CSC 412/2506:
Probabilistic Learning and Reasoning
Week 11: Bayesian Regression & Kernel Methods

Murat A. Erdogdu

University of Toronto

Overview of the first hour

- Continuing in our theme of probabilistic models for continuous variables.
- We give a probabilistic interpretation of linear regression.
- Chapter 3.3 in Bishop's book.

Completing the Square for Gaussians

Useful technique to find moments of Gaussian random variables.

- It is a multivariate generalization of completing the square.
- The density of $\mathbf{x} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ satifies:

$$\begin{aligned}\log p(\mathbf{x}) &= -\tfrac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu}) + \text{const} \\ &= -\tfrac{1}{2}\mathbf{x}^\top \boldsymbol{\Sigma}^{-1}\mathbf{x} + \mathbf{x}^\top \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu} + \text{const}\end{aligned}$$

- Thus, if we know $\mathbf{w}$ is Gaussian with *unknown* mean and covariance, and we also know that

$$\log p(\mathbf{w}) = -\tfrac{1}{2}\mathbf{w}^\top \mathbf{A}\mathbf{w} + \mathbf{w}^\top \mathbf{b} + \text{const}$$

for $\mathbf{A}$ positive definite, then we know that

$$\mathbf{w} \sim \mathcal{N}(\mathbf{A}^{-1}\mathbf{b}, \mathbf{A}^{-1}).$$

Bayesian Linear Regression

- We take the Bayesian approach to linear regression.
  - This is in contrast with the standard regression.
  - By inferring a posterior distribution over the *parameters*, the model can know what it doesn't know.
- How can uncertainty in the predictions help us?
  - Smooth out the predictions by averaging over lots of plausible explanations
  - Assign confidences to predictions
  - Make more robust decisions

Recap: Linear Regression

- Given a training set of inputs and targets $\{(\mathbf{x}^{(i)}, y^{(i)})\}_{i=1}^{N}$
- Linear model:

$$y = \mathbf{w}^\top \boldsymbol{\psi}(\mathbf{x}) + \epsilon$$

- Vectorized, we have the design matrix $\mathbf{X}$ in input space and

$$\boldsymbol{\Psi} = \begin{bmatrix} - & \boldsymbol{\psi}(\mathbf{x}^{(1)}) & - \\ - & \boldsymbol{\psi}(\mathbf{x}^{(2)}) & - \\ & \vdots & \\ - & \boldsymbol{\psi}(\mathbf{x}^{(N)}) & - \end{bmatrix}, \qquad \mathbf{y} = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(N)} \end{bmatrix}$$

and predictions

$$\hat{\mathbf{y}} = \boldsymbol{\Psi}\mathbf{w}$$

Recap: Ridge Regression from 311

- No statistical model.
- Penalized sum of squares (ridge regression):

$$\text{minimize} \quad \frac{1}{2}\|\mathbf{y} - \mathbf{\Psi}\mathbf{w}\|^2 + \frac{\lambda}{2}\|\mathbf{w}\|^2$$

- The gradient: $(\mathbf{\Psi}^\top\mathbf{\Psi} + \lambda\mathbf{I})\mathbf{w} - \mathbf{\Psi}^\top\mathbf{y}$.
- Solution 1: solve analytically by setting the gradient to 0

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

- Solution 2: solve approximately using gradient descent

$$\mathbf{w} \leftarrow (1 - \alpha\lambda)\mathbf{w} - \alpha\mathbf{\Psi}^\top(\mathbf{\Psi}\mathbf{w} - \mathbf{y})$$

Linear Regression as Maximum Likelihood

- We can give linear regression a probabilistic interpretation by assuming a Gaussian noise model:

$$y \,|\, \mathbf{x} \sim \mathcal{N}(\mathbf{w}^\top \boldsymbol{\psi}(\mathbf{x}),\ \sigma^2)$$

- Linear regression is just maximum log-likelihood under this model:

$$\begin{aligned}
\sum_{i=1}^{N} \log p(y^{(i)} \,|\, \mathbf{x}^{(i)}; \mathbf{w}, b) &= \sum_{i=1}^{N} \log \mathcal{N}(y^{(i)}; \mathbf{w}^\top \boldsymbol{\psi}(\mathbf{x}^{(i)}), \sigma^2) \\
&= \sum_{i=1}^{N} \log \left[\frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(y^{(i)} - \mathbf{w}^\top \boldsymbol{\psi}(\mathbf{x}^{(i)}))^2}{2\sigma^2} \right) \right] \\
&= \text{const} - \frac{1}{2\sigma^2} \sum_{i=1}^{N} (y^{(i)} - \mathbf{w}^\top \boldsymbol{\psi}(\mathbf{x}^{(i)}))^2 \\
&= \text{const} - \frac{1}{2\sigma^2} \|\mathbf{y} - \boldsymbol{\Psi}\mathbf{w}\|^2
\end{aligned}$$

Regularized Linear Regression as MAP Estimation

- View an L_2 regularizer as MAP inference with a Gaussian prior.

$$\arg\max_{\mathbf{w}} \log p(\mathbf{w} \mid \mathcal{D}) = \arg\max_{\mathbf{w}} \left[\log p(\mathbf{w}) + \log p(\mathcal{D} \mid \mathbf{w})\right]$$

- We just derived the likelihood term $\log p(\mathcal{D} \mid \mathbf{w})$:

$$\log p(\mathcal{D} \mid \mathbf{w}) = \text{const} - \frac{1}{2\sigma^2} \|\mathbf{y} - \mathbf{\Psi}\mathbf{w}\|^2$$

- Assume a Gaussian prior, $\mathbf{w} \sim \mathcal{N}(\mathbf{m}, \mathbf{S})$:

$$\begin{aligned}\log p(\mathbf{w}) &= \log \left[\frac{1}{(2\pi)^{D/2}|\mathbf{S}|^{1/2}} \exp\left(-\tfrac{1}{2}(\mathbf{w}-\mathbf{m})^\top \mathbf{S}^{-1}(\mathbf{w}-\mathbf{m})\right)\right] \\ &= -\tfrac{1}{2}(\mathbf{w}-\mathbf{m})^\top \mathbf{S}^{-1}(\mathbf{w}-\mathbf{m}) + \text{const}\end{aligned}$$

- Commonly, $\mathbf{m} = \mathbf{0}$ and $\mathbf{S} = \eta\mathbf{I}$, so

$$\log p(\mathbf{w}) = -\frac{1}{2\eta}\|\mathbf{w}\|^2 + \text{const.}$$

This is just L_2 regularization!

