Machine Learning: algorithms, Coding Lecture 2

Data reading and regression

Nicky van Foreest

April 29, 2021

Contents

1 Some things to discuss

1.1 Design desidarata of the course

1. You have to understand how the algorithms work, so that `sklearn` is not a black box for you.
2. The lectures should be spread over the weeks, so that you have time for other courses and your thesis.
3. The course should be finished in the last week, so that you have time left for your thesis, and go on holidays in July.
4. Sundays are for picnics

All of these constraints are reasonable, but combined we get a course in which you only have 2 weeks or so at the end to implement your algorithms and do all the work.

I don't want this, so what to give up?

1.2 Proposal for a small change

Let's give up point 1: You just go ahead, and use `sklearn` or whatever other toolbox, to your heart's content.

Aico and I take up the challenge to make the course material interesting so that:

1. you like to ponder about the topics during Sunday picnics.
2. you are going use the topics for your master thesis.
3. You like to read Kroeze in its entirety (including SVM and kernel methods) during your holidays.
4. You cannot resist the temptation later in life to really read other resources on machine learning.

So, we give in a bit now, in the hope that in a larger sense we really achieve what we want, and that is to get you hooked!

1.3 Remarks

- Is there a need for a shared document (on github?) in which we can post questions and exchange answers?
- I added some simple exercises for you to the lecture notes of my first lecture, the lecture of last Friday. They are real simple, important, and interesting. Check out lecture 1 again on github.

2 Feedback on topic choice

2.1 Project plan

I realize making a project plan is not simple, but:

- Later, your client expects you to make a project plan and an cost overview before the actual project begins.
- You have to get used to estimating what to do, how to do it, how much time it takes, without knowing the details.
- I realize that we did not tackle all tools and methods, but I want you to get started by thinking what you like to do. Like this you spread the work.
- Even though you might find it hard not to know what to do, some of you made an excellent plan. So, the challenge was not too hard.
- The grade I gave is just a sign, nothing serious.

2.2 Proposal 1.

A group turned in the text below.

'We would like to focus on the Ridge regression method which. We are not quite certain about the topic yet. For the relevance we would like to apply it on a COVID time series (for example the number of hospitalized people such that we can say something about the number of beds that is needed). However, we need to do a bit more research on the method (its strengths and weaknesses) as we have not yet discussed it in class. As Ridge regression is a common topic in supervised machine learning, we think that is suitable to study in this course as well.'

What's your opinion? Any suggestions for improvement?

2.3 Proposal 1, things to improve

- First sentence not ok.
- No title, authors, no report template
- No real topic choice. Data set?
- Why start with a method? Methods are not of central interest, problems are interesting.
- How to test? Compare?
- I want to see data sources and papers explicitly mentioned. This is essential for academic integrity: you should enable other people to redo your work.

2.4 Proposal 2.

Topic Heart disease prediction

We chose this topic because ischemic heart disease, i.e. heart disease due to a lack of blood flow to the heart, is a major cause of death both globally and in the Netherlands. We think this is a suitable topic for this course as we can attempt to predict the narrowing of heart arteries based on observed data. The results of this fit can in turn be used to predict the state of the heart arteries on other patients to save on expensive tests.

General intro According to the Global Health Data Exchange (url{httpghdx.healthdata.org}), the main cause of death worldwide in 2019 are cardiovascular diseases. Furthermore, this is emphasised by the WHO, who wrote in their recent article (url{httpswww.who.intnews-roomfact-sheetsdetailthe-top-10-causes-of-death}) that the leading cause of death in 2019 was ischaemic heart disease, which is a heart disease that results from a lack of oxygen to the heart. It is obvious that being able to accurately predict such a disease is of major importance.

Data description The data used was donated in July 1988 by Robert Detrano, M.D., Ph.D. It describes medical parameters of 303 patients admitted to Cleveland Clinic Foundation . The archive can be found here url{httparchive.ics.uci.edumldatasetsHeart+Disease}. The variables of interest are

- Age: Lifetime in years
- Sex: dummy variable for gender (1 for male, 0 for female)

- CP: (Chest Pain) Type of chest pain. 1 typical angina, 2 atypical angina, 3 non-anginal pain and 0 asymptomatic angina (= None).

