05 - Kernel Methods

During this exercise session we will analyse the use of kernel methods to perform regression and classification tasks. At first, we use Gaussian process to compute the value of the petal width given the petal length in the Iris dataset. After that, we use SVMs to classify the setosa and non-setosa flowers.

Gaussian Process

In [1]:
```
import numpy as np
from scipy.stats import zscore
from sklearn import datasets

dataset = datasets.load_iris()

x = zscore(dataset.data[:, 2]).reshape(-1, 1) # column 2 of data is petal length
y = zscore(dataset.data[:, 3]) # column 3 of data is petal width
```

Once we have our dataset, we would like to set up a GP for regression

At first we set up our kernel. In this case we choose a standard Gaussian kernel, a.k.a. RBF kernel:

$$K(\mathbf{x}, \mathbf{y}) := \phi \exp\left\{-\frac{||\mathbf{x}-\mathbf{y}||_2^2}{2l^2}\right\}$$

Moreover, we have to choose the value of the variance σ^2 of the noise of the data we are considering.

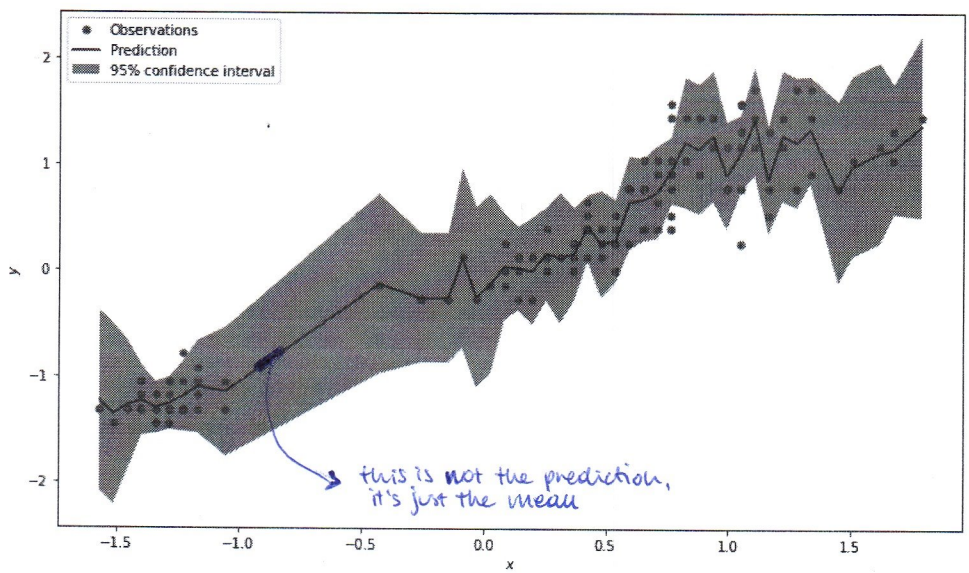
In [2]:
```
from sklearn.gaussian_process import GaussianProcessRegressor
from matplotlib import pyplot as plt
from sklearn.gaussian_process.kernels import RBF, ConstantKernel

#Kernel values
phi = 3
l = 0.02
sigma_sq = 0.2
kernel = ConstantKernel(phi, constant_value_bounds="fixed") * RBF(l, length_scale_bounds="fixed")
gpr = GaussianProcessRegressor(kernel=kernel, alpha=sigma_sq).fit(x, y)
x_pred = np.array(x)
x_pred = np.sort(x_pred, axis=0)
y_pred, sigma = gpr.predict(x_pred, return_std=True)
plt.figure(figsize=(12,7))
plt.plot(x, y, 'r.', markersize=10, label='Observations')
plt.plot(x_pred, y_pred, 'b-', label='Prediction')
plt.fill(np.concatenate([x_pred, x_pred[::-1]]),
         np.concatenate([y_pred - 1.9600 * sigma,
                         (y_pred + 1.9600 * sigma)[::-1]]),
         alpha=.5, fc='b', ec='None', label='95% confidence interval')
plt.xlabel('$x$')
plt.ylabel('$y$')
plt.legend(loc='upper left')
```

bandwidth (= it's controlling the covelation between points, it's also called "horizontal scale")

vertical scale (it affects how the Gaussian Proces behaves vertically)

it controls how the Gaussian Process will behave horizontally

Out[2]: <matplotlib.legend.Legend at 0x7fa8e134b850>



Changing the values of the kernel parameters influences the dynamic of the GP For instance if:

- set $l = 8$ we increase the smoothness of the GP
- set $\sigma^2 = 10$ we increase the noise in each point
- set $\phi = 100$ we increase the influence of the kernel to the results

In principle, one should optimize these parameters before starting the regression procedure. We can use either an independent dataset or cross-validation to estimate them, maximizing the likelihood of the considered samples.

