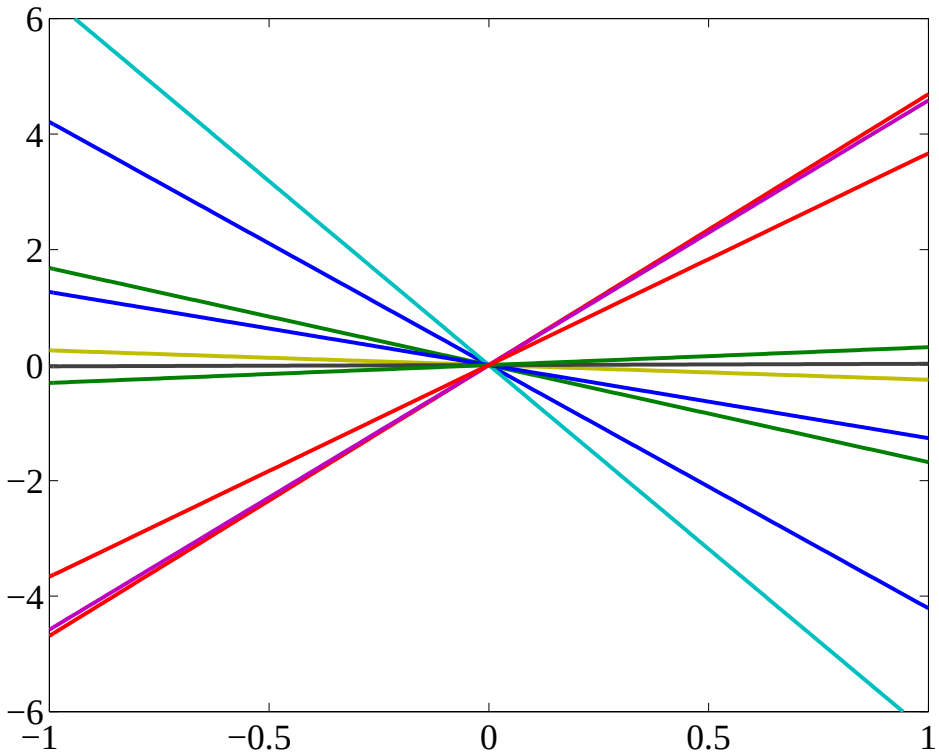


Figure: linear kernel, $\mathbf{K} = \mathbf{X}\mathbf{X}^{\mathrm{T}}$

Covariance Samples

`demCovFuncSample`

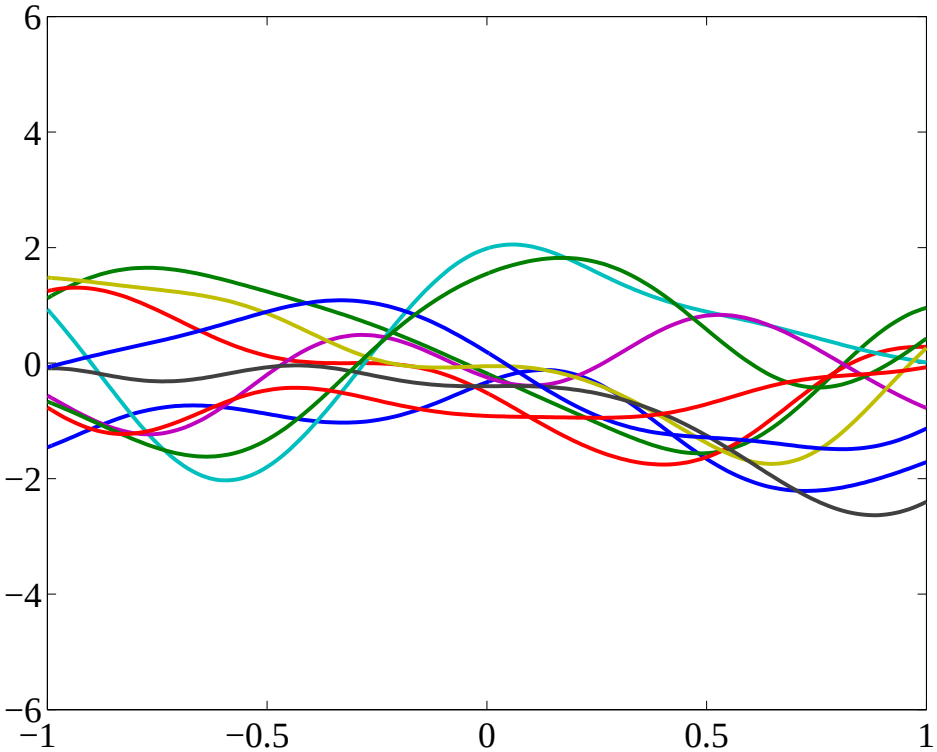


Figure: RBF kernel with $l = 10$, $\alpha = 1$

Covariance Samples

`demCovFuncSample`

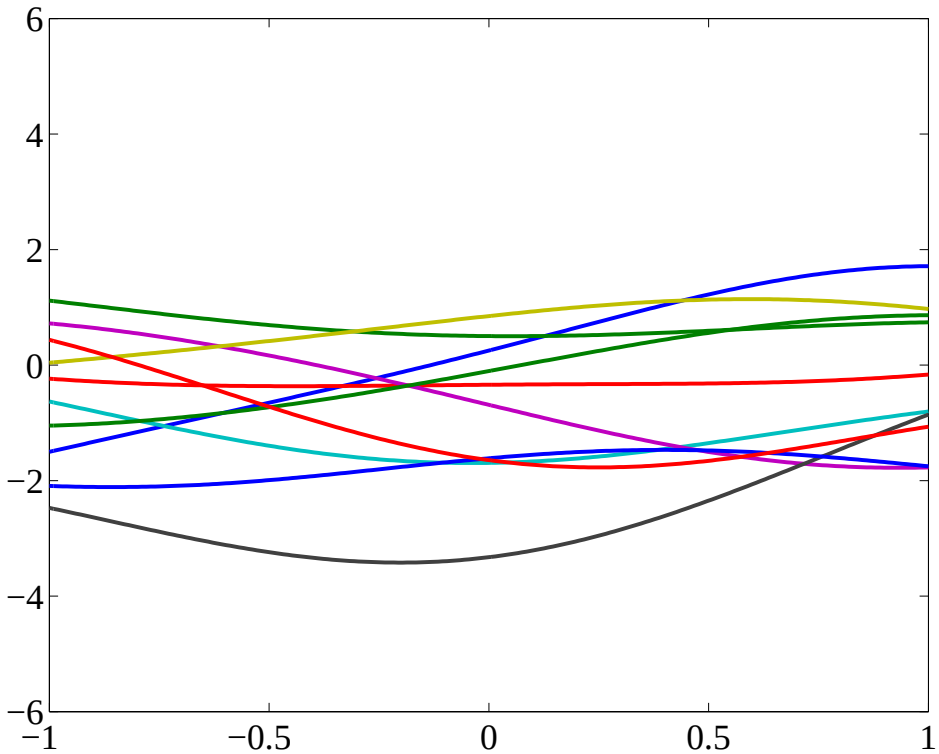


Figure: RBF kernel with $l = 1$, $\alpha = 1$

Covariance Samples

`demCovFuncSample`

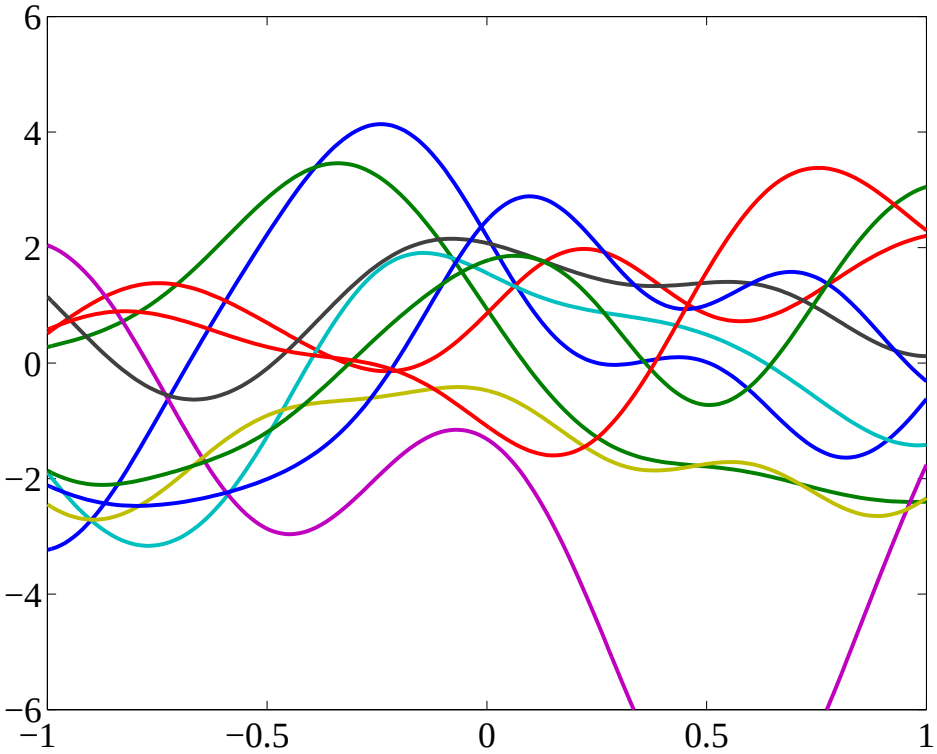


Figure: RBF kernel with $l = 0.3$, $\alpha = 4$

Covariance Samples

`demCovFuncSample`

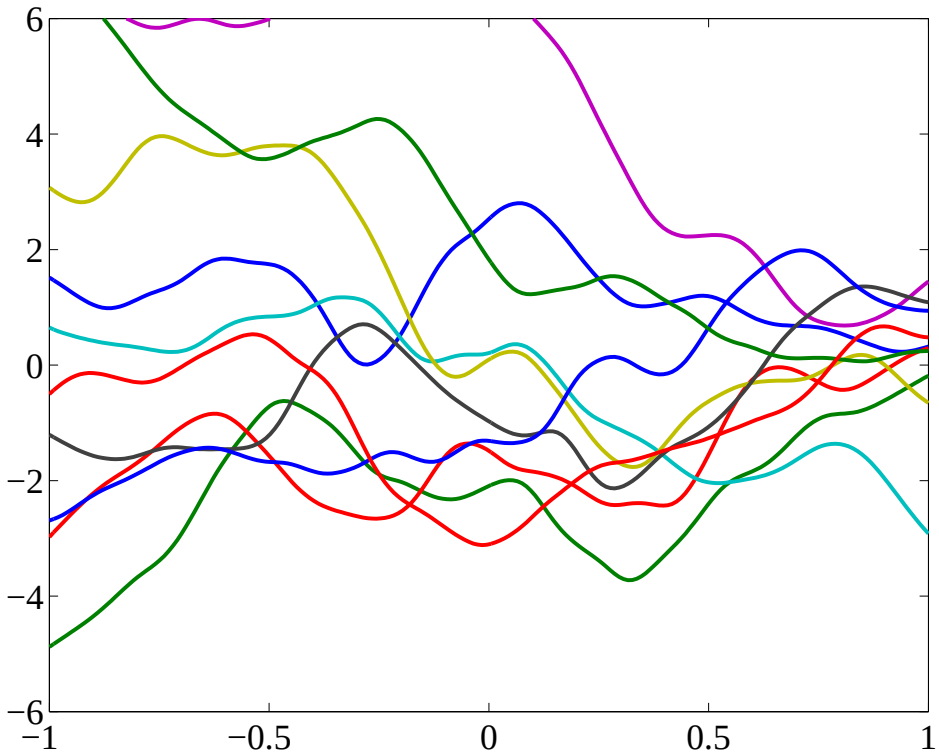