Later we could try to extend this with smoking experience family history and diagnosis on location in any of the heart biggest vessels. Furthermore, there also exists similar datasets on hospitals in Hungary, Switzerland California. Those datasets could perhaps be used for validation and testing (if they are sufficiently complete).

Proposed Methods The dataset has been found by other researchers. Early and notably by, cite{detrano1989international}, who achieve a 77 correct classification accuracy with a logistic-regression-derived discriminant function, cite{aha1988instance} who find a 74.8 accuracy and cite{gennari1989models} who's texttt{CLASSIT} conceptual clustering system achieved a 78.9% accuracy.

We will try to tackle this problem using classification trees, of which a wide variety are available. If applicable it could also be interesting to let a (Bayesian) Graphical Network give diagnosis predictions. These methods somewhat mirror a human doctor trying to diagnose. For example, first pruning chest pain, then probing blood pressure or lung function and measuring cholesterol to finally arrive at a diagnosis. Depending on the construction of the graphs random forests can be a very promising research extension.

Benchmarking It seems reasonable to benchmark the proposed methods against OLS regresession, ridge regression andor probitlogit estimation.

2.5 Proposal 2, Things to improve

- There is nothing to improve.
- All earlier points are met.
- There are also short section titles to help structure the text.
- Great example !

3 Overview

- Last lecture
  - What (not) to program (for this course)
  - Data analysis, example: grade computations
  - Tools to help setup tests
  - Linear regression, Spotify song popularity
- This lecture:
  - Regression with Gradient descent
  - Some bad code, and how to repair
  - Kernel methods
- Next lecture:
  1. Code of DSML, explain and improve
  2. Ridge and Lasso, if time permits.

4 Reading exam and assignment grades, some basics of Python

```python
def get_exam_grade():
    fname = r"gc_EXAM-EBP038A05-20210330_column_2021-04-13-15-27-45.xls"
    df = pd.read_excel(fname, sheet_name=0)  # , skiprows=0)
    number = df["Username"].apply(lambda x: int(x.replace("s", "")))
    total = df["Total"].replace(np.nan, 0)
    return {s: t for s, t in zip(number, total) if t > 0}
```

- Use the pandas library to read data
- `lambda x: int(x)` is an anonymous function. Extremely practical for one-off tasks
- Checks on `nan` (= 'not any number')
- Export only the students with a grade `t>0`.
- Return a `dict`. Practical for look ups and merge data from various sources.
- Reading the assignments works similar.

5 Regression with Gradient descent

5.1 Motivation

- General technique, intuitive
- The convex optimization tools we are going to use for LASSO use similar (but much better) ideas.
- Deep learning uses similar ideas to train neural networks
- A good recap of a topic you learned in Multi Variate Calculus!

5.2 Gradient descent

- Goal is to optimize $f(\beta) = ||y - X\beta||^2$.
- Idea: Take with some initial β^0.
- Update to β^1 by moving from β^0 in the direction of the gradient of~f.
- The gradient: $\nabla f(\beta) = -2X'(y - X\beta)$.
- Choose a *learning parameter* α.
- Update β^n to a new vector according to the scheme:

$$\beta^{n+1} = \beta^n - \alpha \nabla f(\beta^n). \tag{1}$$

5.3 Some interesting maths, from a book

This is from some book that I found on the internet. The task is to optimize

$$l = \frac{1}{N}\sum_{i=1}^{N}(y_i - (wx_i + b))^2, \tag{2}$$

where their w is our β, their b is β_0. They don't use the augmented X (`[1, X]`). This is what they write for their updating scheme:

$$w_i \leftarrow \alpha(-2x_i(y_i - (w_{i-1}x_i + b_{i-1}))/N \tag{3}$$

$$b_i \leftarrow \alpha(-2(y_i - (w_{i-1}x_i + b_{i-1}))/N. \tag{4}$$

What's wrong?