SVM

The application of kernel methods to classification is easily exemplified in the SVMs

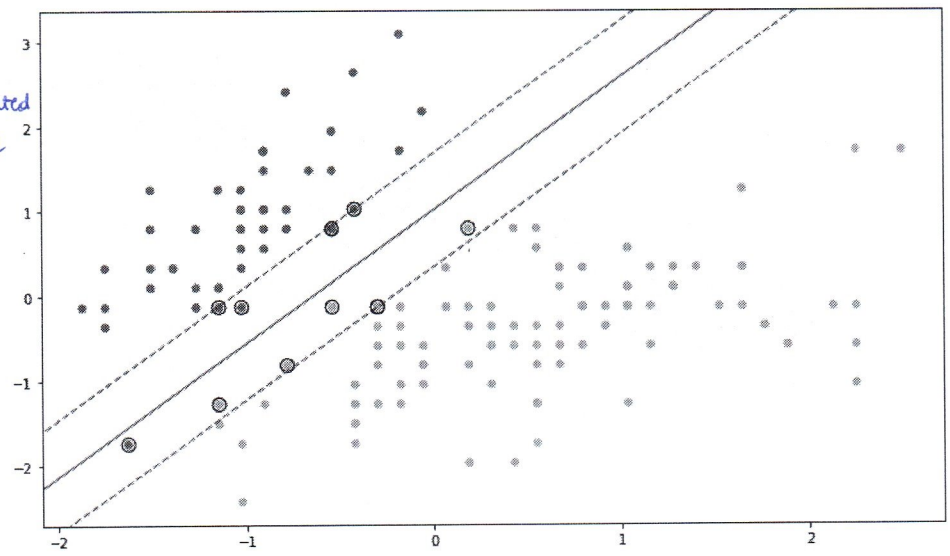
We start with the linear formulation, for which we are applying a linear kernel $K(\mathbf{x}, \mathbf{y}) = \mathbf{x}^\top \mathbf{y}$

In [3]:
```
from sklearn import svm

input = zscore(dataset.data[:, [0, 1]]) # column 0,1 are sepal length and sepal width

# the dataset is stored in the variable iris
target = dataset.target.copy()
# 0 - setosa, 1 - versicolor, 2 - virginica
target[target == 1] = 2
target[target == 0] = 1
target[target == 2] = 0

SVM_model = svm.SVC(kernel='linear')
SVM_model.fit(input, target)

#check the support vectors
SVM_model.support_vectors_
```

Out[3]:
```
array([[-0.7795133 , -0.82256978],
       [-0.29484182, -0.13197948],
       [-0.53717756, -0.13197948],
       [ 0.18982966,  0.78880759],
       [-0.29484182, -0.13197948],
       [-1.14301691, -1.28296331],
       [-1.14301691, -0.13197948],
       [-0.53717756,  0.78880759],
       [-1.02184904, -0.13197948],
       [-0.53717756,  0.78880759],
       [-0.41600969,  1.01900435],
       [-1.62768839, -1.74335684]])
```

In [4]:
```
plt.figure(figsize=(12,7))

plt.scatter(input[:, 0], input[:, 1], c=target, s=30, cmap=plt.cm.Paired)

# plot the decision function
ax = plt.gca()
xlim = ax.get_xlim()
ylim = ax.get_ylim()

# create grid to evaluate model
xx = np.linspace(xlim[0], xlim[1], 30)
yy = np.linspace(ylim[0], ylim[1], 30)
YY, XX = np.meshgrid(yy, xx)
xy = np.vstack([XX.ravel(), YY.ravel()]).T
#evaluate the SVM value for the positive class
Z = SVM_model.decision_function(xy).reshape(XX.shape)

# plot decision boundary (w^Tx =  0) and margins (w^Tx = 1 and w^Tx = -1)
ax.contour(XX, YY, Z, colors='k', levels=[-1, 0, 1], alpha=0.5,
           linestyles=['--', '-', '--'])
# plot support vectors
ax.scatter(SVM_model.support_vectors_[:, 0], SVM_model.support_vectors_[:, 1], s=100,
           linewidth=1, facecolors='none', edgecolors='k')
plt.show()
```

Why do we consider the perceptron if it is dominated by SVM? Because the perceptron is much more efficient in the case of training.