Full Bayesian Inference

- Full Bayesian inference makes predictions by averaging over all likely explanations under the posterior distribution.
- Compute posterior using Bayes' Rule:

$$p(\mathbf{w} \mid \mathcal{D}) \propto p(\mathbf{w}) p(\mathcal{D} \mid \mathbf{w})$$

- Make predictions using the posterior predictive distribution:

$$p(y \mid \mathbf{x}, \mathcal{D}) = \int p(\mathbf{w} \mid \mathcal{D})\, p(y \mid \mathbf{x}, \mathbf{w})\, \mathrm{d}\mathbf{w}$$

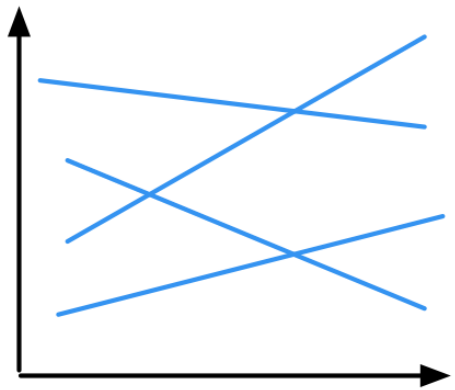
- Doing this lets us quantify our uncertainty.

Bayesian Linear Regression

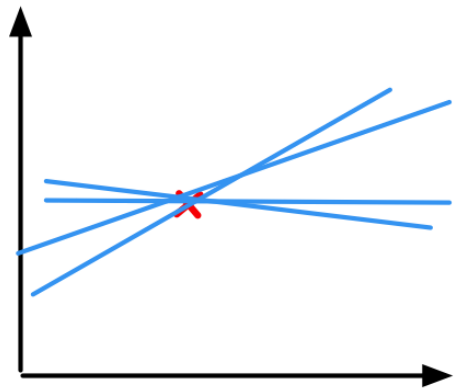
- **Prior distribution:** $\mathbf{w} \sim \mathcal{N}(\mathbf{0}, \mathbf{S})$
- **Likelihood:** $y \,|\, \mathbf{x}, \mathbf{w} \sim \mathcal{N}(\mathbf{w}^\top \boldsymbol{\psi}(\mathbf{x}),\ \sigma^2)$
- Assuming fixed/known $\mathbf{S}$ and σ^2 is a big assumption. More on this later.

Bayesian Linear Regression

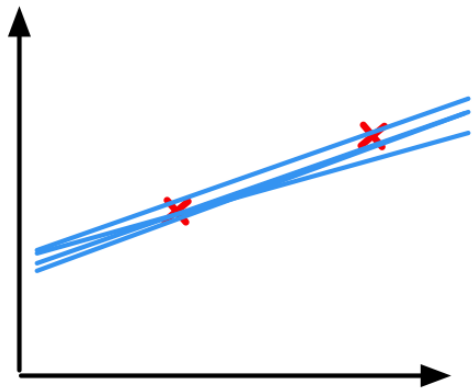
- Bayesian linear regression considers various plausible explanations for how the data were generated.
- It makes predictions using all possible regression weights, weighted by their posterior probability.
- Here are samples from the prior $p(\mathbf{w})$ and posteriors $p(\mathbf{w} \mid \mathcal{D})$



no observations



one observation



two observations

Bayesian Linear Regression: Posterior

- **Deriving the posterior distribution:**

$$\begin{aligned}
\log p(\mathbf{w} \,|\, \mathcal{D}) &= \log p(\mathbf{w}) + \log p(\mathcal{D} \,|\, \mathbf{w}) + \text{const} \\
&= -\tfrac{1}{2}\mathbf{w}^\top \mathbf{S}^{-1}\mathbf{w} - \frac{1}{2\sigma^2}\|\boldsymbol{\Psi}\mathbf{w} - \mathbf{y}\|^2 + \text{const} \\
&= -\tfrac{1}{2}\mathbf{w}^\top \mathbf{S}^{-1}\mathbf{w} - \frac{1}{2\sigma^2}\left(\mathbf{w}^\top \boldsymbol{\Psi}^\top \boldsymbol{\Psi}\mathbf{w} - 2\mathbf{y}^\top \boldsymbol{\Psi}\mathbf{w} + \mathbf{y}^\top \mathbf{y}\right) + \text{const} \\
&= -\tfrac{1}{2}\mathbf{w}^\top \left(\sigma^{-2}\boldsymbol{\Psi}^\top \boldsymbol{\Psi} + \mathbf{S}^{-1}\right)\mathbf{w} + \frac{1}{\sigma^2}\mathbf{y}^\top \boldsymbol{\Psi}\mathbf{w} + \text{const} \text{ (complete the } \square\text{!)}
\end{aligned}$$

Thus $\mathbf{w} \,|\, \mathcal{D} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ where

$$\boldsymbol{\mu} = \left(\boldsymbol{\Psi}^\top \boldsymbol{\Psi} + \sigma^2 \mathbf{S}^{-1}\right)^{-1} \boldsymbol{\Psi}^\top \mathbf{y}$$

$$\boldsymbol{\Sigma} = \sigma^2 \left(\boldsymbol{\Psi}^\top \boldsymbol{\Psi} + \sigma^2 \mathbf{S}^{-1}\right)^{-1}$$

Bayesian Linear Regression: Posterior

- Gaussian prior leads to a Gaussian posterior, and so the Gaussian distribution is the conjugate prior for linear regression model.
- Compare $\boldsymbol{\mu}$ to the closed-form solution for linear regression:

$$\mathbf{w} = (\boldsymbol{\Psi}^\top \boldsymbol{\Psi} + \lambda \mathbf{I})^{-1} \boldsymbol{\Psi}^\top \mathbf{y}$$

  This is the mean of the posterior for $\mathbf{S} = \frac{\sigma^2}{\lambda}\mathbf{I}$.

- As $\lambda \to 0$, the standard deviation of the prior goes to ∞, and the mean of the posterior converges to the MLE.

Bayesian Linear Regression

Illustration of sequential Bayesian learning for $y = w_0 + w_1 x$, $w_0 = -0.3$, $w_1 = 0.5$.