- The use of the i's must be wrong.
- Why all the ugly brackets?

We must be able to do better.

5.4 Implementation, from the same book

They build it like this.

```
dl_dw, dl_db, N = 0, 0, len(spendings)

for i in range(N):
    dl_dw += -2 * spendings[i] * (sales[i] - (w * spendings[i] + b))
    dl_db += -2 * (sales[i] - (w * spendings[i] + b))

w = w - (1 / float(N)) * dw * alpha
b = b - (1 / float(N)) * db * alpha
```

What's not OK here?

- I have no clue how the algo works. What are `spendings`, `sales`? Why not use the math notation?
- Why is the α at the end?
- Why cast N to a `float`, and put it at the front?
- Inconsistent use of `-=` and `+=`. Ugly!

5.5 Repair 1

I modified the code to follow the math notation, so that I at least understand with it means.

```
dw, db = 0, 0

for i in range(N):
    dw += -2 * x[i] * (y[i] - (w * x[i] + b))
    db += -2 * (y[i] - (w * x[i] + b))

w -= dw * alpha / N
b -= db * alpha / N
```

What's still not OK here?

- (for) loops are very slow in R and (perhaps a tiny bit less in) python. Difference can be a factor 200-1000 with C++ and Fortran. Just use numpy right away, as this uses C++ and Fortran.

5.6 Some good and bad code from the internet

```
m, n = X.shape

def update_weights(self):
    Y_pred = self.predict(self.X)
    dW = -(2 * (self.X.T).dot(self.Y - Y_pred)) / self.m
    # ...
    return self
```

- OK: Use `numpy` for matrix vector computations.
- Very wrong: What are `m` and `n` (again, why not use the math notation?) After some thought, I realized that here `n` is what most people use for `p`. This is extremely error-prone. Stick to conventions.
- Ugly and confusing: too many brackets.
- Old python notation: use of `dot` rather than python's `@` operator for matrix multiplication
- Strange: Why the `return self`? It's unnecessary, so why? I get confused about the author's intentions.

5.7 Bad example, continued

Right below the code of the previous slide, the author writes this:

```
# Hypothetical function h( x )
def predict(self, X):
    return X.dot(self.W) + self.b
```

What's very wrong?

- The suggestion is that you can use any (hypothetical) function to predict, i.e, $y = h(x)$.

- But in the computation of the gradient we use the specific form $h(x) = wb + b$.

It's simply not true that we can use the update scheme as coded earlier. If we want to a general h, we need to recompute ∇h.

5.8 Implementation, test

Since we know what β^* is, let's fill it in see whether $\nabla f(\beta^*) = 0$. Of course, this is an exceptional situation; normally we don't know β^*.

```
beta = solve(X.T @ X, X.T @ Y)
d_beta = -2 * X.T @ (Y - X @ beta) / n
print(d_beta) #
```

(This code builds on the code of the previous coding lecture. You should copy it after that code to run it.)

5.9 Implementation, a run

I needed quite a bit of experimentation

- Setting α to some proper value is not entirely straightforward. When α is very small, the convergence speed is bad, when α is too large, the solution explodes.
- Determining the number of iterations is also not simple.

```
alpha = 0.00007
beta = (0, 0)

for i in range(2_000_000):
    d_beta = -2 * X.T @ (Y - X @ beta) / n
    beta -= alpha * d_beta

print(beta)
```

This gave `[-159.43614866 1.61369709]`, while the solution of $X'X\beta = X'y$ gives `[[-329.85141724 3.07913119]`. The difference is still large after $2 \cdot 10^6$ runs. With $\alpha = 0.00008$ the solution explodes.

5.10 Conclusion about gradient descent

- The idea behind gradient descent is nice and intuitive
- Don't use it if you want to get some work done.
- Either we have to learn about smarter, but more difficult, algorithms (appendix of Kroeze et al., which I find hard to read, BTW.). Or, get optimization algorithms from a libary. The latter option is to be preferred.
- These more advanced methods are still based on the same ideas, so our work is not lost; it's simply not good enough.

