# STA 414/2104: Statistical Methods of Machine Learning II

Week 11: Kernel Methods

Piotr Zwiernik

University of Toronto

## Table of contents

## Overview

First we discuss discuss kernel methods.

- Kernel trick
- Kernel regression
- Overview of kernels

## Recap: Linear Regression

## Recap: Linear Regression

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

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

where $\psi(\mathbf{x}) : \mathbb{R}^D \to \mathbb{R}^M$ is the feature map, $\mathbf{w} \in \mathbb{R}^M$.

- We have the design matrix $\mathbf{X} \in \mathbb{R}^{N \times D}$ in input space and

$$\boldsymbol{\Psi} = \begin{bmatrix} - & \psi(\mathbf{x}^{(1)}) & - \\ - & \psi(\mathbf{x}^{(2)}) & - \\ & \vdots & \\ - & \psi(\mathbf{x}^{(N)}) & - \end{bmatrix} \in \mathbb{R}^{N \times M}$$

is the feature matrix, and predictions

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

## Linear Regression as Maximum Likelihood

- Linear regression gets probabilistic interpretation by assuming a Gaussian noise model:

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

- The MLE under the first model leads to ordinary least squares.
- We can also do full Bayesian inference as explained last week.
- 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.

## Kernel trick

## Regularized Linear Regression: towards the kernel trick

- In the ridge regression problem we minimize

$$\begin{aligned} E(\mathbf{w}) &= \frac{1}{2}\|\mathbf{y} - \boldsymbol{\Psi}\mathbf{w}\|^2 + \frac{\lambda}{2}\mathbf{w}^\top\mathbf{w} \\ \nabla E(\mathbf{w}) &= \boldsymbol{\Psi}^\top\boldsymbol{\Psi}\mathbf{w} - \boldsymbol{\Psi}^\top\mathbf{y} + \lambda\mathbf{w}. \end{aligned}$$

## Regularized Linear Regression: towards the kernel trick

- In the ridge regression problem we minimize

$$\begin{aligned} E(\mathbf{w}) &= \frac{1}{2}\|\mathbf{y} - \boldsymbol{\Psi}\mathbf{w}\|^2 + \frac{\lambda}{2}\mathbf{w}^\top \mathbf{w} \\ \nabla E(\mathbf{w}) &= \boldsymbol{\Psi}^\top \boldsymbol{\Psi}\mathbf{w} - \boldsymbol{\Psi}^\top \mathbf{y} + \lambda \mathbf{w}. \end{aligned}$$

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

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

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

## Regularized Linear Regression: towards the kernel trick

- In the ridge regression problem we minimize

$$\begin{aligned} E(\mathbf{w}) &= \frac{1}{2}\|\mathbf{y} - \mathbf{\Psi}\mathbf{w}\|^2 + \frac{\lambda}{2}\mathbf{w}^\top\mathbf{w} \\ \nabla E(\mathbf{w}) &= \mathbf{\Psi}^\top\mathbf{\Psi}\mathbf{w} - \mathbf{\Psi}^\top\mathbf{y} + \lambda\mathbf{w}. \end{aligned}$$

- 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

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

Note: $\mathbf{\Psi}^\top\mathbf{\Psi}$ is $M \times M$ and $\mathbf{\Psi}\mathbf{\Psi}^\top$ is $N \times N$.

## Kernel Ridge Regression

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

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

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

## Kernel Ridge Regression

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

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

  which we call the **kernel matrix**. Function $k(\mathbf{x}, \mathbf{x}')$ is the **kernel**.
- Therefore, we minimize (note: no unique minimum)

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

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

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

## Kernel Ridge Regression

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

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

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

- Therefore, we minimize (note: no unique minimum)

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

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

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

- Substitute back in to the linear regression model

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

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

## 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 $\mathbf{x}^{(i)}$ (large if close).
- 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 $\psi(\mathbf{x})$ (can use features of high dimension).