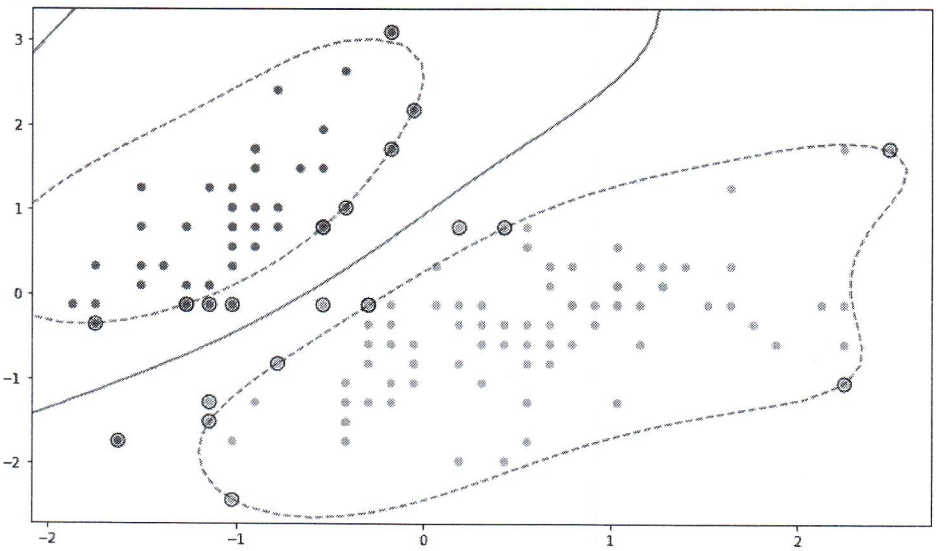


Instead, if we want to add a Gaussian kernel (or RBF):

$$K(\mathbf{x}, \mathbf{y}) = \exp\left\{-\frac{||\mathbf{x} - \mathbf{y}||^2}{l}\right\}$$

In [5]:
```
SVM_model = svm.SVC() # default kernel is 'rbf'
SVM_model.fit(input, target)
```

```
plt.figure(figsize=(12,7))

plt.scatter(input[:, 0], input[:, 1], c=target, s=30, cmap=plt.cm.Paired)

# plot the decision function
ax = plt.gca()
xlim = ax.get_xlim()
ylim = ax.get_ylim()

# create grid to evaluate model
xx = np.linspace(xlim[0], xlim[1], 30)
yy = np.linspace(ylim[0], ylim[1], 30)
YY, XX = np.meshgrid(yy, xx)
xy = np.vstack([XX.ravel(), YY.ravel()]).T
#evaluate the SVM value for the positive class
Z = SVM_model.decision_function(xy).reshape(XX.shape)

# plot decision boundary (w^Tx =  0) and margins (w^Tx = 1 and w^Tx = -1)
ax.contour(XX, YY, Z, colors='k', levels=[-1, 0, 1], alpha=0.5,
           linestyles=['--', '-', '--'])
# plot support vectors
ax.scatter(SVM_model.support_vectors_[:, 0], SVM_model.support_vectors_[:, 1], s=100,
           linewidth=1, facecolors='none', edgecolors='k')
plt.show()
```



The margins are not linear anymore. We still have the support vectors (How can we identify them?) which are those data that are used to provide a prediction.

Homeworks

Here we propose some exercises in python for you. They are not mandatory, but they can be helpful to better understand the contents of the lecture, by giving you the opportunity to develop some code by yourself.

1) Predicting petal width

Consider again the Iris dataset, and write a code to predict the petal width by using, this time, all the other features as input. Use as predictor the GP model.

2) Using GP for classification

Instead use GP as a classification tool to predict setosa vs. non-setosa flowers. Use the appropriate techniques to determinte the performance of this methods we might have on newly seen data. Is the mean squared error a good performance metric even under the probabilistic model provided by GPs?

3) Implementing SVM for multiple outputs

Extend the SVM to predict the three classes present in the Iris Dataset. Do we need to apply a specific methods to handle the three classes? Compare the results obtained with the ones provided by other classification methods presented during the course.

6 Kernel Methods

Exercise 6.1

Comment the following statements about adding new features to your model:

1. It is always a good idea to add some feature in classification since they increase the chance to consider feature spaces where it is possible to linearly separate the classes;
2. The addition of new features requires a longer time for the training of the model;
3. The addition of new features requires a longer time in prediction of newly seen samples;
4. It is not a trivial task to chose properly the features which might improve your learner capabilities;
5. You need to know the right set of features if we want to make use of them.