Left column:

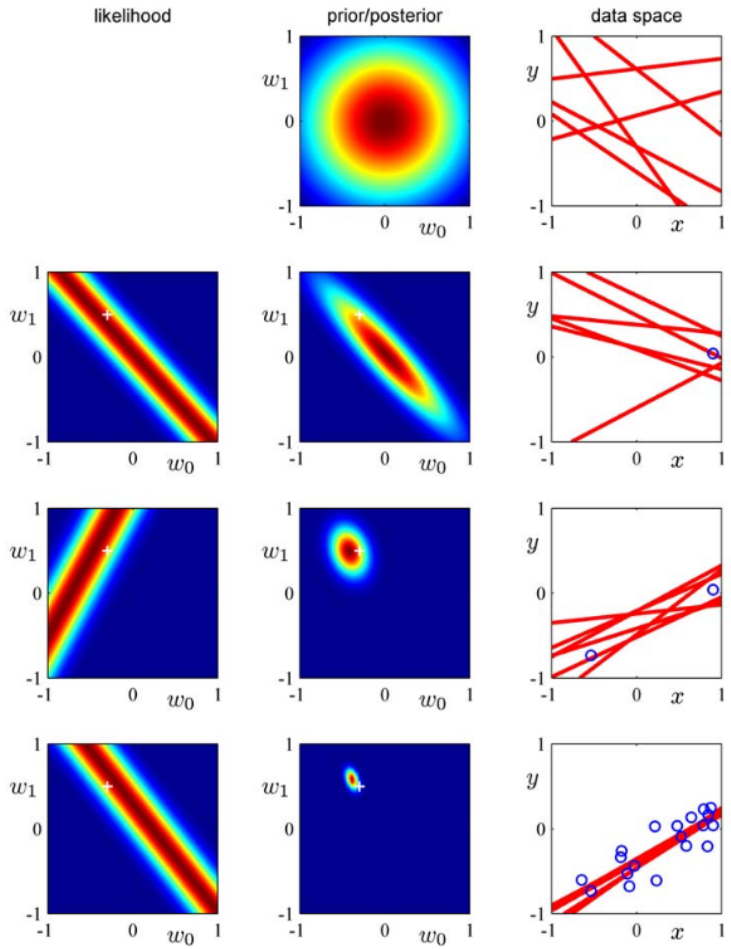
- Likelihood of a single data point.
- Single point does not identify a line.
- Fix (x, y) then $w_0 = y - w_1 x$.

Middle column:

- Prior/posterior.

Right column:

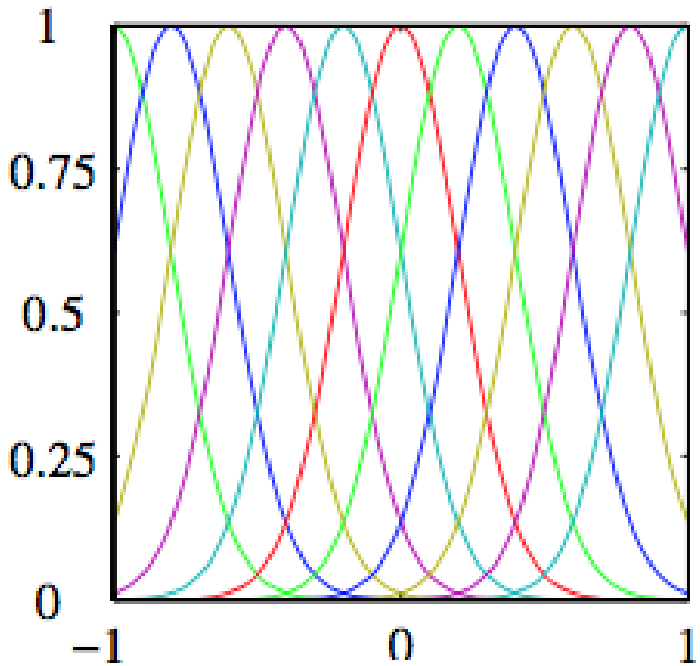
- Lines: samples from the posterior.
- Dots: data points.



Radial bases example

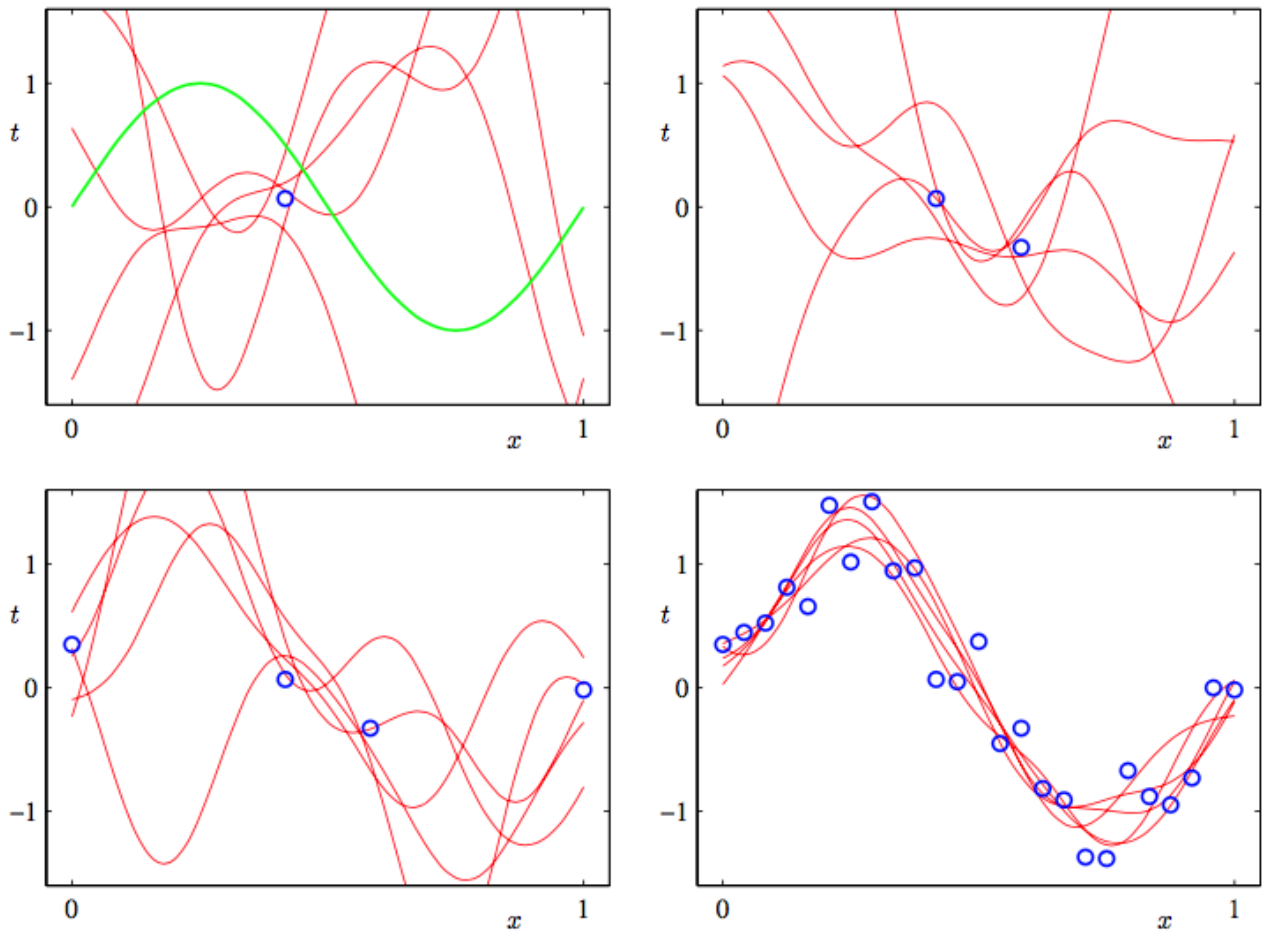
- Example with radial basis function (RBF) features