## Kernel regression

## Kernels: Formal definition

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

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

## Kernels: Formal definition

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

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

**Definition: Kernel function (Schoenberg 1938)**

A **kernel** $k(\mathbf{x}, \mathbf{x}')$ is any function such that for any $N \geq 1$ and for any data points $\mathbf{x}^{(i)}$ for $i = 1, \ldots, N$, the kernel matrix $\boldsymbol{K} \in \mathbb{R}^{N \times N}$ with entries $K_{ij} = k(\mathbf{x}^{(i)}, \mathbf{x}^{(j)})$ is PSD.

## Kernels: Formal definition

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

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

**Definition: Kernel function (Schoenberg 1938)**

A **kernel** $k(\mathbf{x}, \mathbf{x}')$ is any function such that for any $N \geq 1$ and for any data points $\mathbf{x}^{(i)}$ for $i = 1, \dots, N$, the kernel matrix $\boldsymbol{K} \in \mathbb{R}^{N \times N}$ with entries $K_{ij} = k(\mathbf{x}^{(i)}, \mathbf{x}^{(j)})$ is PSD.

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

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

- Feature maps define kernels but not all kernels are like that (this can be generalized to "infinite dimensional" feature maps).

## 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! (a bit of functional analysis)

## 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:

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

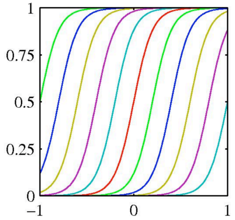
where $q$ polynomial with $\geq 0$ coefficients.

## 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.

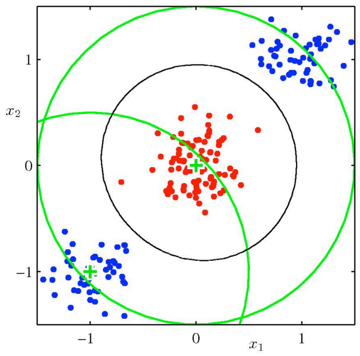
$$\psi_j(\mathbf{x}) = h(\|\mathbf{x} - \boldsymbol{\mu}_j\|).$$



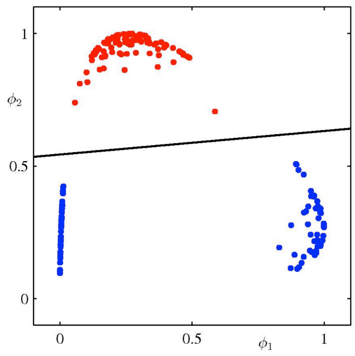
- Sigmoidal basis functions: $h$ is sigmoid.
- Gaussian basis functions: $h$ is normal pdf

## Example: Radial basis functions

Original input space



Corresponding feature space using two Gaussian 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).

## Summary of the first 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.
- In the tutorial we will try to get some more intuition and discuss explicit examples.
- Next hour we will continue with kernel methods and introduce Gaussian processes.

## Bayesian Linear Regression

## Recap: Linear Regression

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

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

where $\psi(\mathbf{x})$ is the feature map.

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

$$\mathbf{\Psi} = \begin{bmatrix} - & \psi(\mathbf{x}^{(1)}) & - \\ - & \psi(\mathbf{x}^{(2)}) & - \\ & \vdots & \\ - & \psi(\mathbf{x}^{(N)}) & - \end{bmatrix} \in \mathbb{R}^{N \times M}$$

and predictions $\hat{\mathbf{y}} = (\hat{y}(\mathbf{x}^{(1)}), \ldots, \hat{y}(\mathbf{x}^{(N)}))$

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

## Recap: Bayesian Linear Regression

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

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

## Recap: Bayesian Linear Regression

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

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

- and a Gaussian prior

$$\mathbf{w} \sim \mathcal{N}(0, \frac{1}{\alpha}\mathbf{I}_M)$$

**The prior induces a probability distribution over $\hat{\mathbf{y}}$**

$$\hat{\mathbf{y}} \;=\; \boldsymbol{\Psi}\mathbf{w} \;\sim\; \mathcal{N}(0, \tfrac{1}{\alpha}\boldsymbol{\Psi}\boldsymbol{\Psi}^\top)$$