- In these notes, we follow straightforward, hence algorithmically dumb, procedures. The intent is to first learn the simple things.
- Once we know how to deal with the simple things, we can optimize our own code for speed, or perhaps copy it from others.
- But, be aware, not all code you find on the web is correct, or smart, or robust.
- Test!

6 Kernel regression

6.1 Kernels

To understand something, I prefer to explain it in my own words. Here is my attempt to explaining kernels.

For a kernel function k, the kernel estimator for the density is given by

$$\hat{f}_X(x) = \frac{1}{n}\sum_{i=1}^{n} k\left(X_i - x\right), \tag{5}$$

where $\{X_i\}$ is a set of observations.

A kernel k is a nice function that satisfies in particular

$$k \geq 0, \qquad \int k(x)\,\mathrm{d}x = 1, \qquad \int xk(x)\,\mathrm{d}x = 0. \tag{6}$$

For example, consider the following kernel function,

$$k(x) = I_{|x|<1/2}. \tag{7}$$

Clearly, this just count the number of observations in a box of width 1 around 0.

Instead of the kernel k, we can also take $k_h(x) = k(x/h)/h$ Hence, by changing h, we just modify the width of the box. Thus, h is a *hyper parameter* that we can/have to tune to the problem. To maintain the property that $\int k(x)\,\mathrm{d}x = 1$ we also have to scale the height.

Observe that this is the same procedure as we follow when we integrate numerically an integral. To see why, realize that we can interprete the notation $f(x)\,\mathrm{d}x$ as the probability mass in a box of size $[x-1/h, x+1/h]$, i.e, $F(x+1/h)-F(x-1/h)$, times the width $\mathrm{d}x = 1/h$. And then we sum over all boxes.

6.2 Implementation of the uniform kernel

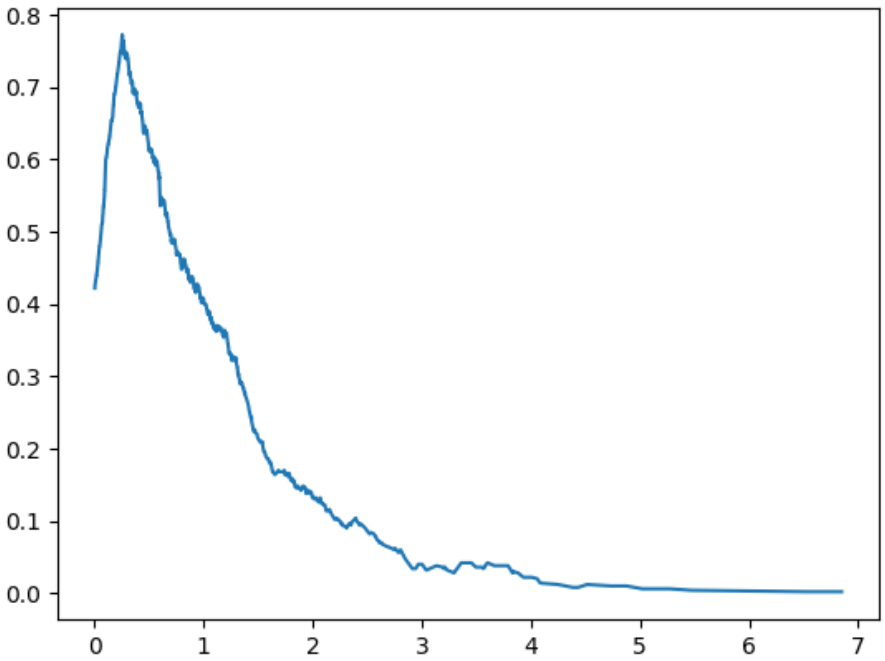
```
import numpy as np


class One_D_Kernel_estimator:
    def __init__(self, X, h=1):
        self.X = X
        self.h = h

    def kernel(self, x):
```

```
        return (np.abs(x / self.h) < 0.5) / self.h

    def predict(self, x):
        return self.kernel(self.X - x).sum() / len(X)
```

Note: If you like to see how things work, include a few print statements.
Note: classes are very useful ideas in software development.