$$\psi_j(x) = \exp\left(-\frac{(x-\mu_j)^2}{2s^2}\right)$$



Radial bases example

Functions sampled from the posterior:



Posterior predictive distribution

- The posterior just gives us distribution over the parameter space, but if we want to make predictions, the natural choice is to use the posterior predictive distribution.
- Posterior predictive distribution:

$$p(y \,|\, \mathbf{x}, \mathcal{D}) = \int \underbrace{p(y \,|\, \mathbf{x}, \mathbf{w})}_{\mathcal{N}(y\,;\, \mathbf{w}^\top \boldsymbol{\psi}(\mathbf{x}), \sigma)} \underbrace{p(\mathbf{w} \,|\, \mathcal{D})}_{\mathcal{N}(\mathbf{w}\,;\, \boldsymbol{\mu}, \boldsymbol{\Sigma})} \, \mathrm{d}\mathbf{w}$$

- Another interpretation: $y = \mathbf{w}^\top \boldsymbol{\psi}(\mathbf{x}) + \varepsilon$, where $\varepsilon \sim \mathcal{N}(0, \sigma)$ is independent of $\mathbf{w} \,|\, \mathcal{D} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$.

Recall

$$\boldsymbol{\mu} = \left(\boldsymbol{\Psi}^\top \boldsymbol{\Psi} + \sigma^2 \mathbf{S}^{-1}\right)^{-1} \boldsymbol{\Psi}^\top \mathbf{y}$$

$$\boldsymbol{\Sigma} = \sigma^2 \left(\boldsymbol{\Psi}^\top \boldsymbol{\Psi} + \sigma^2 \mathbf{S}^{-1}\right)^{-1}$$

Bayesian Linear Regression

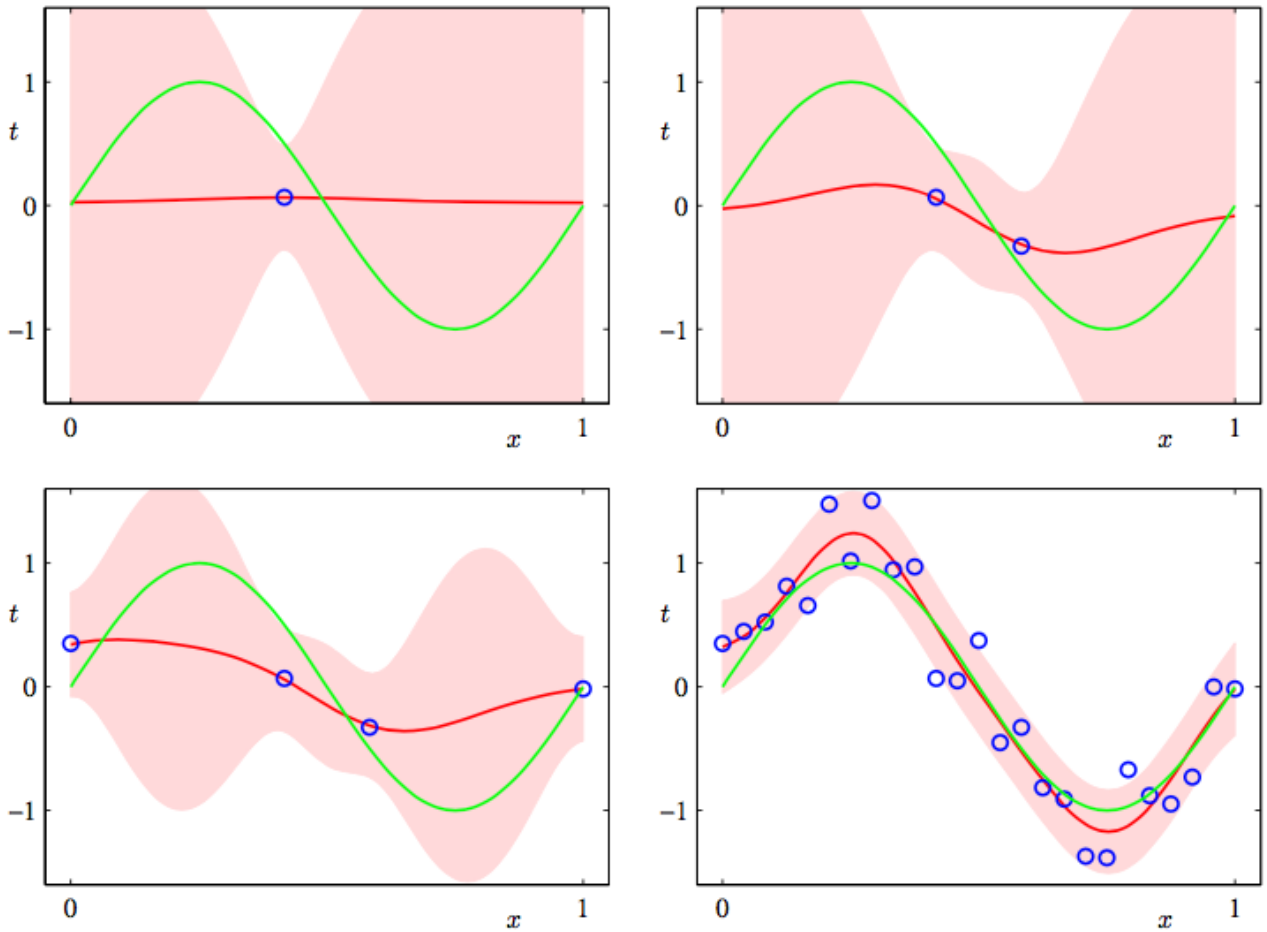
- Another interpretation: $y = \mathbf{w}^\top \boldsymbol{\psi}(\mathbf{x}) + \varepsilon$, where $\varepsilon \sim \mathcal{N}(0, \sigma)$ is independent of $\mathbf{w} \,|\, \mathcal{D} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$.
- Again by the fact that affine transformations of Gaussian vectors are Gaussian, y is a Gaussian distribution with parameters