Figure: MLP kernel with $\alpha = 8$, $w = 100$ and $b = 100$

Covariance Samples

`demCovFuncSample`

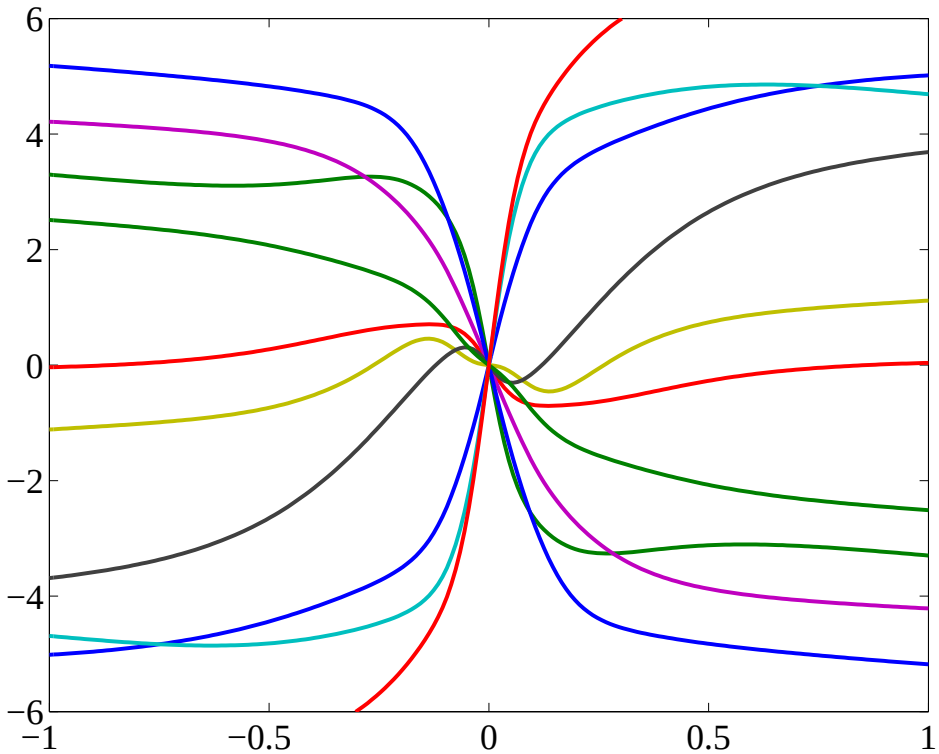


Figure: MLP kernel with $\alpha = 8$, $b = 0$ and $w = 100$

Covariance Samples

`demCovFuncSample`

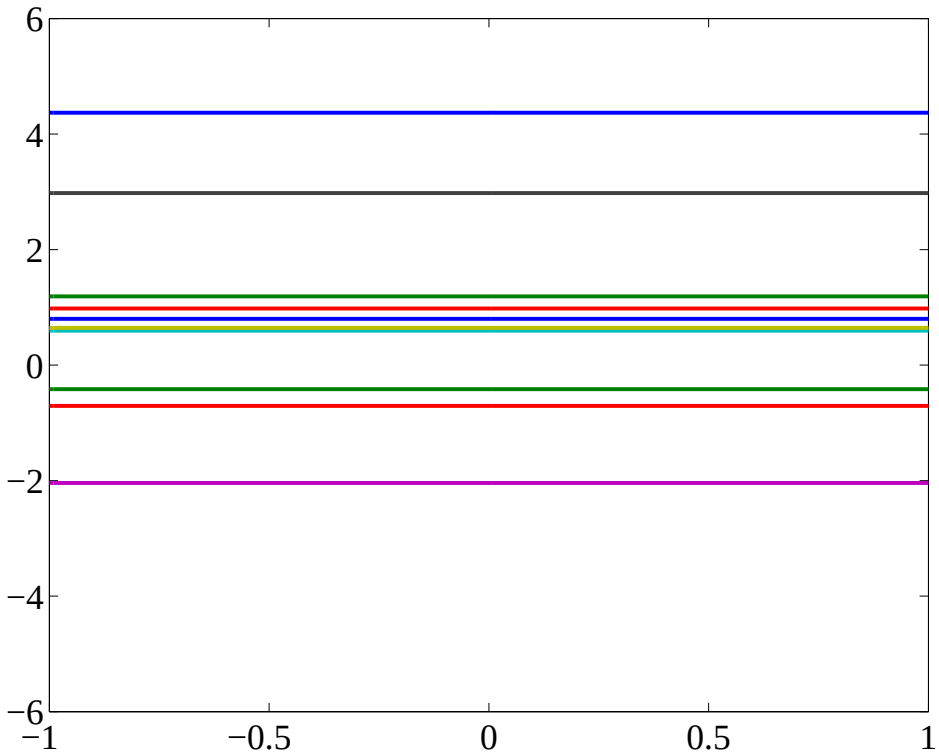


Figure: bias kernel with $\alpha = 1$ and

Covariance Samples

`demCovFuncSample`

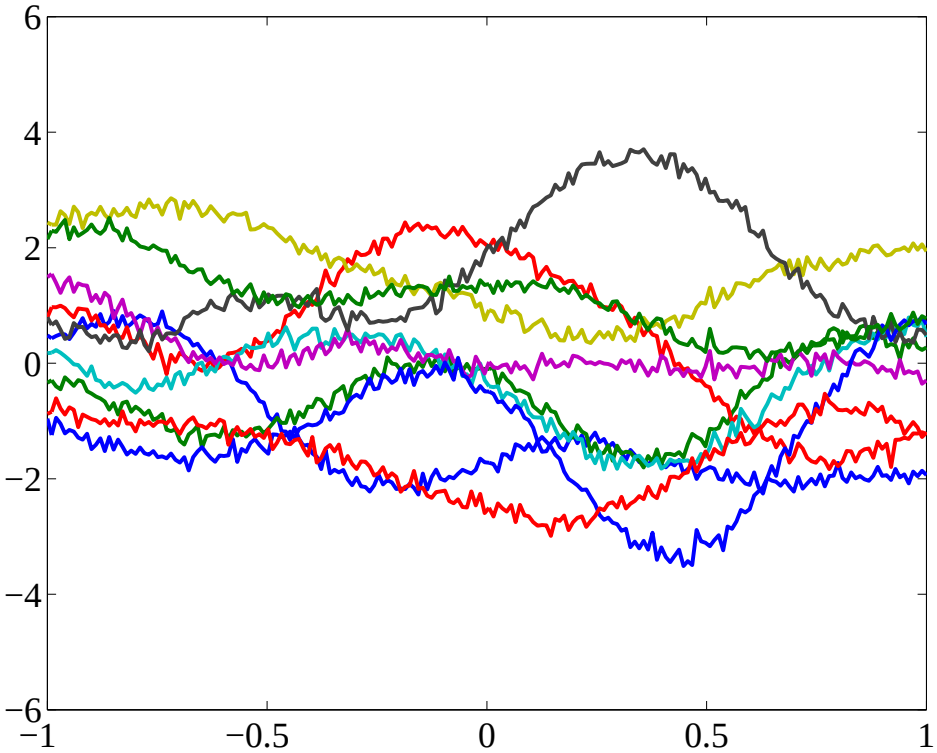


Figure: summed combination of: RBF kernel, $\alpha = 1$, $l = 0.3$; bias kernel, $\alpha = 1$; and white noise kernel, $\beta = 100$

Gaussian Process Regression

Posterior Distribution over Functions