Indeed: $\mathbb{E}(\boldsymbol{\Psi}\mathbf{w}) = \boldsymbol{\Psi}\mathbb{E}(\mathbf{w}) = 0$ and $\mathrm{var}(\boldsymbol{\Psi}\mathbf{w}) = \mathbb{E}(\boldsymbol{\Psi}\mathbf{w}\mathbf{w}^\top\boldsymbol{\Psi}^\top) = \boldsymbol{\Psi}\mathbb{E}(\mathbf{w}\mathbf{w}^\top)\boldsymbol{\Psi}^\top = \frac{1}{\alpha}\boldsymbol{\Psi}\boldsymbol{\Psi}^\top.$

## Distribution over prediction function

- In practice, we evaluate the prediction function $\hat{y}(\mathbf{x})$ at specific points, for example at the training data points $\mathbf{x}^{(i)}$ for $i = 1, ..., N$.
- So we are interested in the joint distribution of the function values

$$\hat{y}(\mathbf{x}^{(1)}), \dots, \hat{y}(\mathbf{x}^{(N)})$$

which we denote by the vector $\hat{\mathbf{y}} = (\hat{y}(\mathbf{x}^{(1)}), \dots, \hat{y}(\mathbf{x}^{(N)}))$.

## Distribution over prediction function

- In practice, we evaluate the prediction function $\hat{y}(\mathbf{x})$ at specific points, for example at the training data points $\mathbf{x}^{(i)}$ for $i = 1, ..., N$.
- So we are interested in the joint distribution of the function values

$$\hat{y}(\mathbf{x}^{(1)}), \dots, \hat{y}(\mathbf{x}^{(N)})$$

  which we denote by the vector $\hat{\mathbf{y}} = (\hat{y}(\mathbf{x}^{(1)}), \dots, \hat{y}(\mathbf{x}^{(N)}))$.
- We showed that

$$\hat{\mathbf{y}} \sim \mathcal{N}(0, \boldsymbol{K}) \qquad \boldsymbol{K} = \frac{1}{\alpha}\boldsymbol{\Psi}\boldsymbol{\Psi}^\top$$

  where $\boldsymbol{K}$ is the (scaled) Gram matrix

$$K_{ij} = \frac{1}{\alpha}k(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) = \frac{1}{\alpha}\psi(\mathbf{x}^{(i)})^\top\psi(\mathbf{x}^{(j)})$$

## Gaussian processes

## Gaussian process

**Definition:**

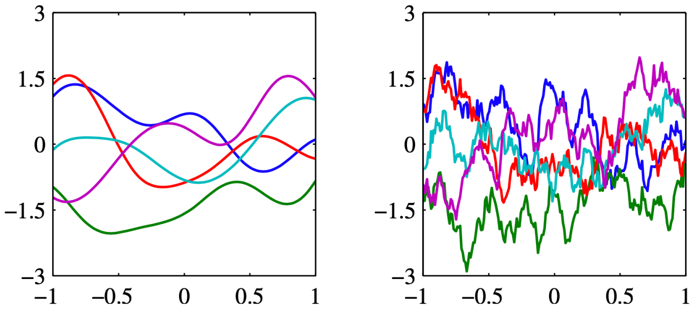
A **Gaussian process** is a probability distribution over functions $\hat{y}(\mathbf{x})$ such that for any $N \geq 1$ and any set of $N$ points $\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots, \mathbf{x}^{(N)}$ in $\mathbb{R}^D$, the vector $(\hat{y}(\mathbf{x}^{(1)}), \ldots, \hat{y}(\mathbf{x}^{(N)}))$ is jointly Gaussian.

- The joint distribution is specified completely by the second-order statistics, i.e. the mean and the covariance functions.
- In most applications, the mean function of $\hat{y}(\mathbf{x})$ can be set to zero and then the Gaussian process is completely specified by the covariance function

$$\mathbb{E}[\hat{y}(\mathbf{x})\hat{y}(\mathbf{x}')] = \frac{1}{\alpha} k(\mathbf{x}, \mathbf{x}')$$

## Gaussian process (GP)

- Directly define the kernel of a Gaussian process, not worrying about the feature map.