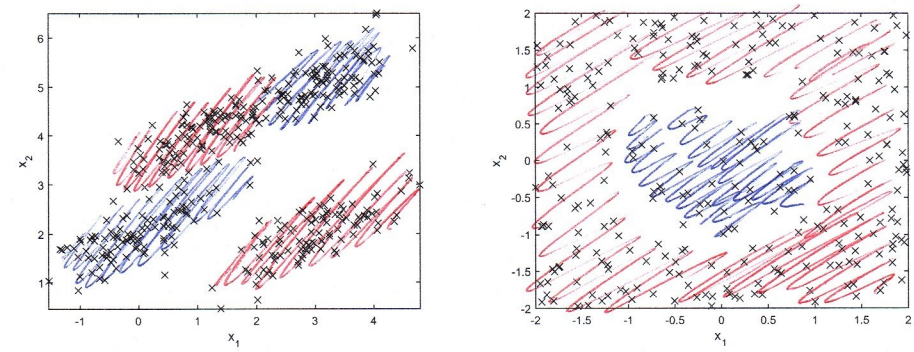
Exercise 6.2

For which one of the dataset in Figure 6.1 you would use the kernel trick to represent your data? Would you use some other methodology? Provide motivation for your choice.

Exercise 6.3

Answer the following questions about kernels. Motivate your answers.

1. Can you define a kernel over a feature set composed of colors? For instance the set could be $\mathcal{F} = \{red, green, blue, black, white\}$.
2. Can you define a kernel over a feature set composed of graphs?
3. Do you prefer to have a larger hard drive and/or a faster CPU to apply a kernel method?
4. Assume to have a non-linearly separable dataset, but you know which mapping is able project them in a linearly separable space. Are there still reasons to consider

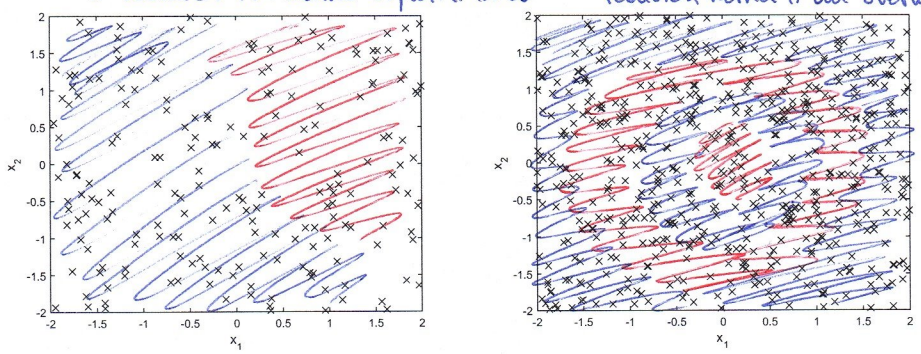


(a) (b)

YES: there is a structure in the data but we cannot exploit it in 2D

NO: we can define quadratic features, kernel is an overkill (unnecessary)

(we use kernels when we don't know the right dimension to use to do the task)



(c) (d)

NO: it seems to be linearly separable

YES: the structure is evident, however it may be difficult to understand how to set the features (not easy as in the (b) case)

Figure 6.1: Different datasets.

the use of kernels?

* Exercise 6.4

Derive the kernel formulation for the ridge regression, when we consider $\phi(x)$ as input features.

Is $k(x, x') = \phi(x)^T \phi(x') + \lambda I$ always a valid kernel?.

Exercise 6.5

Consider $x, y \in \mathbb{R}^d$, which ones of these are similarity measure:

1. $k(x, y) = x^T y$ (dot product);
2. $k(x, y) = x^T y + (x^T y)^2$;

3. $k(x, y) = ck_1(x, y) + k_2(x, y) \times k_3(x, y)$, where k_1, k_2 and k_3 are valid kernels in $\mathbb{R}^d$;
4. $k(x, y) = \log(x)e^{-y}$ $(d = 1)$;
5. $k(x, y) = x^T Ay$ with $A = \begin{bmatrix} 4 & 6 \\ 6 & 9 \end{bmatrix}$ $(d = 2)$;
6. $k(x, y) = \sqrt{(1 - \cos^2(x))}\cos(y - \pi/2)$, $(d = 1)$.

Exercise 6.6

Suppose you want to use a GP for a regression problem. You know that it varies a lot in some dimensions and less in others. Which kind of covariance kernel would you use? Provide the analytic form of the kernel and motivate why you would choose it.