- Gaussian processes are often used for regression.
- We are given a known inputs $\mathbf{X}$ and targets $\mathbf{Y}$.
- We assume a prior distribution over functions by selecting a kernel.
- Combine the prior with data to get a *posterior* distribution over functions.

Gaussian Process Regression

`demRegression`

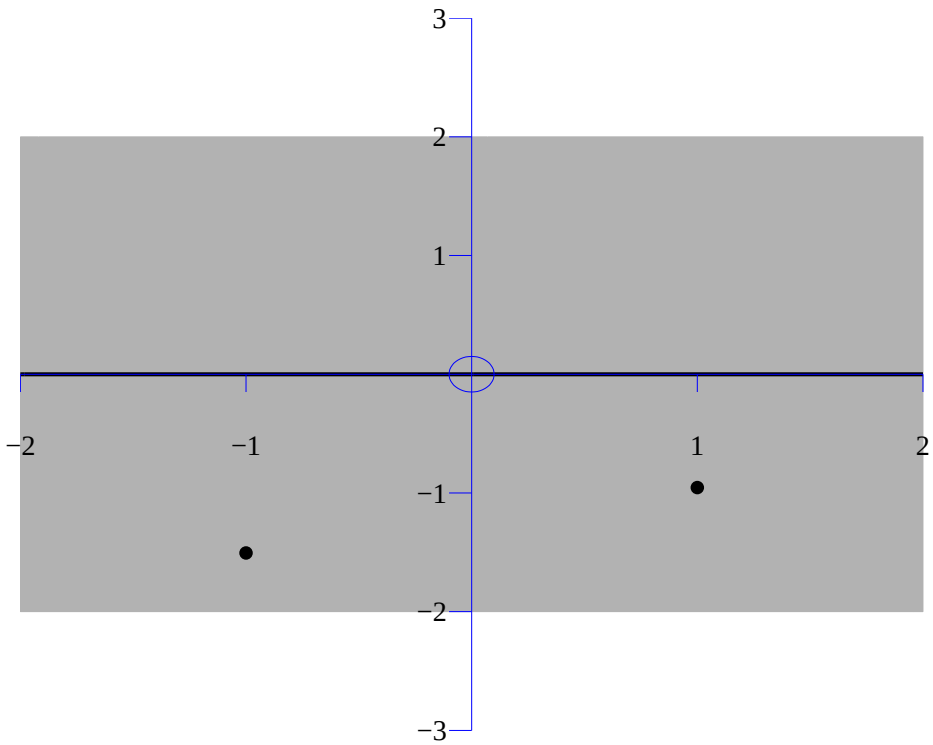


Figure: Examples include WiFi localization, C14 callibration curve.

Gaussian Process Regression

`demRegression`

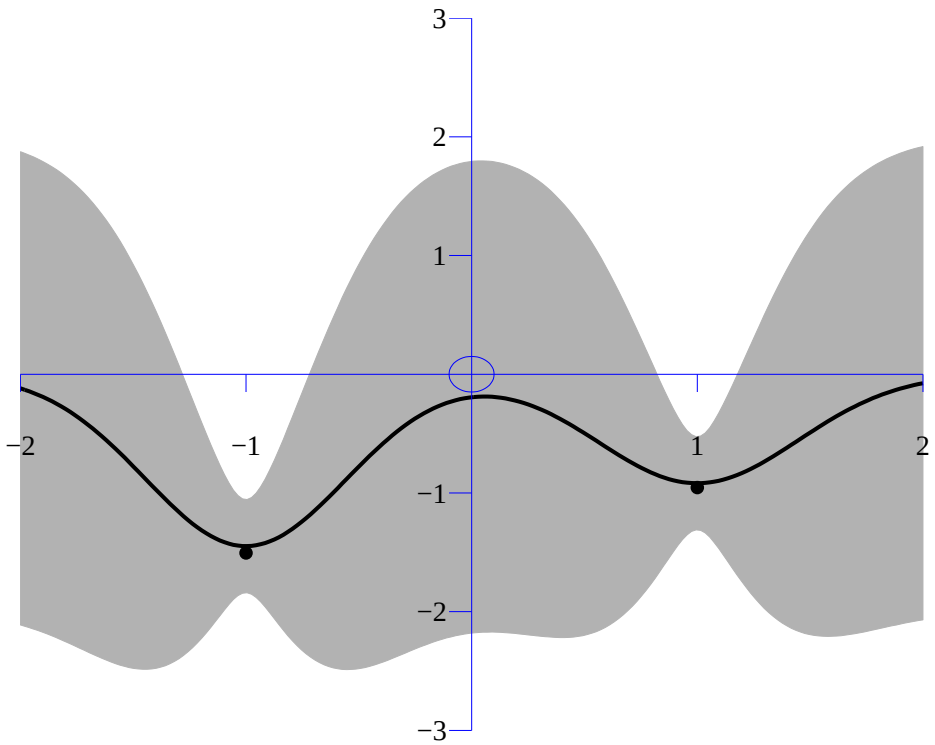


Figure: Examples include WiFi localization, C14 callibration curve.