$$\mu_{\text{pred}} = \boldsymbol{\mu}^\top \boldsymbol{\psi}(\mathbf{x})$$
$$\sigma^2_{\text{pred}} = \boldsymbol{\psi}(\mathbf{x})^\top \boldsymbol{\Sigma} \boldsymbol{\psi}(\mathbf{x}) + \sigma^2$$

- Hence, the posterior predictive distribution is $\mathcal{N}(y \,|\, \mu_{\text{pred}}, \sigma^2_{\text{pred}})$.

Bayesian Linear Regression

Here we visualize confidence intervals based on the posterior predictive mean and variance at each point:



Summary

- This lecture covered the basics of Bayesian regression.

Key points:

- Posterior can be computed by completing the square.
- Posterior predictive distribution.
- Uncertainty quantification.

Linear Regression as Maximum Likelihood

- We gave linear regression a probabilistic interpretation by assuming a Gaussian noise model:

$$y \,|\, \mathbf{x} \sim \mathcal{N}(\mathbf{w}^\top \boldsymbol{\psi}(\mathbf{x}),\ \sigma^2)$$

- The MLE under the first model leads to ordinary least squares.
- We can also do full Bayesian inference as explained last hour.
  - Recall MAP estimator with a special Gaussian prior becomes equivalent to the ridge regression estimator.

Some problems with this formulation

- The MLE will not be uniquely defined if $N < M$.
  - We can use ridge regression or other regularization.
- Flexibility may require a large number M of features, which may need to depend on N.
- We would like to have a method that is more automatic.
- Kernel regression offers such a flexible framework.

Kernel methods are applicable widely beyond regression problems.

- We cover classification later in the context of Gaussian Processes.

Regularized Linear Regression: towards kernel trick

- In the ridge regression problem we minimize

$$E(\mathbf{w}) = \frac{1}{2}\|\mathbf{y} - \mathbf{\Psi}\mathbf{w}\|^2 + \frac{\lambda}{2}\mathbf{w}^\top\mathbf{w}$$

$$\nabla E(\mathbf{w}) \;=\; \mathbf{\Psi}^\top\mathbf{\Psi}\mathbf{w} - \mathbf{\Psi}^\top\mathbf{y} + \lambda\mathbf{w}.$$

- Taking $\nabla E(\mathbf{w}) = 0$ is equivalent to solving:

$$\mathbf{w} = \frac{1}{\lambda}\mathbf{\Psi}^\top(\mathbf{y} - \mathbf{\Psi}\mathbf{w}) = \mathbf{\Psi}^\top\mathbf{a} \;\in\; \mathbb{R}^M,$$

  where $\boldsymbol{a} = (\mathbf{y} - \mathbf{\Psi}\mathbf{w})/\lambda \in \mathbb{R}^N$.

- Substitute $\mathbf{w} = \mathbf{\Psi}^\top\mathbf{a}$ back in $E(\mathbf{w})$, we get

$$E(\mathbf{a}) = \frac{1}{2}\|\mathbf{y} - \mathbf{\Psi}\mathbf{\Psi}^\top\mathbf{a}\|^2 + \frac{\lambda}{2}\mathbf{a}^\top\mathbf{\Psi}\mathbf{\Psi}^\top\mathbf{a}$$

Kernel Ridge Regression

- Introduce the gram matrix $\boldsymbol{K} = \boldsymbol{\Psi}\boldsymbol{\Psi}^\top$, i.e.

$$K_{ij} = \boldsymbol{\psi}(\mathbf{x}^{(i)})^\top \boldsymbol{\psi}(\mathbf{x}^{(j)}) =: k(\mathbf{x}^{(i)}, \mathbf{x}^{(j)})$$

  which we call the **kernel matrix**. Function $k(\mathbf{x}, \mathbf{x}')$ is the **kernel**.
- Therefore, we minimize

$$E(\mathbf{a}) = \frac{1}{2}\|\mathbf{y} - \boldsymbol{K}\mathbf{a}\|^2 + \frac{\lambda}{2}\mathbf{a}^\top \boldsymbol{K}\mathbf{a}$$

- Plugging $\mathbf{w} = \boldsymbol{\Psi}^\top \mathbf{a}$ to $\mathbf{a} = (\mathbf{y} - \boldsymbol{\Psi}\mathbf{w})/\lambda$ we get

$$\mathbf{a} = (\boldsymbol{K} + \lambda \mathbf{I}_N)^{-1}\mathbf{y}.$$

- Substitute back into the linear regression model

$$\hat{y}(\mathbf{x}) = \boldsymbol{\psi}(\mathbf{x})^\top \mathbf{w} = \boldsymbol{\psi}(\mathbf{x})^\top \boldsymbol{\Psi}^\top \mathbf{a} = \mathbf{k}(\mathbf{x})^\top (\boldsymbol{K} + \lambda \mathbf{I}_N)^{-1}\mathbf{y}$$

  where $\mathbf{k}(\mathbf{x}) = \boldsymbol{\Psi}\boldsymbol{\psi}(\mathbf{x}) = [\boldsymbol{\psi}(\mathbf{x}^{(i)})^\top \boldsymbol{\psi}(\mathbf{x})]_i = [k(\mathbf{x}^{(i)}, \mathbf{x})]_i$.