There exist other techniques which are able to handle this problem? Are there any drawbacks in doing so?

Why you should not consider such a model in the case you have the information that each dimension is equivalent to the others.

Exercise 6.7

Comment the following statements about GPs. Motivate your answers.

1. The more the samples we have in a point of the input space x the more it is likely that the variance of the process decreases in x.
2. We can choose any kind of prior distribution for a GP and we are assured to reach the true function if we get enough samples.
3. Gaussian process can be used only for regression problems.
4. Far from the region where we have points the variance of the GP gets larger and larger.
5. As in linear models, we are considering different variance for the noise in each point of the input space x.

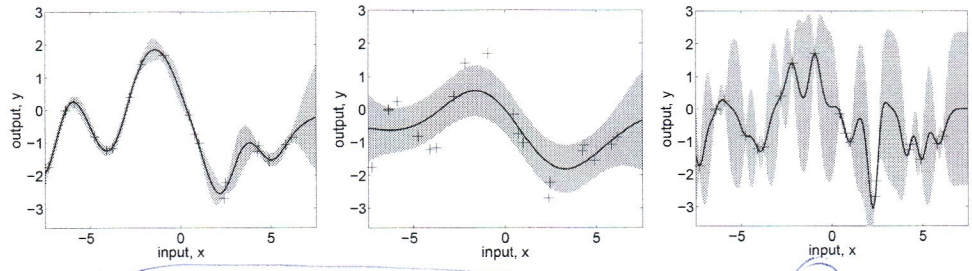
✗ Exercise 6.8

Associate the following set of parameters:

1. $\phi = 1, l = 1$ and $\sigma = 0.1$;
2. $\phi = 1.08, l = 0.3$ and $\sigma = 0.000005$;

3. $\phi = 1.16, l = 3$ and $\sigma = 0.89$;

of the Gaussian covariance $k(x, x') = \phi \exp\left(-\frac{1}{2l}(x - x')^2\right) + \sigma^2$ with the following figures:



in b) we have more vertical variance, hence we can observe longer variance by just looking at σ: 1. a 3. b

Figure 6.2: Different GPs.

the bandwidth agrees: the b) model is very smooth, a) too is smooth, just a little less than b)

when we have a point we have very small variance (it's basically deterministic), while the confidence regions go wild where we don't have points (the covariation factor is small → the bandwidth is very small) ⇒ 2.

where the shaded areas represent the confidence intervals at 95%.

Provide motivations for your answers.

Exercise 6.9

Consider the following formulation of a SVM:

- Hypothesis space: $y_n = f(\mathbf{x}_n, \mathbf{w}) = sign\left(\mathbf{w}^T \mathbf{x}_n + b\right)$;
- Loss measure: $L(X, \mathbf{w}) := ||\mathbf{w}||^2 + C \sum_i \zeta_i \qquad$ s.t. $t_n(\mathbf{w}^T \mathbf{x}_n + b) \geq 1 - \zeta_i \; \forall n,$

where ζ_i is the violations of the margin provided by sample $\mathbf{x}_i$. Answer the following questions about SVM. Provide adequate explanation for your answer.

1. If we increase the regularization parameter C in an SVM, how do you expect the margin to behave? Do they become thinner or thicker?
2. If no linear boundary can perfectly classify all the training data, this means we need to use a feature expansion. True or false?
3. The computational effort required to solve a kernel support vector machine becomes greater and greater as the dimension of the basis increases. True or false?
4. Suppose that after our computer works for an hour to fit an SVM on a large data set, we notice that x_4, the feature vector for the fourth example, was recorded incorrectly, i.e., we use $\hat{x}_4$ instead of x_4 to train our model. However, your coworker notices that the pair $(\hat{x}_4\ y_4)$ did not turn out to be a support point in the

original fit. He says there is no need to train again the SVM on the corrected data set, because changing the value of a non–support point can't possibly change the fit. True or false?

Exercise 6.10

Which of the following statements are true?

1. Suppose you have 2D input examples (i.e., $\mathbf{x_i} \in \mathbb{R}^2$). The decision boundary of the SVM (with the linear kernel) is a straight line.
2. If you are training multi-class SVM with the one-vs-all method, it is not possible to use a kernel.
3. The maximum value of the Gaussian kernel is 1.
4. If the data are linearly separable, an SVM using a linear kernel will return the same parameters $\mathbf{w}$ regardless of the chosen value of C.

Exercise 6.11

Consider the linear two-class SVM classifier defined by the parameters $w = [2\ 1]$ $b = 1$. Answer the following questions providing adequate motivations.