Gaussian Process Regression

`demRegression`

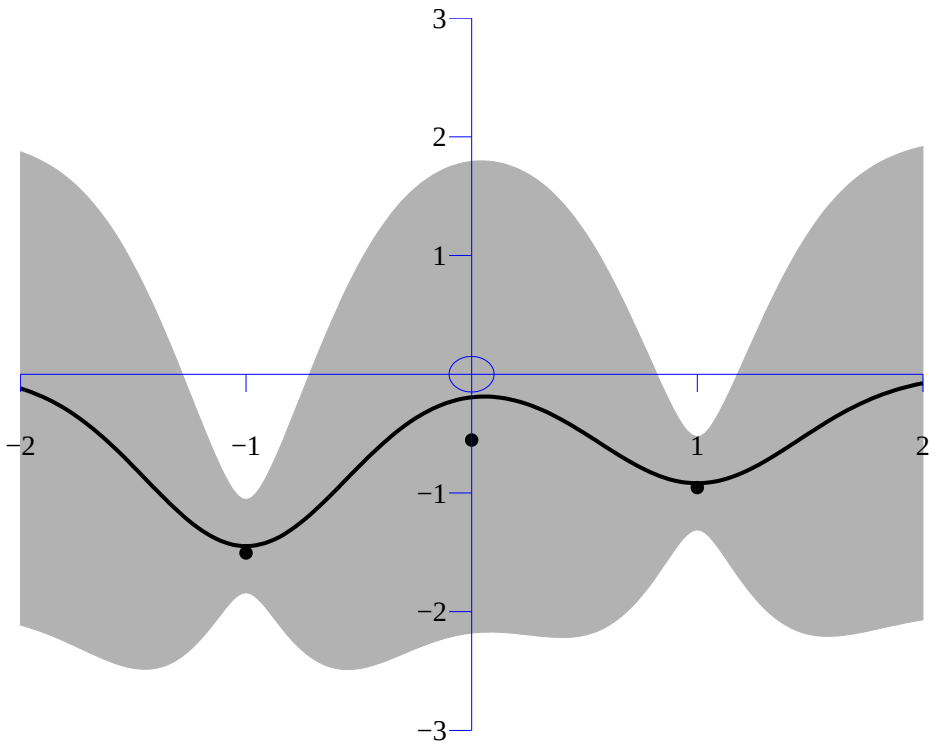


Figure: Examples include WiFi localization, C14 callibration curve.

Gaussian Process Regression

`demRegression`

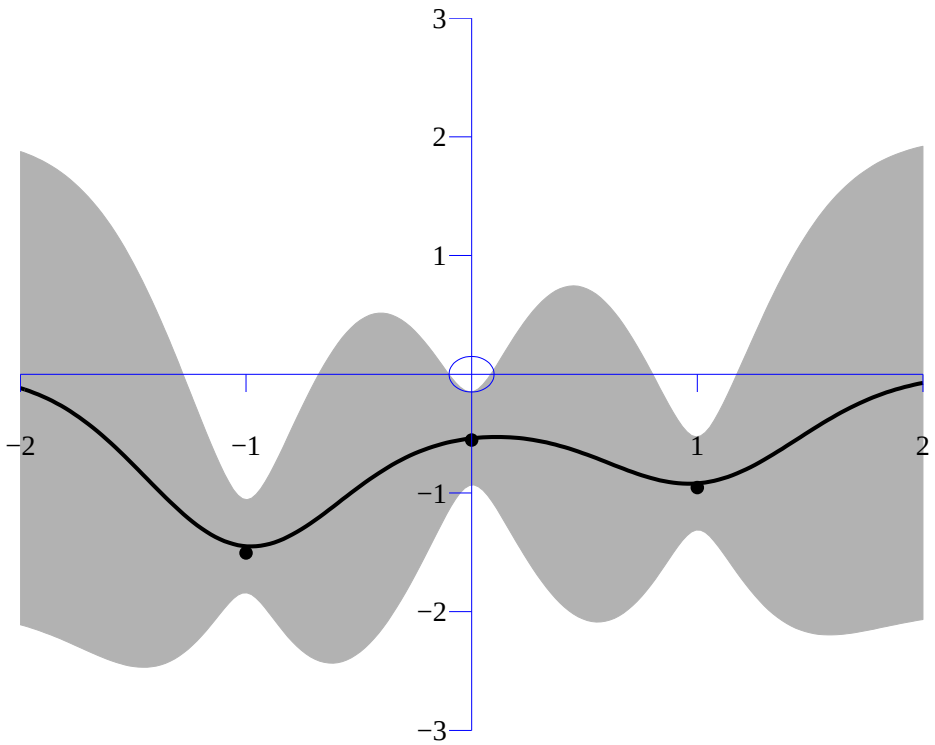


Figure: Examples include WiFi localization, C14 callibration curve.

Gaussian Process Regression

`demRegression`

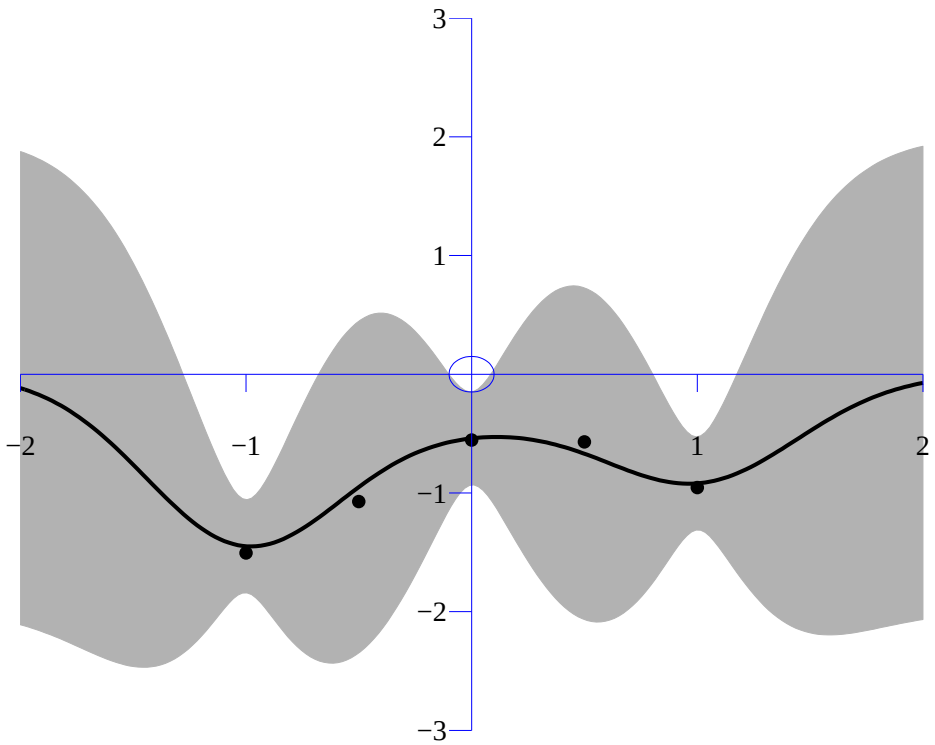


Figure: Examples include WiFi localization, C14 callibration curve.

Gaussian Process Regression

`demRegression`

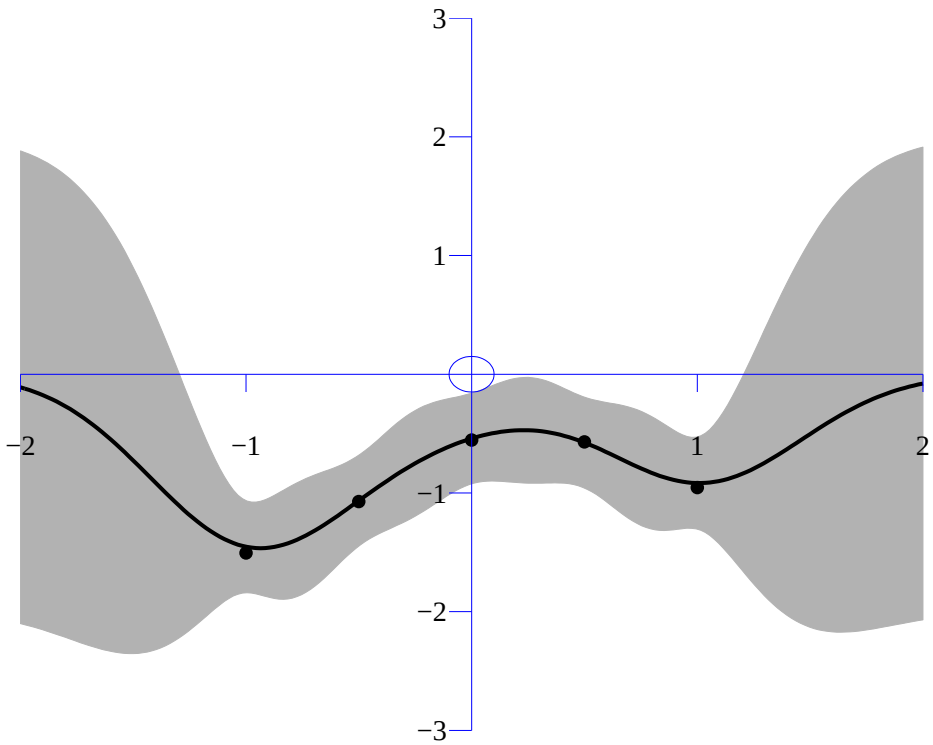


Figure: Examples include WiFi localization, C14 callibration curve.