6.3 Illustration

```
import matplotlib.pyplot as plt

np.random.seed(3)

X = np.random.exponential(1, size=1000)
f_X = One_D_Kernel_estimator(X, h=0.5)

X.sort()
y = [f_X.predict(x) for x in X]

plt.clf()
plt.plot(X, y)
# plt.show()
fname = "figures/one_D_kernel.png"
plt.savefig(fname)
```



Remarks:

- This kernel does not estimate the exponential density well around 0. To handle the asymmetry around 0 it is necessary to modify the kernel around 0. This may be tricky, because kernels are supposed to be symmetric. For the moment we will ignore this.
- DSML Section 4.4 discusses this point. They provide a faster kernel (KDE theta) that does not seem to suffer from this problem.
- The above implementation is clear and efficient for the computation of a single x. However, to make the graph it's quite bad, since we use an algorithm with $O(n^2)$ ($n = \dim(X)$) complexity. With a bit of thought (interesting challenge!) it must be possible to make the graph with an $O(n)$ algorithm.

7 Nadaraya Watson, 1D

7.1 Model (Continuation of my explanation)

Recall what we like to achieve: to come up with a predictor of Y given an observation x. So, suppose we have a function $g : \mathbb{R} \to \mathbb{R}$, we would like to find a region A in $\mathbb{R}$ such that when $x \in A$, we classify (or estimate) $y = g(x)$ as belonging to class 1, and when $x \in A^c$ we classify it as 2, or perhaps some other values.

Before we build the Nadaraya-Watson estimator for this problem, let us derive it. We need to estimate

$$g(x) = \mathsf{E}[Y|X = x] = \int y f_{Y|X}(x|y)\,\mathrm{d}y = \int y \frac{f_{X,Y}(x,y)}{f_X(x)}\,\mathrm{d}y, \tag{8}$$

where the last step follows from the definition of conditional density. Now replace:

$$f_{X,Y}(x,y) \approx \hat{f}_{X,Y}(x,y) = \frac{1}{n}\sum_{i=1}^{n} k_h(X_i - x)\, k_h(Y_i - y), \tag{9}$$

$$f_X(x) \approx \hat{f}_X(x) = \frac{1}{n}\sum_{i=1}^{n} k_h(X_i - x). \tag{10}$$

BTW, who can explain me why there is a split in X and Y, why don't we take $k_h(|(X_i, Y_i) - (x,y)|)$?

With this,

$$\int y \hat{f}_{X,Y}(x,y)\,\mathrm{d}y = \frac{1}{n}\sum_{i=1}^{n} k_h(X_i - x) \int y\, k_h(Y_i - y)\,\mathrm{d}y. \tag{11}$$

But,

$$\int y\, k_h(Y_i - y)\,\mathrm{d}y = \int (u + Y_i)\, k_h(u)\,\mathrm{d}u = Y_i, \tag{12}$$

by the kernel properties $\int xk(x)\,\mathrm{d}x = 0$ and $\int k(x)\,\mathrm{d}x = 1$. Therefore,

$$\int y \hat{f}_{X,Y}(x,y)\,\mathrm{d}y = \frac{1}{n}\sum_{i=1}^{n} k_h(X_i - x)\, Y_i, \tag{13}$$

by which we get the following (Nadaraya-Watson) estimator for $g(x)$

$$\hat{g}(x) = \frac{1}{n}\sum_{i=1}^{n} \frac{k_h\left(X_i - x\right) Y_i}{\hat{f}_X(x)}, \tag{14}$$

$$= \frac{\sum_{i=1}^{n} Y_i k_h\left(X_i - x\right)}{\sum_{i=1}^{n} k_h(X_i - x)}. \tag{15}$$