- Is the point $x_1 = [-2\ 4]$ a support vector?
- Give an example of a point which is on the boundary of the SVM.
- How the point $x_2 = [3\ -1]$ is classified according to the trained SVM?

Exercise 6.12

After training a logistic regression classifier with gradient descent on a given dataset, you find that it does not achieve the desired performance on the training set, nor the cross validation one.

Which of the following might be a promising step to take?

1. Use an SVM with a Gaussian Kernel.
2. Introduce a regularization term.
3. Add features by basing on the problem characteristics.
4. Use an SVM with a linear kernel, without introducing new features.

Exercise 6.13

Derive the dual formulation from the primal SVM minimization problem with soft margins.

Exercise 6.14

What's the black magic of SVMs? More specifically, which parameters we can tune to better fit a specific classification task with a SVM?

Solutions

Answer of exercise 6.1

1. FALSE: if we add unnecessary features we are increasing the variance of the considered model without having any benefit for decreasing the model bias.
2. TRUE/FALSE: it depends if the approach is parametric or non–parametric. In the first case the training generally increase in computational time. If we consider a non–parametric method, which does not require training, we do not require any computation.
3. TRUE: To compute the prediction of a new samples using more features does require a larger computational time (both for parametric and non–parametric methods), unless you are resorting to the kernel trick, which might keep the computational cost dependent only on the dataset dimension N.
4. TRUE: Usually the choice of features to be added to your model requires a priori information on the problem.
5. FALSE: If you resort to the kernel trick you project your data into a higher dimensional space without requiring to choose the specific features. It is still true that the use of the proper kernel for a specific problem might give better generalization capabilities to your learner.

Answer of exercise 6.2

In Figure 6.1a we have clearly a structure in the data, but it is difficult to evidence a set of features s.t. the classes are linearly separable. In this specific case we might consider the use of Gaussian kernels, since each class seems to be composed by a set of Gaussian in the original input space.

In Figure 6.1b it is possible to find a regularity of the two classes by considering the radius of a circle centred in $(0, 0)$. Thus a transformation of polar coordinates might do the trick. In this case it is better to use the information we extracted from the data then use an arbitrarily complex feature space induced by kernels.

In Figure 6.1c we have a linear separating hyperplane in the feature space. We should use some linear techniques directly applied in this space.

In Figure 6.1d there is a clear structure in the problem. If you are able to find a suitable feature space one might use it. Another simple option is to resort to kernels and "hope" that the problem is linearly separable in this new feature space.

Answer of exercise 6.3

1. In principle, one should define a metric in the colour space. As long as we are able to define it, we can apply kernels. The other possibility is to transform the colours into binary vectors and apply the kernel to that transformation. Generally, Gaussian kernels are not so effective on binary vectors, therefore one should choose a different shape for the kernel.

2. The same holds for graphs. We need to have a metric to understand how two graphs are similar to each other. For instance, one might use the minimum number of operations needed to transform a graph into the other.

3. The kernel does not increase the number of operations one should perform (computing a distance should be linear in the number of features), while we need to store data about all/most of the training set, therefore we might need a larger hard drive to store it.

4. No: if we have a priori information on the dataset, we should use them and project into a specifically crafted feature space. Yes: if the space in which we are projecting is an infinite/high dimensional one, we might try to use the kernel trick not to work on large feature vectors.

Answer of exercise 6.5

According to *Mercer's theorem*:

Theorem 1. *Any continuous, symmetric, positive semi–definite kernel function $k(x, y)$ can be expressed as a dot product in a high–dimensional space.*

Thus:

1. TRUE;
2. TRUE;
3. TRUE;
4. FALSE (not symmetric);
5. TRUE;
6. FALSE if $x, y \in \mathbb{R}$, TRUE if $x, y \in [2k\pi, (2k+1)\pi], k \in \mathbb{N}$.

Answer of exercise 6.6

In the case we have different scales for the different dimensions we might consider a different bandwidth for each dimension. For instance:

$$K(\mathbf{x}_i, \mathbf{x}_j) = \theta_0 \exp\left\{ -\sum_h \frac{(x_{ih} - x_{jh})^2}{2\theta_h^2} \right\},$$

where we have $\theta \in \mathbb{R}^{M+1}$ parameter vector, so that the correlation of different dimension influence differently the prediction process.

Another possible choice is to perform z-scoring on the input variables, s.t., all the variables have the same range, thus are more likely to have the same behaviour over all the dimensions. With this method we are not sure that the problem of different lengthscale is completely solved.