Gaussian Process Regression

`demRegression`

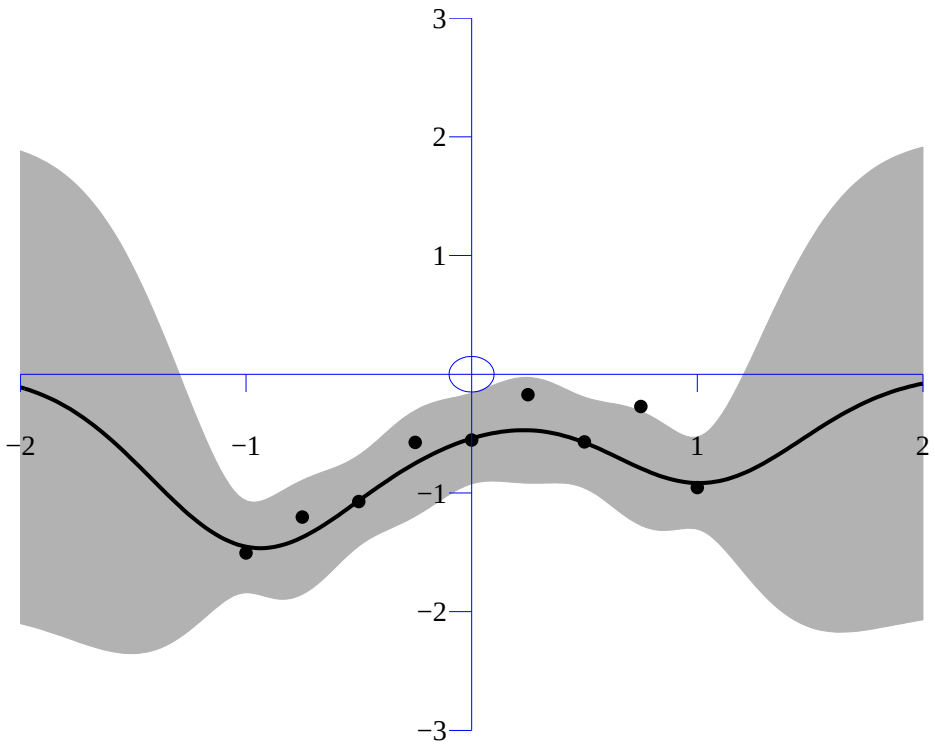


Figure: Examples include WiFi localization, C14 callibration curve.

Gaussian Process Regression

`demRegression`

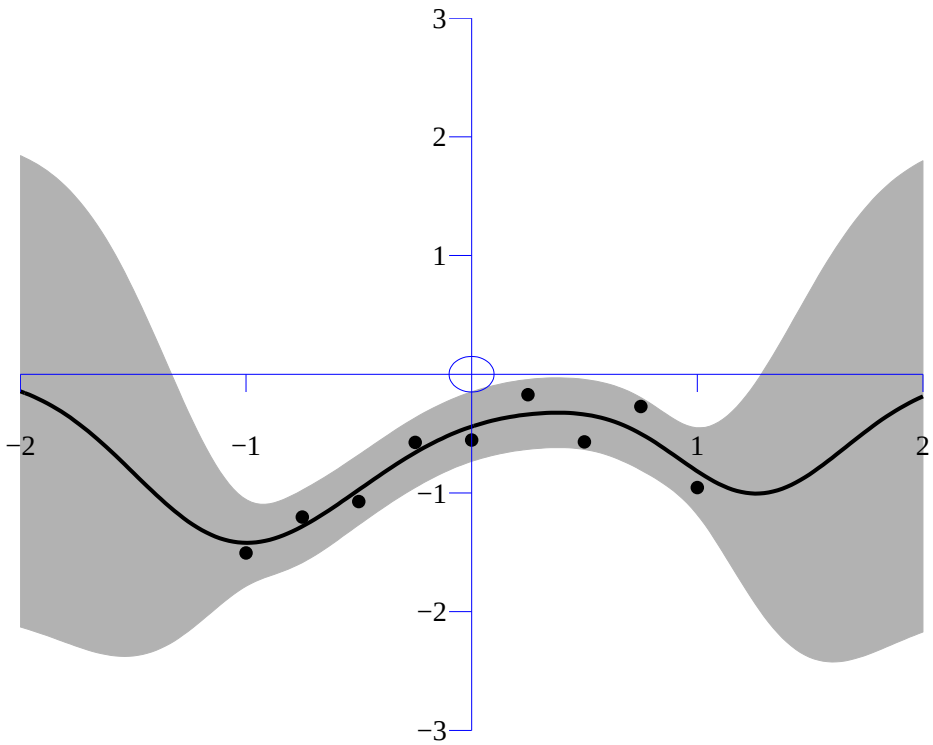


Figure: Examples include WiFi localization, C14 callibration curve.

Marginal likelihood

- The marginal log likelihood is

$$p(\mathbf{y}|\mathbf{X}) = \int p(\mathbf{y}|\mathbf{f},\mathbf{X})p(\mathbf{f}|\mathbf{X})d\mathbf{f} = \mathcal{N}\left(\mathbf{y};\mathbf{0},\mathbf{K}+\sigma^2\mathbf{I}\right)$$

- The negative log marginal likelihood is

$$-\log p(\mathbf{y}|\mathbf{X}) = \frac{1}{2}\mathbf{y}^T(\mathbf{K}+\sigma^2\mathbf{I})^{-1}\mathbf{y} + \frac{1}{2}\log|\mathbf{K}+\sigma^2\mathbf{I}| + \frac{n}{2}\log 2\pi$$

Marginal likelihood

- The marginal log likelihood is

$$p(\mathbf{y}|\mathbf{X}) = \int p(\mathbf{y}|\mathbf{f},\mathbf{X})p(\mathbf{f}|\mathbf{X})d\mathbf{f} = \mathcal{N}\left(\mathbf{y};\mathbf{0},\mathbf{K}+\sigma^2\mathbf{I}\right)$$

- The negative log marginal likelihood is

$$-\log p(\mathbf{y}|\mathbf{X}) = \frac{1}{2}\mathbf{y}^T(\mathbf{K}+\sigma^2\mathbf{I})^{-1}\mathbf{y} + \frac{1}{2}\log|\mathbf{K}+\sigma^2\mathbf{I}| + \frac{n}{2}\log 2\pi$$

Learning the GP

- Learning the GP means estimating the hyperparameters.
- We do not need to estimate the weights since we have marginalized them.
- The hyperparameters are typically estimated by maximizing the likelihood, or equivalently by minimizing the negative log likelihood, which ignoring constants is

$$\Theta = \underset{\Theta}{\operatorname{argmin}} \frac{1}{2}\mathbf{y}^T(\mathbf{K}+\sigma^2\mathbf{I})^{-1}\mathbf{y} + \frac{1}{2}\log|\mathbf{K}+\sigma^2\mathbf{I}|$$

Learning the GP

- Learning the GP means estimating the hyperparameters.
- We do not need to estimate the weights since we have marginalized them.
- The hyperparameters are typically estimated by maximizing the likelihood, or equivalently by minimizing the negative log likelihood, which ignoring constants is

$$\Theta = \underset{\Theta}{\operatorname{argmin}} \frac{1}{2}\mathbf{y}^T(\mathbf{K}+\sigma^2\mathbf{I})^{-1}\mathbf{y} + \frac{1}{2}\log|\mathbf{K}+\sigma^2\mathbf{I}|$$

Predictive distribution and representer theorem

- Using compact notation

$$\bar{\mathbf{f}}_* = \mathbf{k}_*^T (\mathbf{K} + \sigma^2 \mathbf{I})^{-1} \mathbf{y}$$

- The mean prediction is a linear combination of the observations $\mathbf{y}$.