Kernel Ridge Regression

- This is known as a dual formulation, aka Kernel trick.
- We have

$$\hat{y}(\mathbf{x}) = \mathbf{k}(\mathbf{x})^\top (\boldsymbol{K} + \lambda \mathbf{I}_N)^{-1} \mathbf{y},$$

  where $[\mathbf{k}(\mathbf{x})]_i = k(\mathbf{x}^{(i)}, \mathbf{x})$, $\mathbf{K}_{ij} = k(\mathbf{x}^{(i)}, \mathbf{x}^{(j)})$.
- The prediction at $\mathbf{x}$ is given by a linear combination $\mathbf{y}$.
- The coefficients depend on "proximity" of $\mathbf{x}$ to $\mathbf{x}^{(i)}$.
- Dual formulation requires inverting an $N \times N$ matrix, whereas the standard one requires inverting an $M \times M$ matrix.
- The advantage of the dual formulation is that it is expressed entirely in terms of the kernel function with no explicit reference to the feature map $\boldsymbol{\psi}(\mathbf{x})$ (can use features of high dimension).

Kernels: Formal definition

Positive semidefinite matrix (PSD)

A symmetric matrix $\boldsymbol{A} \in \mathbb{R}^{N \times N}$ is PSD if for every vector $\mathbf{u} \in \mathbb{R}^N$

$$\mathbf{u}^\top \boldsymbol{A} \mathbf{u} \geq 0.$$

- We can use feature maps $\boldsymbol{\psi} : \mathbb{R}^D \to \mathbb{R}^M$ to define kernels:

$$k(\mathbf{x}, \mathbf{x}') = \boldsymbol{\psi}(\mathbf{x})^\top \boldsymbol{\psi}(\mathbf{x}').$$

  But we can consider a (slightly) more general definition.

- A **kernel** $k(\mathbf{x}, \mathbf{x}')$ is any function such that for any N data points $\mathbf{x}^{(i)}$ for $i = 1, ..., N$, the kernel matrix $\boldsymbol{K}$ with entries $K_{ij} = k(\mathbf{x}^{(i)}, \mathbf{x}^{(j)})$ is positive semidefinite (Schoenberg 1938).

Feature map defines a kernel

- Let $k(\mathbf{x}, \mathbf{x}') = \boldsymbol{\psi}(\mathbf{x})^\top \boldsymbol{\psi}(\mathbf{x}')$
- The kernel matrix is given as $K_{ij} = k(\mathbf{x}^{(i)}, \mathbf{x}^{(j)})$, $\boldsymbol{K} = \boldsymbol{\Psi}\boldsymbol{\Psi}^\top$.
- We show that this matrix is positive semi-definite, $\forall \mathbf{u} \in \mathbb{R}^N$,

$$\mathbf{u}^\top \boldsymbol{K} \mathbf{u} = \mathbf{u}^\top \boldsymbol{\Psi}\boldsymbol{\Psi}^\top \mathbf{u} = (\boldsymbol{\Psi}^\top \mathbf{u})^\top \boldsymbol{\Psi}^\top \mathbf{u} = \|\boldsymbol{\Psi}^\top \mathbf{u}\|^2 \geq 0.$$

Main points:

- Forget the feature map.
- We can directly choose a kernel and work with it!
- The dimension of the feature space does not matter anymore.
- Kernels provide a measure of proximity between $\mathbf{x}$ and $\mathbf{x}'$.

Kernels: Examples

Example 1:

- D-dimensional inputs: $\mathbf{x} = (x_1, x_2, ..., x_D)^\top$ and $\mathbf{z} = (z_1, z_2, ...z_D)^\top$

$$\begin{aligned} k(\mathbf{x}, \mathbf{z}) =& (\mathbf{x}^\top \mathbf{z})^2 = (x_1 z_1 + x_2 z_2 + ...)^2 \\ =& x_1^2 z_1^2 + 2 x_1 z_1 x_2 z_2 + x_2^2 z_2^2 + ... \\ =& (x_1^2, x_2^2, ..., \sqrt{2} x_1 x_2, ...)^\top (z_1^2, z_2^2, ..., \sqrt{2} z_1 z_2, ...) \\ =& \boldsymbol{\psi}(\mathbf{x})^\top \boldsymbol{\psi}(\mathbf{z}) \end{aligned}$$

Example 2 (Gaussian kernel): $k(\mathbf{x}, \mathbf{z}) = \exp(-\|\mathbf{x} - \mathbf{z}\|^2 / 2\sigma^2)$.

- The feature vector has infinite dimension here!

Kernels: Example

- Predictions in the kernel ridge regression:

$$y(\mathbf{x}) = \mathbf{w}^T \boldsymbol{\psi}(\mathbf{x}) = \mathbf{a}^T \boldsymbol{\Psi} \boldsymbol{\psi}(\mathbf{x}) = \mathbf{k}(\mathbf{x})^T (\boldsymbol{K} + \lambda I)^{-1} \mathbf{y}$$

- Lets look at the predictions for the scaled targets $\mathbf{a} = (\boldsymbol{K} + \lambda I)^{-1} \mathbf{y}$

$$y(\mathbf{x}) = \mathbf{k}(\mathbf{x})^T \mathbf{a} = \sum_{i=1}^{N} k(\mathbf{x}, \mathbf{x}^{(i)}) \; a_i$$

- Which looks very much like k-NN!