In the case we have prior knowledge that all the dimensions have the same behaviour we could consider a single parameter for the bandwidth, otherwise we are considering an unnecessary complex model for the data we would like to analyse.

Answer of exercise 6.7

1. TRUE/FALSE If we assume a zero variance noise, the posterior shrinks as new data are coming. Otherwise, even with infinite number of points we have that the GP in a point x has some variance.

2. FALSE As usual in ML we are assured to converge if the prior is either uninformative or generic enough, i.e., if properly integrates some prior knowledge without biasing the learning process too much. There are cases in which a wrong prior prevents from converging to the true function (e.g., null prior probability on the true parameters).

3. FALSE By considering, for instance, a logit function in conjunction with the GP we can also tackle binary classification problems (the output in this case is a probability).

4. TRUE Since the kernel usually influence more the nearest points and is does not induce any change in the behaviour of the far points (which are mainly determined by the process noise).

5. FALSE In linear models we are using a single variance for the noise throughout the entire input space x. This is also true for the GPs. The different variance is determined by the covariance structure.

Answer of exercise 6.8

By looking at the bandwidth we can associate each figure to each parameter vectors.

Indeed, the more we have larger bandwidth the more the process is smooth (i.e., the frequency over the input space of th GP is lower). Thus, the correct correspondence is:

- $1 \rightarrow (a)$
- $2 \rightarrow (c)$
- $3 \rightarrow (b)$

Answer of exercise 6.9

1. THINNER The C parameter tells the SVM optimization how much you want weight each point which is misclassified or between the margins. For large values of C, the optimization will choose a smaller-margin hyperplane since that hyperplane does a better job of getting all the training points classified correctly.
2. FALSE As in any statistical problem, we will always do better on the training data if we use a feature expansion, but that does not mean we will improve the test error. Not all regression lines should perfectly interpolate all the training points, and not all classifiers should perfectly classify all the training data.
3. FALSE Thanks to the "kernel trick" even if there is an infinite-dimensional basis, we need the N^2 inner products between training data points. The computational cost might increase if the computation of the inner products is costly.
4. FALSE When we change $\hat{x}_4$ to x_4, the fourth example $(x_4\ y_4)$ might become a support point; if so, the fit may change. However, we could check whether $(x_4\ y_4)$ is still not a support point even after correcting the value. If so, then we really do not need to train the model again.

Answer of exercise 6.10

1. TRUE The boundary is linear in the feature space as specified since it has the same shape as the one we analyse when talking about the perceptron.
2. FALSE The use of the kernel is completely independent on the method used for classifying multiple classes. Sometimes the fact that the boundaries between many classes is not jointly linear suggests to use kernels.
3. TRUE Since $\max_x e^{-x^2} = 1$.
4. FALSE The parameter C regulates somehow the thickness of the box between the margins and thus influence the boundary parameters $\mathbf{w}$ too.

Answer of exercise 6.11

A point is a support vector if $|w^T x + b| \leq 1$, thus:

$$|w^T x + b| = |-4 + 4 + 1| = 1$$

meaning that x_1 is a support vector.

A point on the boundary has to satisfy $w^T x + b = 0$ thus by considering $x_{11} = 0$:

$$2 \cdot 0 + 1 \cdot x_{22} + 1 = 0 \rightarrow x_{22} = -1$$

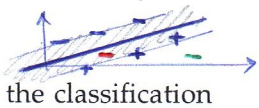
thus $x = [0 \ -1]$ is on the boundary.

A point is classified either in the positive class or in the negative one if $w^T x + b$ is positive or negative, respectively, thus:

$$w^T x_2 + b = 2 \cdot 3 - 1 \cdot 1 + 1 = 6$$

which means that the point it classified in the positive class.

! Knowing what we know, what if we get a misclassified point? Do we have to re-train?
Two cases:
- if the point is a support vector then YES
- if the point is not a support vector then NO



if we get − and we retrain nothing changes, if we get + then retraining we get new result!

Answer of exercise 6.12

1. OK In this case we are considering a new feature space where the classification task might be viable and the amount of samples we have allows us to expand our features space further.

2. NO In this case the regularization would even increase the error on the training set.

3. OK In higher dimensional space there might be the chance that the classes are linearly separable and thus that logistic regression is performing well. This clearly requires you to have a priori information about the considered problem.

4. NO With different optimization techniques and target, both logistic regression and SVM are finding a separating hyperplane. If has not been found by logistic regression, we have no chance that the SVM is able to find this separating surface.