Predictive distribution and representer theorem

- Using compact notation

$$\bar{\mathbf{f}}_* = \mathbf{k}_*^T (\mathbf{K} + \sigma^2 \mathbf{I})^{-1} \mathbf{y}$$

- The mean prediction is a linear combination of the observations $\mathbf{y}$.
- It is also a linear combination of n kernel functions, each center at a training point

$$\bar{\mathbf{f}}_* = \sum_{i=1}^{n} \alpha_i k(\mathbf{x}_i, \mathbf{x}_*)$$

where the $\alpha = (\mathbf{K} + \sigma^2 \mathbf{I})^{-1} \mathbf{y}$.

Predictive distribution and representer theorem

- Using compact notation

$$\bar{\mathbf{f}}_* = \mathbf{k}_*^T(\mathbf{K} + \sigma^2\mathbf{I})^{-1}\mathbf{y}$$

- The mean prediction is a linear combination of the observations $\mathbf{y}$.
- It is also a linear combination of n kernel functions, each center at a training point

$$\bar{\mathbf{f}}_* = \sum_{i=1}^{n} \alpha_i k(\mathbf{x}_i, \mathbf{x}_*)$$

  where the $\alpha = (\mathbf{K} + \sigma^2\mathbf{I})^{-1}\mathbf{y}$.
- This is the representer theorem!

Predictive distribution and representer theorem

- Using compact notation

$$\bar{\mathbf{f}}_* = \mathbf{k}_*^T (\mathbf{K} + \sigma^2 \mathbf{I})^{-1} \mathbf{y}$$

- The mean prediction is a linear combination of the observations $\mathbf{y}$.
- It is also a linear combination of n kernel functions, each center at a training point

$$\bar{\mathbf{f}}_* = \sum_{i=1}^{n} \alpha_i k(\mathbf{x}_i, \mathbf{x}_*)$$

  where the $\alpha = (\mathbf{K} + \sigma^2 \mathbf{I})^{-1} \mathbf{y}$.
- This is the representer theorem!
- What's the difference with SVMs?

Predictive distribution and representer theorem

- Using compact notation

$$\bar{\mathbf{f}}_* = \mathbf{k}_*^T (\mathbf{K} + \sigma^2 \mathbf{I})^{-1} \mathbf{y}$$

- The mean prediction is a linear combination of the observations $\mathbf{y}$.
- It is also a linear combination of n kernel functions, each center at a training point

$$\bar{\mathbf{f}}_* = \sum_{i=1}^{n} \alpha_i k(\mathbf{x}_i, \mathbf{x}_*)$$

  where the $\alpha = (\mathbf{K} + \sigma^2 \mathbf{I})^{-1} \mathbf{y}$.
- This is the representer theorem!
- What's the difference with SVMs?
- Answer: α has closed-form solution.

Predictive distribution and representer theorem

- Using compact notation

$$\bar{\mathbf{f}}_* = \mathbf{k}_*^T (\mathbf{K} + \sigma^2 \mathbf{I})^{-1} \mathbf{y}$$

- The mean prediction is a linear combination of the observations $\mathbf{y}$.
- It is also a linear combination of n kernel functions, each center at a training point

$$\bar{\mathbf{f}}_* = \sum_{i=1}^{n} \alpha_i k(\mathbf{x}_i, \mathbf{x}_*)$$

  where the $\alpha = (\mathbf{K} + \sigma^2 \mathbf{I})^{-1} \mathbf{y}$.
- This is the representer theorem!
- What's the difference with SVMs?
- Answer: α has closed-form solution.

Is regression a good approach for pose estimation?

- Regression cannot model mutimodal mappings
- Solution is to use a mixture of experts

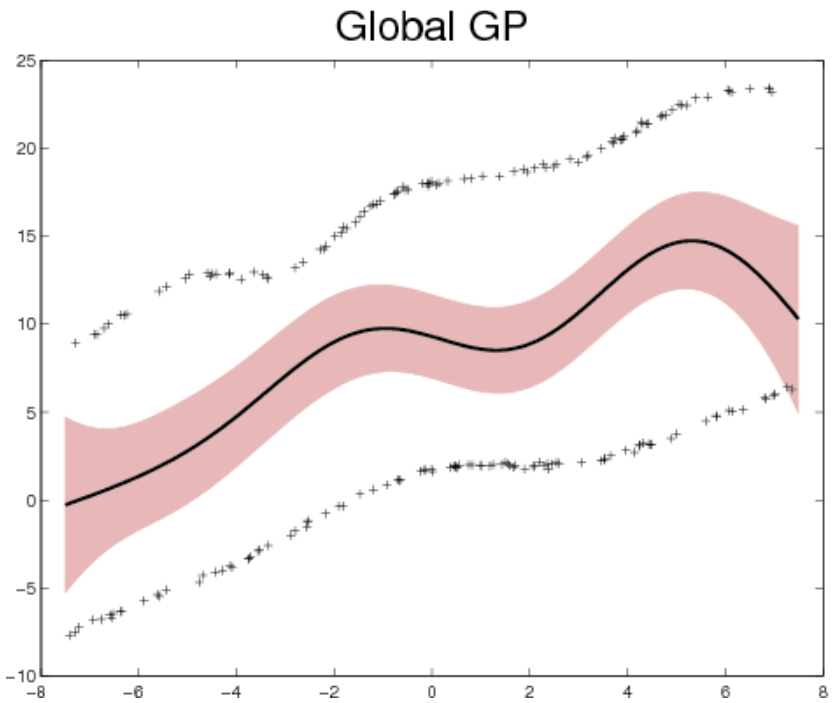


Figure: Regression is not a good model for pose estimation (Urtasun et al. 08)

Mixture of experts

- One solution to the multimodal problem is to use a mixture of experts where each expert focus on a modality.
- The problem is still there when there is a continium of solutions.
- In that case the best you can do is use NN.

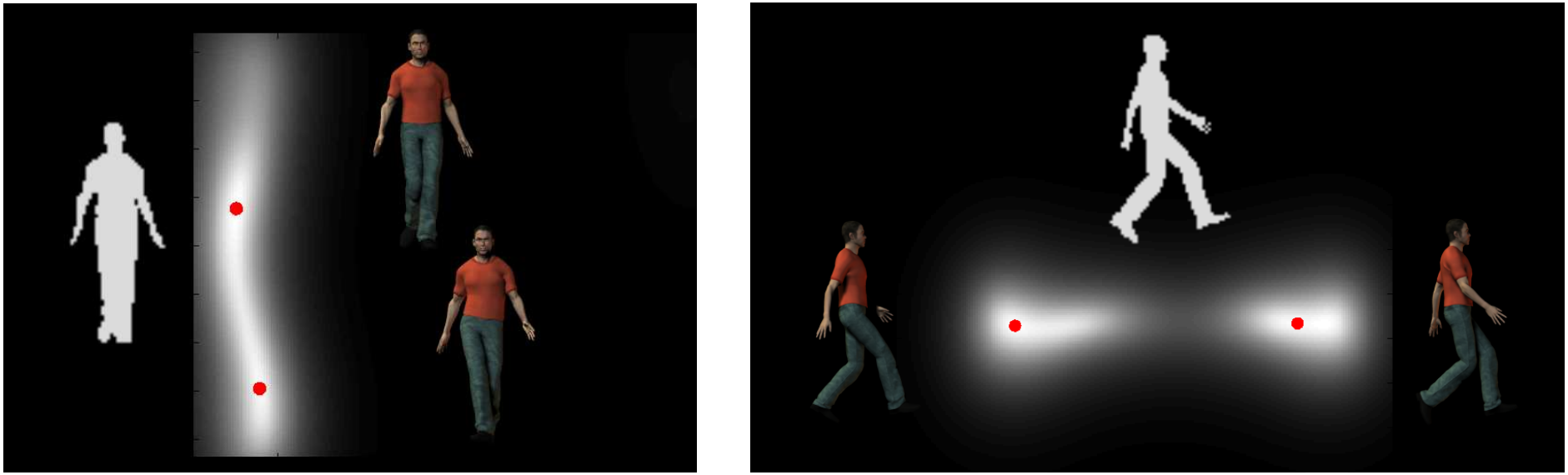


Figure: Illustration of ambiguities inherent to pose estimation from a single image (Ek 2009)

Simplest mixture: partition the space

- The simplest mixture model is to partition the space and learn a regressor for each cluster.
- Make sure that for each cluster all the examples are of the same mode
- Then learn a regressor independently for each cluster.
- The advantage is that it's fast, e.g., $\mathcal{O}(Tk^3)$ with k the number of points per cluster and T the number of clusters for kernel regressors.
- Problem of discontinuities in the boundaries of the clusters