Samples from GP for a Gaussian kernel (left) and an exponential kernel (right).

(How do you think these plots are generated?)

## Gaussian processes for regression: what we learn from the data

- We have the linear model

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

- Given $N$ independent observations, we have

$$\mathbf{y} \,|\, \hat{\mathbf{y}} \sim \mathcal{N}(\hat{\mathbf{y}},\ \sigma^2 \mathbf{I}_N), \qquad \hat{\mathbf{y}} \sim \mathcal{N}(0, \boldsymbol{K}), \qquad \boldsymbol{K} = \frac{1}{\alpha} \boldsymbol{\Psi}\boldsymbol{\Psi}^\top.$$

- Therefore the marginal of $\mathbf{y}$ is given by

$$\mathbf{y} \sim \mathcal{N}(0, \boldsymbol{C}) \qquad \boldsymbol{C} = \boldsymbol{K} + \sigma^2 \mathbf{I}_N$$

where the corresponding kernel is

$$c(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) = \frac{1}{\alpha} k(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) + \sigma^2 \delta(\mathbf{x}^{(i)}, \mathbf{x}^{(j)})$$

$\delta(\mathbf{x}, \mathbf{x}') = 1$ if $\mathbf{x} = \mathbf{x}'$ and $\delta(\mathbf{x}, \mathbf{x}') = 0$ otherwise.

## Gaussian processes for regression: predictive distributions

- Denote now $\mathbf{y}_N = (y^{(1)}, y^{(2)}, ..., y^{(N)})$.
- We have the marginal of $\mathbf{y}_N$ given by

$$\mathbf{y}_N \sim \mathcal{N}(0, \boldsymbol{C}_N) \qquad \boldsymbol{C}_N = \boldsymbol{K}_N + \sigma^2 \mathbf{I}_N.$$

- This reflects the two Gaussian sources of randomness.

**Goal:** We want to predict for a new output $y^{(N+1)}$ given a new input $\mathbf{x}^{(N+1)}$.

- We need

$$p(y^{(N+1)} \mid \mathbf{y}_N)$$

- Note that $\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(N)}, \mathbf{x}^{(N+1)}$ are treated as constants.

## Gaussian processes for regression: predictive distributions

- We have

$$\mathbf{y}_{N+1} \sim \mathcal{N}(0, \boldsymbol{C}_{N+1}) \qquad \boldsymbol{C}_{N+1} = \boldsymbol{K}_{N+1} + \sigma^2 \mathbf{I}_{N+1}$$

  where

$$\boldsymbol{C}_{N+1} = \begin{bmatrix} \boldsymbol{C}_N & \mathbf{k} \\ \mathbf{k}^\top & c \end{bmatrix}.$$

  - Here, $c = \frac{1}{\alpha} k(\mathbf{x}^{(N+1)}, \mathbf{x}^{(N+1)}) + \sigma^2$
  - $\mathbf{k}$ is a vector with entries $k_i = \frac{1}{\alpha} k(\mathbf{x}^{(i)}, \mathbf{x}^{(N+1)})$

## Gaussian processes for regression: predictive distributions

- We have

$$\mathbf{y}_{N+1} \sim \mathcal{N}(0, \boldsymbol{C}_{N+1}) \qquad \boldsymbol{C}_{N+1} = \boldsymbol{K}_{N+1} + \sigma^2 \mathbf{I}_{N+1}$$

  where

$$\boldsymbol{C}_{N+1} = \begin{bmatrix} \boldsymbol{C}_N & \mathbf{k} \\ \mathbf{k}^\top & c \end{bmatrix}.$$

  - Here, $c = \frac{1}{\alpha} k(\mathbf{x}^{(N+1)}, \mathbf{x}^{(N+1)}) + \sigma^2$
  - $\mathbf{k}$ is a vector with entries $k_i = \frac{1}{\alpha} k(\mathbf{x}^{(i)}, \mathbf{x}^{(N+1)})$
- Since the vector $\mathbf{y}_{N+1}$ is Gaussian, we easily find $y^{(N+1)} \mid \mathbf{y}_N$.