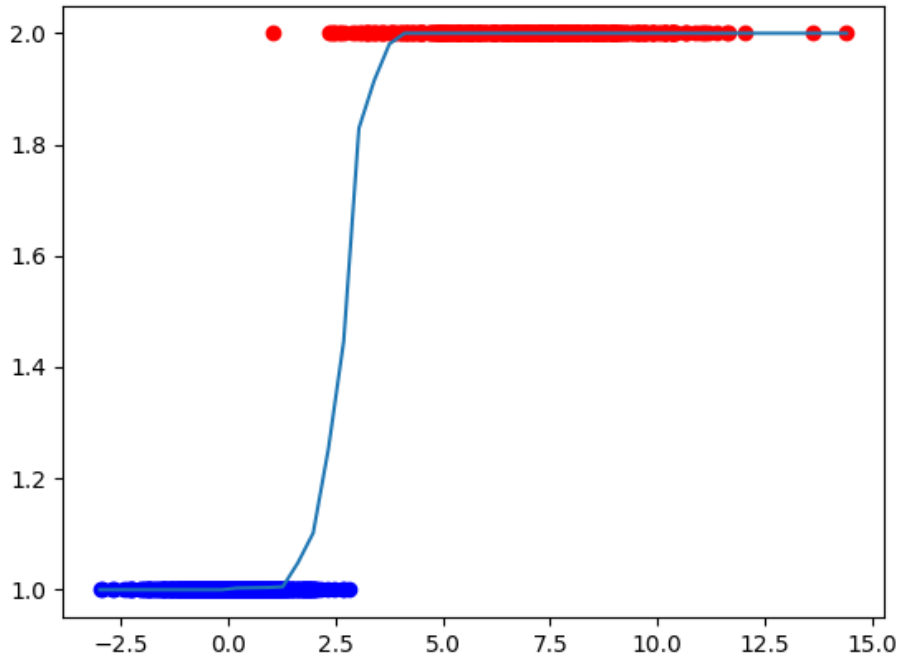
7.2 Implementation

```python
import numpy as np

np.random.seed(3)


class Nadaraya_Watson:
    def __init__(self, X, Y, h=1):
        self.X = X
        self.Y = Y
        self.h = h

    def kernel(self, x):
        return (np.abs(x / self.h) < 0.5) / self.h

    def predict(self, x):
        res = self.Y @ self.kernel(self.X - x)
        res /= self.kernel(X - x).sum()
        return res
```

7.3 Illustration of how to use the NW estimator

```python
import matplotlib.pyplot as plt

num = 500
mu1, mu2 = 0, 7
sigma1, sigma2 = 1, 2


X = np.ones(2 * num)
Y = np.ones(2 * num)
X[:num] = np.random.normal(mu1, sigma1, size=num)
X[num : 2 * num] = np.random.normal(mu2, sigma2, size=num)
Y[num : 2 * num] = 2
# print(X, Y)

nw = Nadaraya_Watson(X, Y, h=2)

xx = np.linspace(X.min(), X.max(), num=50)
yy = [nw.predict(x) for x in xx]

plt.clf()
plt.plot(xx, yy)
plt.scatter(X[:num], Y[:num], c="blue")
plt.scatter(X[num:], Y[num:], c="red")
```

```
fname = "figures/one_D_nw.png"
# plt.show()
plt.savefig(fname)
```