Constructing kernels from kernels

Given valid kernels $k_1(\mathbf{x}, \mathbf{x}')$ and $k_2(\mathbf{x}, \mathbf{x}')$, the following kernels will also be valid:

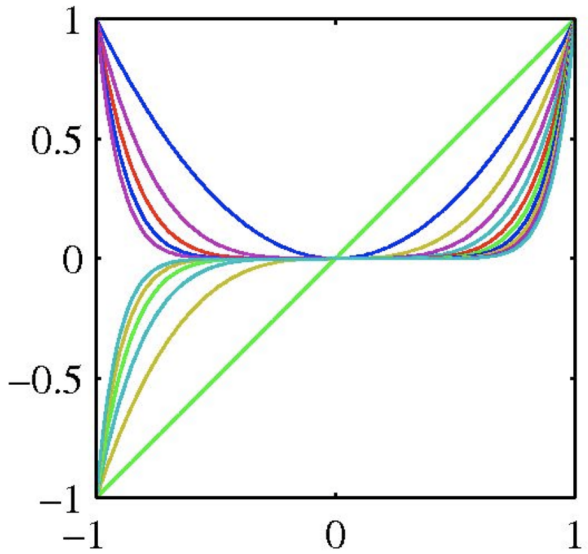
$$
\begin{aligned}
k(\mathbf{x}, \mathbf{x}') &= c k_1(\mathbf{x}, \mathbf{x}') \quad \text{for} \;\; c > 0, \\
k(\mathbf{x}, \mathbf{x}') &= f(\mathbf{x}) k_1(\mathbf{x}, \mathbf{x}') f(\mathbf{x}') \\
k(\mathbf{x}, \mathbf{x}') &= k_1(\mathbf{x}, \mathbf{x}') + k_2(\mathbf{x}, \mathbf{x}') \\
k(\mathbf{x}, \mathbf{x}') &= k_1(\mathbf{x}, \mathbf{x}') \cdot k_2(\mathbf{x}, \mathbf{x}') \\
k(\mathbf{x}, \mathbf{x}') &= \mathbf{x}^\top A \mathbf{x} \qquad (A \text{ PSD}) \\
k(\mathbf{x}, \mathbf{x}') &= \exp(k_1(\mathbf{x}, \mathbf{x}')) \\
k(\mathbf{x}, \mathbf{x}') &= q(k_1(\mathbf{x}, \mathbf{x}'))
\end{aligned}
$$

where q polynomial with ≥ 0 coefficients.

Local vs Global Kernels

Polynomial basis functions:

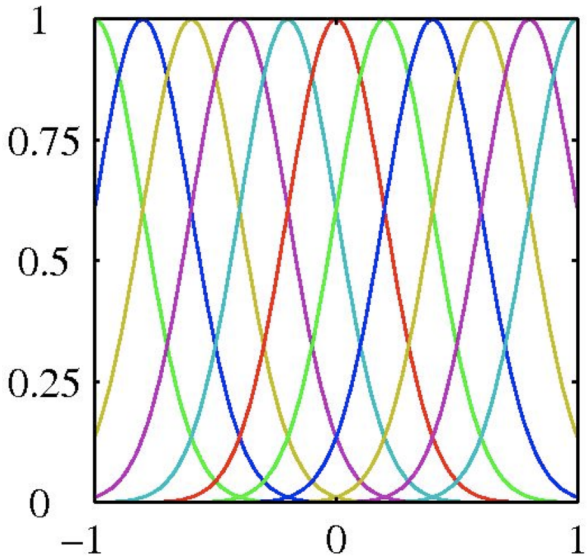
$$\phi_j(x) = x^j.$$



Basis functions are global: small changes in x affect all basis functions.

Gaussian basis functions:

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



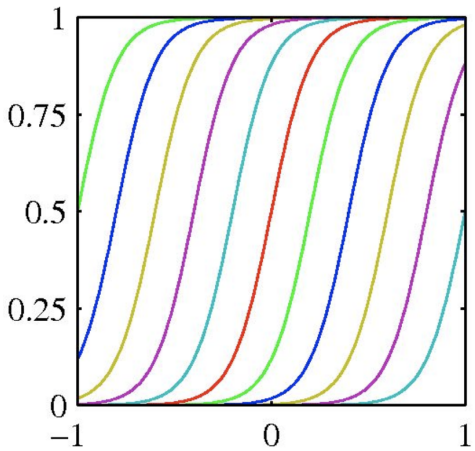
Basis functions are local: small changes in x only affect nearby basis functions.
μ_j and s control location and scale (width).

Radial basis functions

To get a better feeling for the kernel method consider the case where kernel is defined by a radial basis function.

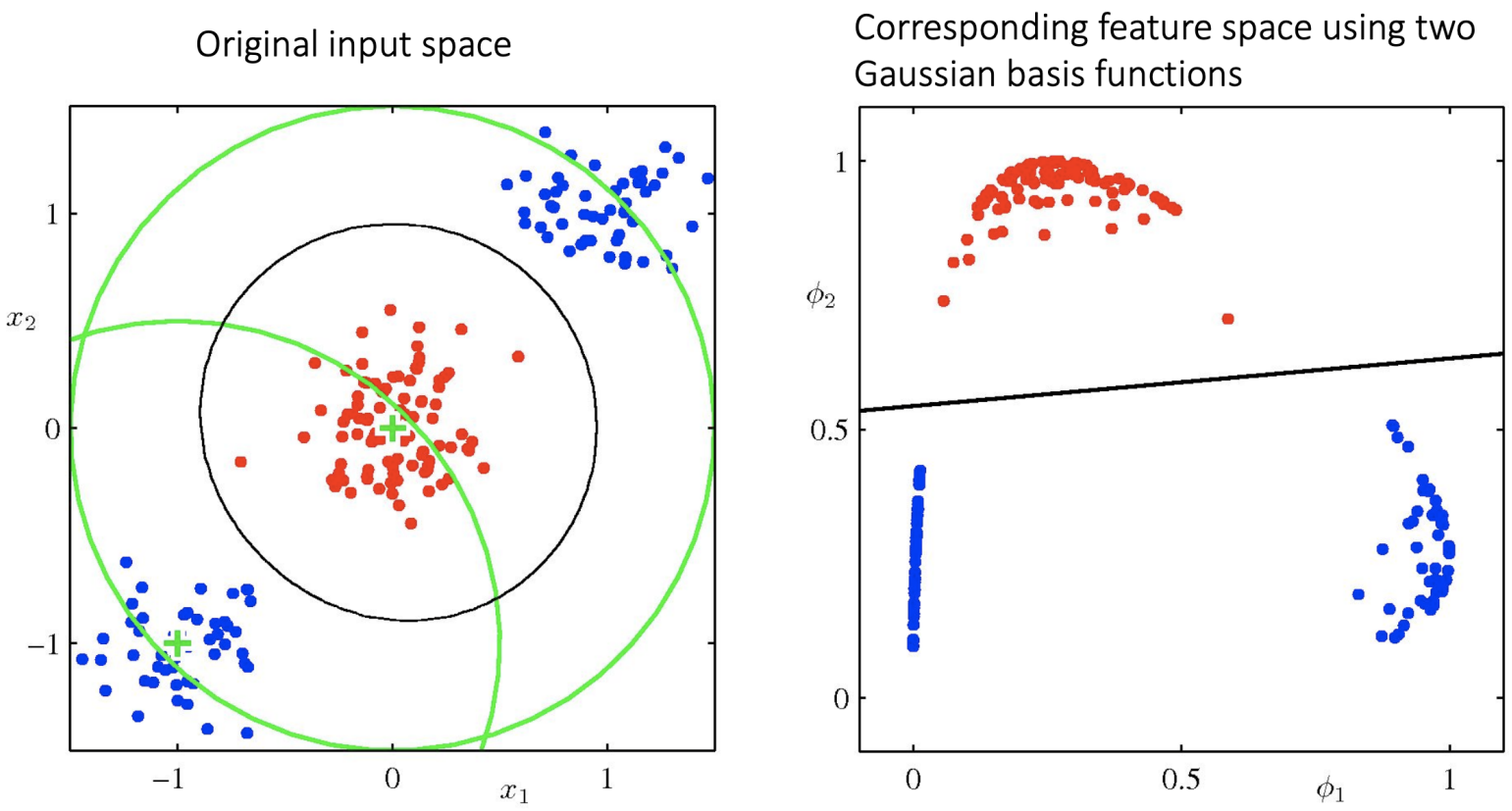
- Radial basis functions depend only on the distance from $\boldsymbol{\mu}_j$, i.e.