## Property of Multivariate Gaussian Distribution

Recall:

- If we have $\mathbf{x} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ with

$$\mathbf{x} = \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \end{bmatrix} \qquad \boldsymbol{\mu} = \begin{bmatrix} \boldsymbol{\mu}_1 \\ \boldsymbol{\mu}_2 \end{bmatrix} \qquad \boldsymbol{\Sigma} = \begin{bmatrix} \boldsymbol{\Sigma}_{11} & \boldsymbol{\Sigma}_{12} \\ \boldsymbol{\Sigma}_{21} & \boldsymbol{\Sigma}_{22} \end{bmatrix}$$

- Then,

$$\mathbf{x}_2 \,|\, (\mathbf{x}_1 = \boldsymbol{a}) \sim \mathcal{N}(\boldsymbol{m}, \boldsymbol{C})$$

with

$$\boldsymbol{m} = \boldsymbol{\mu}_2 + \boldsymbol{\Sigma}_{21}\boldsymbol{\Sigma}_{11}^{-1}(\boldsymbol{a} - \boldsymbol{\mu}_1), \qquad \boldsymbol{C} = \boldsymbol{\Sigma}_{22} - \boldsymbol{\Sigma}_{21}\boldsymbol{\Sigma}_{11}^{-1}\boldsymbol{\Sigma}_{12}.$$

## Gaussian processes for regression

Recall:

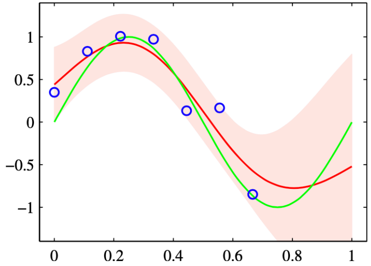
$$\mathbf{y}_{N+1} \sim N(\mathbf{0}, \boldsymbol{C}_{N+1}), \qquad \boldsymbol{C}_{N+1} = \begin{bmatrix} \boldsymbol{C}_N & \mathbf{k} \\ \mathbf{k}^\top & c \end{bmatrix}.$$

- Since $\mathbf{y}_{N+1}$ is multivariate Gaussian, $y^{(N+1)} \mid \mathbf{y}_N$ is also Gaussian with mean and variance

$$\mathrm{mean} = \mathbf{k}^\top \boldsymbol{C}_N^{-1} \mathbf{y}_N \qquad \mathrm{variance} = c - \mathbf{k}^\top \boldsymbol{C}_N^{-1} \mathbf{k}$$

- These are the key results that define Gaussian process regression.
- The vector $\mathbf{k}$ is a function of the new test input $\mathbf{x}^{(N+1)}$.
- The predictive distribution is a Gaussian whose mean and variance both depend on $\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(N)}, \mathbf{x}^{(N+1)}$.

## GPs for regression



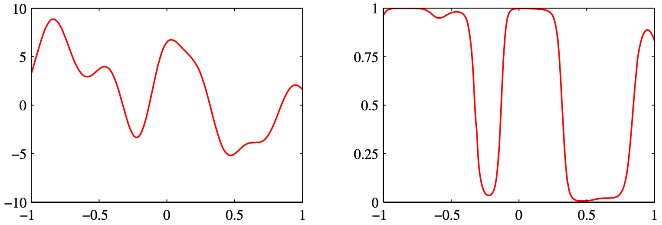
- The green curve is the true sinusoidal function from which the data points, shown in blue, are obtained.
- The red line shows the **mean** of the Gaussian process predictive distribution.
- The shaded region corresponds to plus and minus two standard deviations.