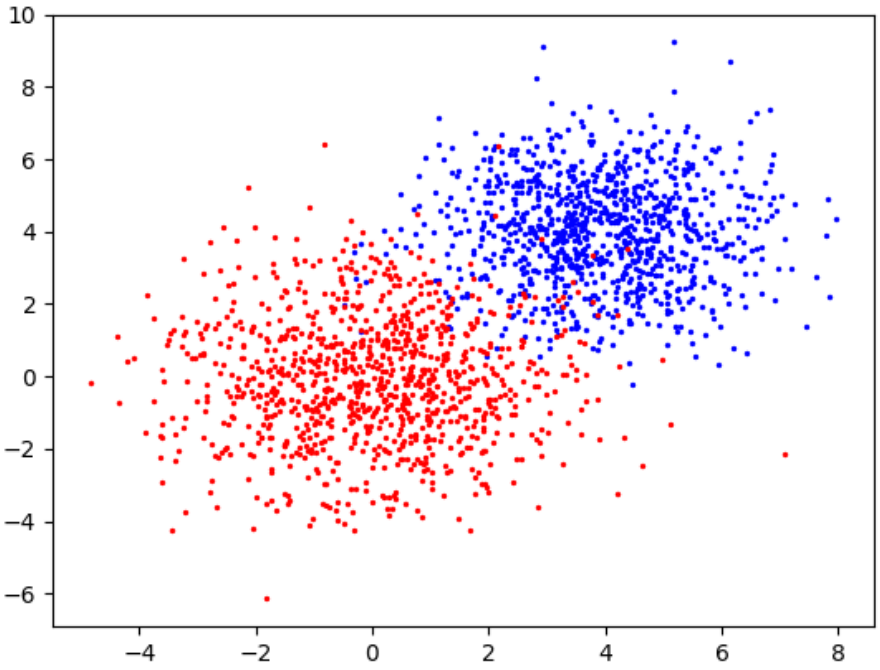
Remarks:

- We see that g moves from 1 to 2 for increasing x, as it should, given that $\mu_1 = 0$ and $\mu_2 = 7$.
- The present algorithm is not protected against an input x that lies quite a bit outside the interval $[\min\{X\}, \max\{X\}]$. In fact, when x is such that $\hat{f}_X(x) = 0$, then we run into a problem. This happens in particular when $n = \dim X$ is small and we use the uniform kernel. Perhaps it's better for this reason not to use the uniform kernel. However, I don't know, so you should test, and check the literature, for instance DSML.4.4.
- To deal with corner cases, it is better to use the implementation of a tool box. (But at least now we understand the basics.)

8 Nadaraya Watson, 2D

An interesting challenge for you: extend the NW estimator to 2D. For instance, you can try to write code to classify a new point $x = (2, 4)$ as red or blue based on the following data (see the graph).

```
np.random.seed(3)


num = 1000
mean = [4, 4]
cov = [[2, 0], [0, 2]]
X1 = np.random.multivariate_normal(mean, cov, num)

mean = [0, 0]
cov = [[3, 0], [0, 3]]
X2 = np.random.multivariate_normal(mean, cov, num)

plt.clf()
plt.scatter(X1[:, 0], X1[:, 1], c="blue", s = 2)
plt.scatter(X2[:, 0], X2[:, 1], c="red", s=2)
fname = "figures/NW_2D_data.png"
plt.savefig(fname)
# plt.show()
```



Hint, we map a point in $\mathbb{R}^2$ to $\{1, 2\}$, i.e., the two labels red and blue. From the 1D case we already have a mapping from $\mathbb{R}$ to $\{1, 2\}$. The only thing we have to do is to extend the kernel function k from 1d to 2d. Once choice is to take $k(x) = I_{|x|\leq 1/2}$, but such that $|x| = \sqrt{x_1^2 + x_2^2}$, i.e., the pythagorean distance in 2d.

9 General advice

9.1 Good Coding:

- Let the code follow the math notation. This saves lots of documentation, and confusion, and makes the code generic.
- Stick to notational conventions.
- Don't use overly long variable names
- Use consistent symbols, operators
- If you use shortcuts, explain what they do
- Study good code of others. Your own style and knowledge will increase.
- Understaning code is (hard) work, just like mathematics.

9.2 Code/books and advice from the internet

- Be very careful with using code and examples from the internet
- Don't believe anything from books that say you can learn machine learning in 20 seconds, 20 minutes, 20 hours, or 20 days.
- Best is to rely on heavily used libraries such as `numpy`, `sklearn`, etc.

10 Exercises

The exercises are not hard, but your programming skills will improve.

- Implement a Gaussian kernel, see DSML section 4.4, for the 1D case and compare with the kernel we used here. Use the code of DSML.4.4 as inspiration.
- Extend the NW estimator to tackle 2d regression (or classification) problems.