$$\psi_j(\mathbf{x}) = h(\|\mathbf{x} - \boldsymbol{\mu}_j\|).$$



- Sigmoidal basis functions: h is sigmoid.
- Gaussian basis functions: h is normal pdf

Example: Radial basis functions



- We define two Gaussian basis functions with centers shown by the green crosses, and with contours shown by the green circles.
- Linear decision boundary (right) corresponds to the nonlinear decision boundary in the input space (left, black curve).

Radial basis functions: motivation

- Given a set of data samples $(\mathbf{x}^{(i)}, y^{(i)})$ for $i = 1, .., N$, we want to find a smooth function f that fits data as

$$f(\mathbf{x}^{(i)}) \approx y^{(i)} \qquad \text{for } i = 1, \ldots, N.$$

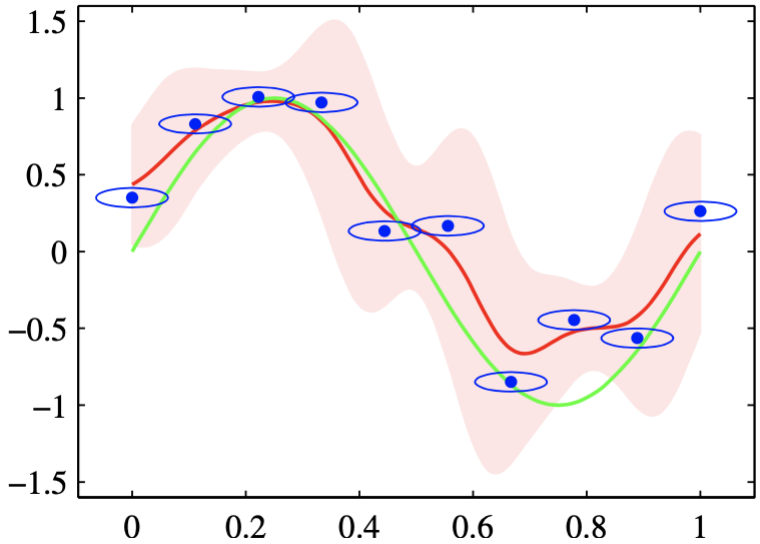
- This is achieved by expressing $f(\mathbf{x})$ as a linear combination of radial basis functions, one centred on every data point

$$f(\mathbf{x}) = \sum_{i=1}^{N} w_i h(\|\mathbf{x} - \mathbf{x}^{(i)}\|)$$

  where w_i are found by least squares.
- In practice we may use many less functions than N.

Radial basis functions: Illustration

- Kernel regression model using isotropic Gaussian kernels:

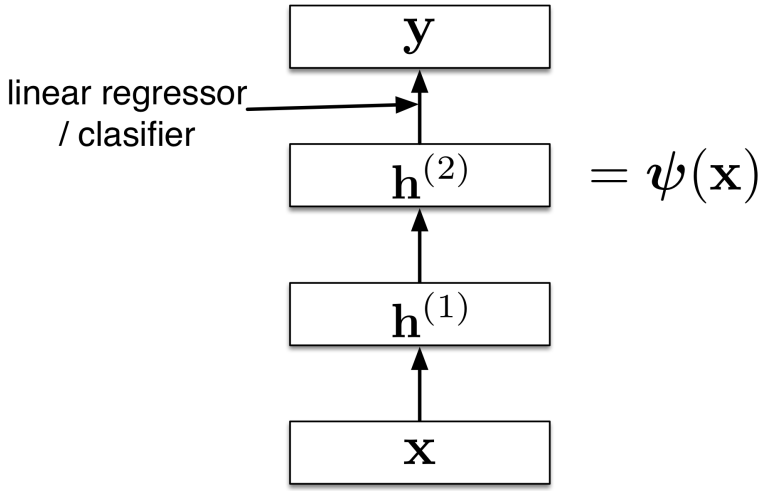


- The original sine function is shown by the green curve.
- The data points are shown in blue, and each is the centre of an isotropic Gaussian kernel.
- The resulting regression function is shown by the red line.

Neural Networks and Feature learning

Last layer in Neural networks:

- If task is regression: choose
  $\mathbf{y} = f^{(L)}(\mathbf{h}^{(L-1)}) = (\mathbf{w}^{(L)})^\top \mathbf{h}^{(L-1)} + b^{(L)}$
- If task is binary classification: choose
  $\mathbf{y} = f^{(L)}(\mathbf{h}^{(L-1)}) = \sigma((\mathbf{w}^{(L)})^\top \mathbf{h}^{(L-1)} + b^{(L)})$
- Neural nets can be viewed as a way of learning features:



Summary of the second hour

- This lecture covered the basics of kernel-based methods.
- Kernels can be used directly for regression and classification.
- These are useful functions that capture a measure of proximity between inputs, and express predictions based on this measure.
- Next week, we will continue with kernel methods and introduce Gaussian processes.