Answer of exercise 6.13

See Page 410 of **Friedman, Jerome, Trevor Hastie, and Robert Tibshirani**. The elements of statistical learning. Vol. 1. Springer, Berlin: Springer series in statistics, 2001.

Answer of exercise 6.14

Even if it seems that the SVM does not require to be tuned, we have many different design choice to make when implementing one. More specifically:

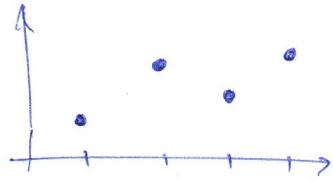
- Slack weight C: it is analogous to (the inverse of) a regularization coefficient, because it controls the trade-off between minimizing training errors and controlling model complexity. Small values for C implies strong regularization, i.e., even if we have many points between the margins we do not care so much (we have larger margins). Conversely, if we consider large values for C we are less prone to include points in the support vector set. The more we decrease the value of C, the more we are robust to possible outliers;

- Kernel: the choice of the right kernel is fundamental to have good performance in an SVM. The choice might be driven by additional information about the problem we are considering or by crossvalidation. Note that it is not possible to provide a general methodology to select the kernel and the choice might dramatically influence the classification performances;

- Kernel parameters (if present): for instance, if we consider a a Gaussian kernel we might change the bandwidth. Indeed, by using a wider kernel we are less prone to overfitting, but if we select a unnecessarily large kernel we might incorporate too few information from the training set (i.e., in a Gaussian Kernel setting a large bandwidth is equivalent to have a uniform prior over the input space).

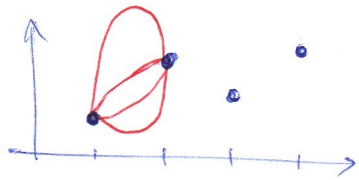
KERNEL METHODS — GAUSSIAN PROCESS (idea)

suppose we know that a function passes through some points (we know how it behaves in some points):



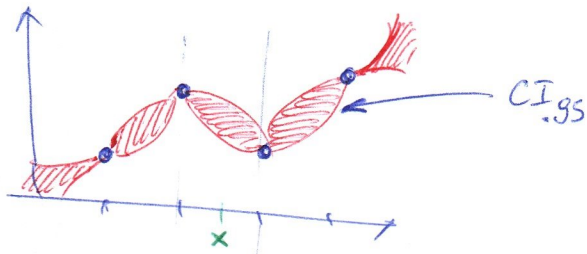
we don't know how the function goes from one point to the other (we don't know how it behaves in the connecting regions)

We may suspect some covelations among points and so we can suppose some possible functions:

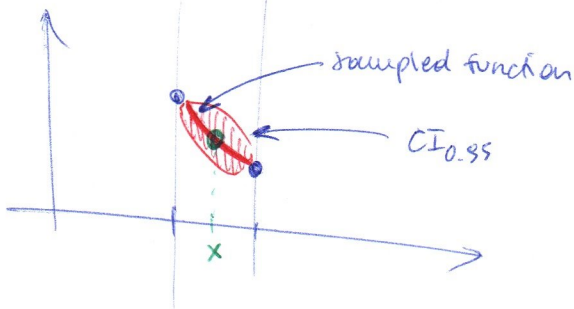


The gaussian process "says" that we don't have to guess one function, we have a distribution over possible f ctions.
(prob. distr.)

We can build confidence intervals, where we think that with a certain prob. the function will be inside the confidence intervals (e.g. 95%)

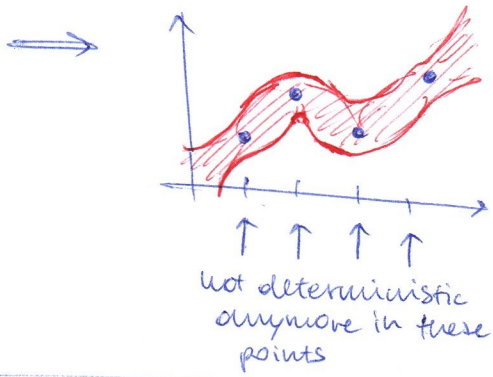


Then someone may ask: what is the value of the regression function in x?
We sample one of the function belonging to $CI_{0.95}$ and we say the value of the sampled function in x:



We can see it as a probability distribution over functions **or** as a collection of infinite-many random variables (one for each input in the input space (x))

We can also say that also for the point that we have it was a sampled point:



basically for a new point to predict we have a gaussian

The gaussian process is the distribution of the targets given the inputs

---

In the case of Gaussian process we consider the distribution as gaussian, otherwise it can be anything else (general kernel method)