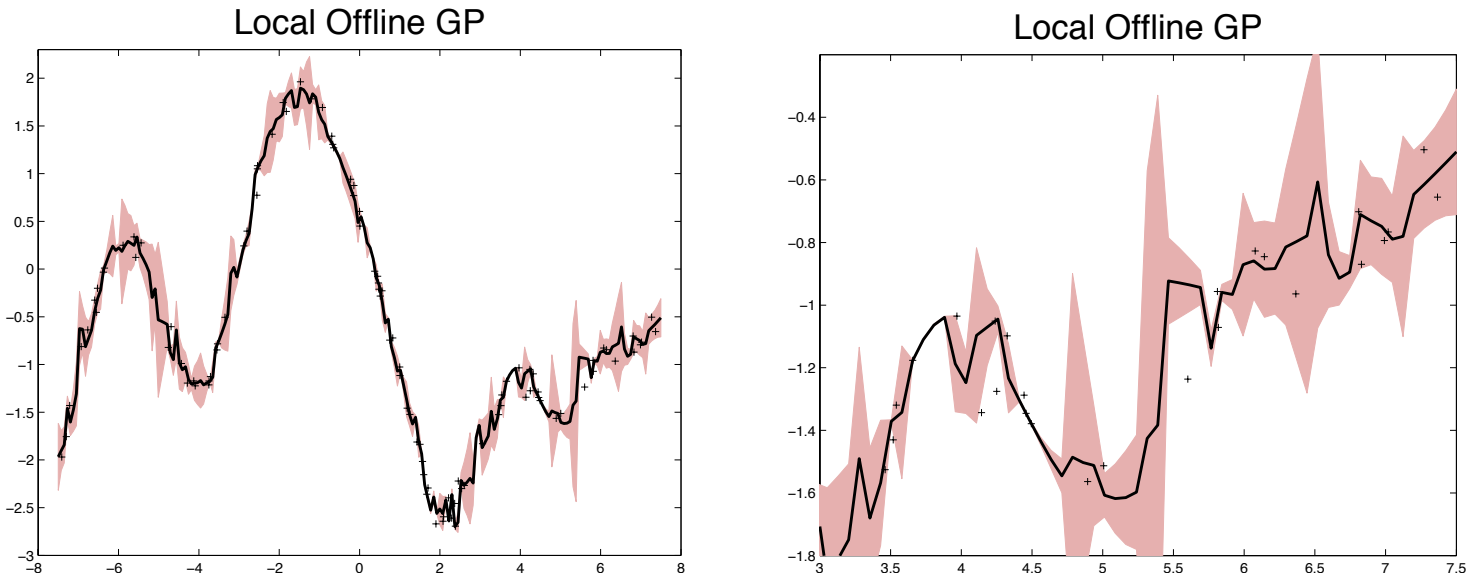


Figure: Discontinuities when using clustered experts (Urtasun et al. 08)

Online partitioning

- To avoid the discontinuities due to clustering, Urtasun et al. 08 proposed to center the predictor at each test point.
- The difficulty is that for each test point we don't know the pose.
- Solution: compute the NN of the new point, and split those into modes, and learn a regressor for each.

Online partitioning

- To avoid the discontinuities due to clustering, Urtasun et al. 08 proposed to center the predictor at each test point.
- The difficulty is that for each test point we don't know the pose.
- Solution: compute the NN of the new point, and split those into modes, and learn a regressor for each.
- Problem, as many regressors as training points.

Online partitioning

- To avoid the discontinuities due to clustering, Urtasun et al. 08 proposed to center the predictor at each test point.
- The difficulty is that for each test point we don't know the pose.
- Solution: compute the NN of the new point, and split those into modes, and learn a regressor for each.
- Problem, as many regressors as training points.
- Solution: Urtasun et al. 08 proposed to use GP since they are close form if no hyperparameter is estimated.
- Hyperparameters learned offline, and assume smoothness of the space.
- For each online regressor take the hyperparameters of the regressor that is located closer to it.
- One advantage with respect to global learning is that the regressors are adapted to the local characteristics of the data.

Online partitioning

- To avoid the discontinuities due to clustering, Urtasun et al. 08 proposed to center the predictor at each test point.
- The difficulty is that for each test point we don't know the pose.
- Solution: compute the NN of the new point, and split those into modes, and learn a regressor for each.
- Problem, as many regressors as training points.
- Solution: Urtasun et al. 08 proposed to use GP since they are close form if no hyperparameter is estimated.
- Hyperparameters learned offline, and assume smoothness of the space.
- For each online regressor take the hyperparameters of the regressor that is located closer to it.
- One advantage with respect to global learning is that the regressors are adapted to the local characteristics of the data.

Online vs Offline

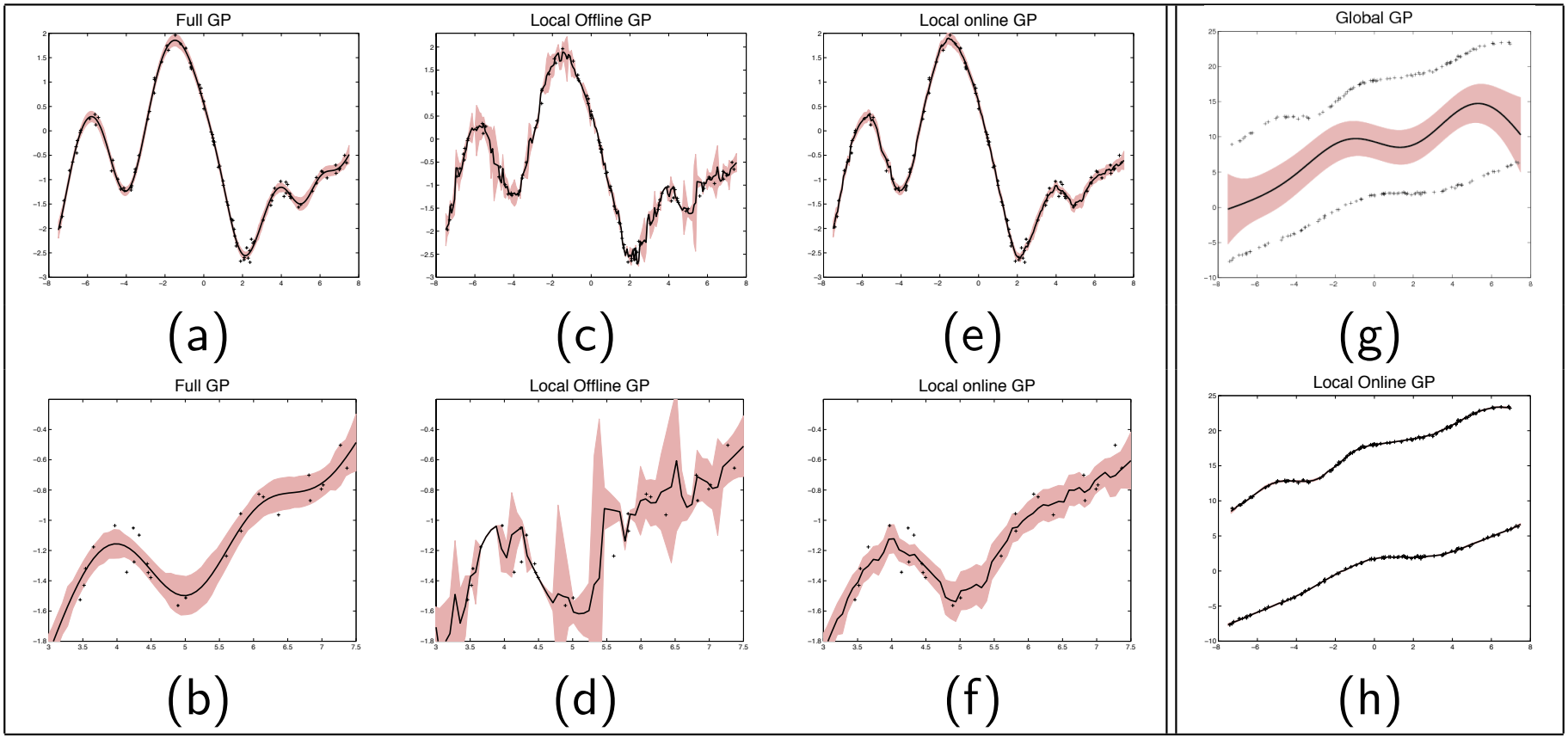


Figure: Advantages of online local regression (Urtasun et al. 08). (a,b) global GP, (c,d) Offline clustering, (e,f) online GP (g) Global GP with multimodal mappings (h) local GP with multimodal mappings.