## GPs for classification

- Consider a classification problem with target variables $y \in \{0, 1\}$
- We define a Gaussian process over a function $a(\mathbf{x})$ and then transform the function using sigmoid $\hat{y}(\mathbf{x}) = \sigma(a(\mathbf{x}))$.
- We obtain a non-Gaussian stochastic process over functions $\hat{y}(\mathbf{x}) \in (0, 1)$.

## GPs for classification

- Consider a classification problem with target variables $y \in \{0, 1\}$
- We define a Gaussian process over a function $a(\mathbf{x})$ and then transform the function using sigmoid $\hat{y}(\mathbf{x}) = \sigma(a(\mathbf{x}))$.
- We obtain a non-Gaussian stochastic process over functions $\hat{y}(\mathbf{x}) \in (0, 1)$.



Left: $a(\mathbf{x})$ Right: $\hat{y}(\mathbf{x})$

## GPs for classification

- The probability distribution over target is then given by

$$p(y|a) = \sigma(a)^y (1-\sigma(a))^{1-y}, \quad y \in \{0,1\}.$$

- We need to compute

$$p(y^{(N+1)} \mid \mathbf{y}_N)$$

and notice that $a(\mathbf{x})$ is a Gaussian process but $\hat{y}(\mathbf{x})$ is not.
- We have $\boldsymbol{a}_{N+1} \sim \mathcal{N}(0, \boldsymbol{C}_{N+1})$, where

$$C_{N+1}(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) = \frac{1}{\alpha} k(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) + \nu \delta_{ij}.$$

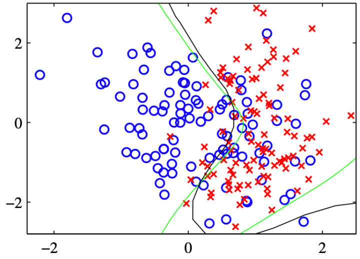
- But $\boldsymbol{a}_N$ is not observed, so we write

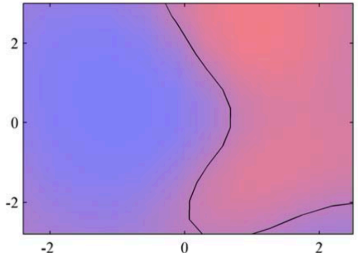
$$p(y^{(N+1)} \mid \mathbf{y}_N) = \int p(y^{(N+1)} \mid \boldsymbol{a}_{N+1}) p(\boldsymbol{a}_{N+1} \mid \mathbf{y}_N) d\boldsymbol{a}_{N+1}$$

- This is intractable. We need MCMC based methods, or numerical integration to approximate this integral.

## GPs for classification: Illustration

- Illustration of GPs for classification:





- Left: optimal decision boundary from the true distribution in green, and the decision boundary from the Gaussian process classifier in black.
- Right: predicted posterior for the blue and red classes together with the Gaussian process decision boundary.

## Learning the hyperparameters

- We didn't do any learning other than choosing a kernel!
- Rather than fixing the covariance function $\frac{1}{\alpha}k(\mathbf{x}, \mathbf{x}')$, we may prefer to use a parametric family of functions and then infer the parameter values from the data.

## Learning the hyperparameters

- We didn't do any learning other than choosing a kernel!
- Rather than fixing the covariance function $\frac{1}{\alpha}k(\mathbf{x}, \mathbf{x}')$, we may prefer to use a parametric family of functions and then infer the parameter values from the data.
- Denoting the hyperparameters with $\theta$, one can easily write down the likelihood of the Gaussian process model.

$$\log p(\mathbf{y} \mid \theta) = -\frac{1}{2}\log|\boldsymbol{C}_N| - \frac{1}{2}\mathbf{y}^\top \boldsymbol{C}_N^{-1}\mathbf{y} - \frac{N}{2}\log(2\pi)$$

- The next step is standard: gradient based optimization, grid search etc.

## Summary of the second hour

- Gaussian processes are flexible tools that can be used in regression and classification tasks.
- One can simply choose a kernel and find the predictive density!
- They can be used together with modern tools, creating powerful learning methods.