Algorithm for learning and inference

OFFLINE: Learning hyperparameters

R: number of local GP to learn

for $n = 1 \dots R$ **do**

$i = rand(N)$

$\kappa = findNN(\mathbf{X}, \mathbf{x}_i, S)$

$\{\bar{\beta}^i\} \leftarrow \max \; p(\mathbf{X}_\kappa, \bar{\beta}^i | \mathbf{Y}_\kappa)$

$\mathbf{Y}_R = [\mathbf{Y}_R, \mathbf{y}_i]$

end for

ONLINE: Inference of test point $\mathbf{x}_*$

T: number of experts, S: size of each expert

$\eta = findNN(\mathbf{X}, \mathbf{x}_*, T)$

for $j = 1 \dots T$ **do**

$\zeta = findNN(\mathbf{Y}, \mathbf{y}_{\eta_j}, S)$

$t = findNN(\mathbf{Y}_R, \mathbf{y}_{\eta_j}, 1)$

$\bar{\beta} = \bar{\beta}^t$

$\mu_j = K_{*,\zeta}(\mathbf{K}_{\zeta,\zeta} + \sigma^2_{noise}\mathbf{I})^{-1}\mathbf{Y}_\zeta$

$\sigma_j = k_{*,*} - K_{*,\zeta}(\mathbf{K}_{\zeta,\zeta} + \sigma^2_{noise}\mathbf{I})^{-1}K_{\zeta,*}$

end for

$p(\mathbf{f}_*|\mathbf{y}) \approx \sum_{i=1}^{T} \pi_i \mathcal{N}(\mu_i, \sigma_i^2)$

Results with local online GPs

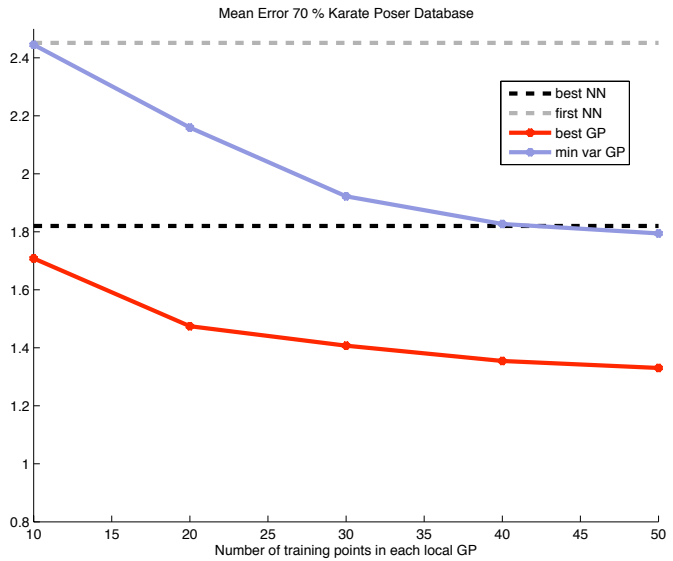
	walk	jog	box	mono.	discrim.	dyn.
Lee et al. I	3.4	–	–	yes	no	no
Lee et al. II	**3.1**	–	–	yes	no	yes
Pope	4.53	4.38	9.43	yes	yes	no
Muendermann et al.	5.31	–	4.54	no	no	yes
Li et al.	–	–	20.0	yes	no	yes
Brubaker et al.	10.4	–	–	yes	no	yes
Our approach	3.27	**3.12**	**3.85**	yes	yes	no

Table: Comparison with state of the art (error in cm).

Learning from very large training sets

- When used with fast NN techniques, it can be trained from millions of examples and represent any possible pose.

DB size	1-NN	Best of-10-NN	GP ($S = 10$)	GP ($S = 20$)	GP ($S = 30$)	GP ($S = 40$)
1,500	0.88 $\pm$ 1.77	0.71 $\pm$ 1.38	0.83 $\pm$ 1.53	0.98 $\pm$ 1.70	**0.56 $\pm$ 1.40**	0.70 $\pm$ 1.45
15,000	1.92 $\pm$ 2.76	1.49 $\pm$ 1.81	1.32 $\pm$ 2.07	1.10 $\pm$ 1.88	1.03 $\pm$ 1.81	**0.99 $\pm$ 1.77**
50,000	1.83 $\pm$ 2.62	1.34 $\pm$ 1.47	1.10 $\pm$ 1.85	0.91 $\pm$ 1.64	0.90 $\pm$ 1.66	**0.87 $\pm$ 1.58**



More results

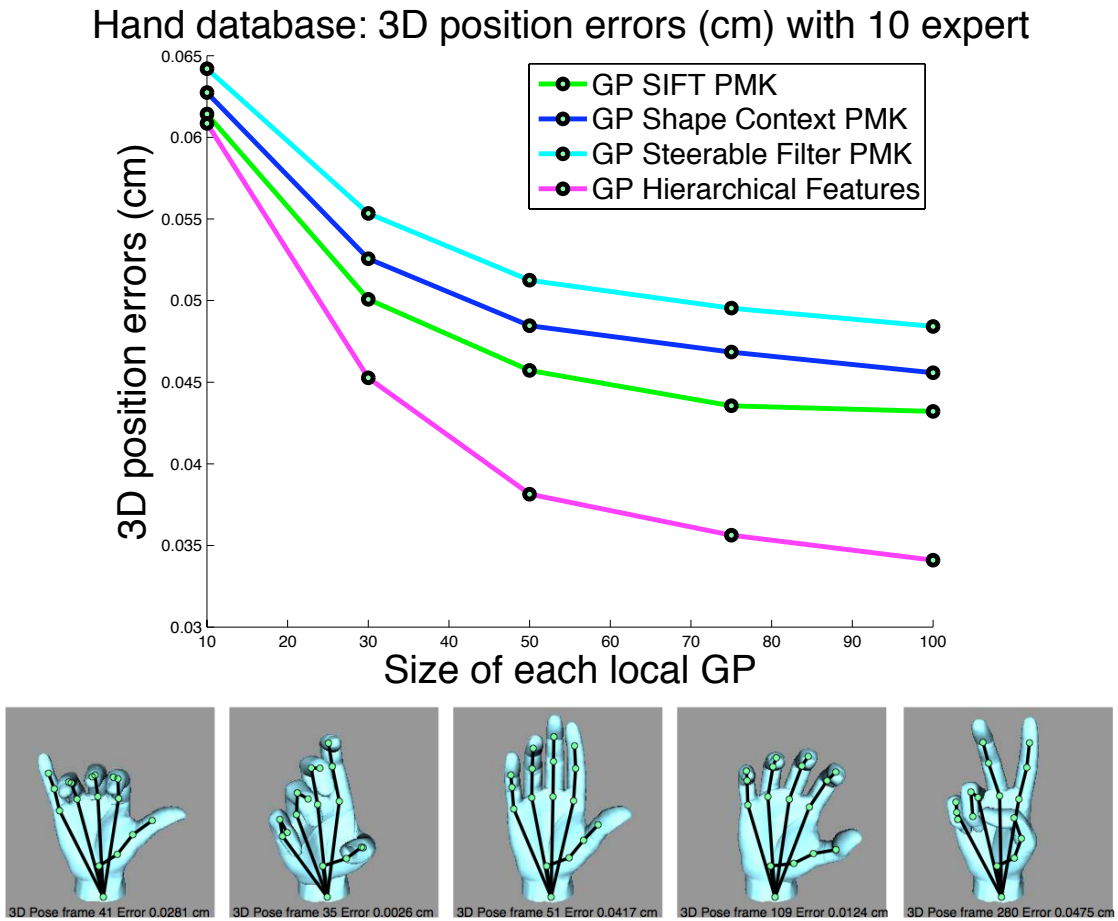


Figure: Hand pose from multiple features

More?

- Discriminative techniques are great since they do not require initialization.
- Difficult to model the multimodal mappings.
- If you want to learn more, look at the additional material.
- Otherwise, do the research project on this topic!
- Next week we will look more into discriminative prediction