10907 Pattern Recognition

Lecturers
Prof. Dr. Ivan Dokmanić ⟨ivan.dokmanic@unibas.ch⟩

Tutors
Felicitas Haag ⟨felicitas.haag@unibas.ch⟩
Alexandra Spitzer ⟨alexandra.spitzer@unibas.ch⟩
Cheng Shi ⟨cheng.shi@unibas.ch⟩
Vinith Kishore ⟨vinith.kishore@unibas.ch⟩

Problem set 1 [Model Solutions]

Summary

There are two parts in this problem set: math and coding. The math part contains some problems in linear algebra and probability, where you will use the material from the lecture notes. The coding part will ask you to build basic functions. Then you will have a chance to classify random signals and hand-written digits and text documents.

Math

Exercise 1 (Curses and blessings of dimensionality $\star$)

1. Let $C_D = [-1, 1]^D$, be a D-dimensional hypercube with side length 2, and $B_D = \{\boldsymbol{x} \mid \|\boldsymbol{x}\| \leq 1\}$ an inscribed D-dimensional ball with radius 1. Compute the ratio of their volumes,

$$r_D = \frac{\mathrm{Vol}(B_D)}{\mathrm{Vol}(C_D)}.$$

How much is $\lim_{D\to\infty} r_D$? Try to interpret this result in words. Where is most of the volume ("mass") of the hypercube located?

2. Let $B_D(r)$ be a D-dimensional ball of radius r, $B_D(r) = \{\boldsymbol{x} \mid \|\boldsymbol{x}\| \leq r\}$ and let $0 < \epsilon < 1$ (think of ϵ as being a small number). Compute

$$\varrho_D(\epsilon) = \frac{\mathrm{Vol}(B_D(1-\epsilon))}{\mathrm{Vol}(B_D(1))}.$$

How much is $\varrho_{500}(0.01)$? Again, try to interpret the result in words. Where is most of the volume of the sphere located?

1. To compute the ratio r_D, we first need to know how to compute the volume of D-dimensional hypercube with sidelength 2: $\mathrm{Vol}(C_D)$
For this, we consider a small (informal) example:

$$\begin{aligned}
&\text{volume of a square } (D=2) \text{ with side length } 2: && \mathrm{Vol}(C_2) = 2\cdot 2\\
&\text{volume of a cube } (D=3) \text{ with side length } 2: && \mathrm{Vol}(C_3) = 2\cdot 2\cdot 2\\
&\text{volume of a 4-cube } (D=4) \text{ with side length } 2: && \mathrm{Vol}(C_4) = 2\cdot 2\cdot 2\cdot 2\\
&\dots\\
&\text{volume of a } D\text{-dimensional hypercube with side length } 2: && \mathrm{Vol}(C_D) = 2^D
\end{aligned}$$

Now that we know $\mathrm{Vol}(C_D) = 2^D$, we can move on to the formula for $\mathrm{Vol}(B_D)$:
$\mathrm{Vol}(B_D(R)) = \frac{\pi^{D/2}R^D}{\Gamma(\frac{D}{2}+1)}$. Since in our problem, we have $R = 1$, and we know that $R^D =$

$1^D = 1$ we get $\mathrm{Vol}(B_D(1)) = \frac{\pi^{D/2}}{\Gamma(\frac{D}{2}+1)}$ Finally, we can take the ratio:

$$r_D = \frac{\mathrm{Vol}(B_D)}{\mathrm{Vol}(C_D)} = \frac{\pi^{D/2}}{\Gamma\big(\frac{D}{2}+1\big)\ 2^D}.$$

This ratio corresponds to the proportion of the D-dimensional hypercube's volume that is occupied by the inscribed D-dimensional ball.
How much is $\lim_{D\to\infty} r_D$? Using Stirling's approximation in the form

$$\Gamma(n+1) \sim \sqrt{2\pi n}\left(\tfrac{n}{e}\right)^n \quad \text{with } n = \tfrac{D}{2},$$

we get

$$\Gamma\Big(\tfrac{D}{2}+1\Big) \sim \sqrt{\pi D}\left(\tfrac{D}{2e}\right)^{D/2}.$$

Hence

$$r_D \sim \frac{\pi^{D/2}}{\sqrt{\pi D}\left(\frac{D}{2e}\right)^{D/2} 2^D} = \frac{1}{\sqrt{\pi D}}\left(\frac{\pi e}{2D}\right)^{D/2}.$$

Therefore $r_D \to 0$ super-exponentially fast as $D \to \infty$.

$$\lim_{D\to\infty} r_D = 0$$

Interpretation: In higher dimensions (larger D), a larger proportion of the hypercube's volume lies outside the inscribed ball. Differently said: As D increases, the volume of the inscribed ball B_D becomes an increasingly smaller fraction of the volume of the hypercube. When $D \to \infty$, the ball occupies none of the hypercube's volume.
This is counterintuitive! In high-dimensional spaces, most of the "mass" or volume of the hypercube is concentrated near the faces/edges rather than near its center. Our intuition from $2D$ and $3D$ does not apply to higher dimensions.

2. Using the ball-volume formula,

$$\mathrm{Vol}(B_D(1-\varepsilon)) = \frac{\pi^{D/2}}{\Gamma\big(\frac{D}{2}+1\big)}(1-\varepsilon)^D, \qquad \mathrm{Vol}(B_D(1)) = \frac{\pi^{D/2}}{\Gamma\big(\frac{D}{2}+1\big)}.$$

Therefore

$$\varrho_D(\varepsilon) = \frac{\mathrm{Vol}(B_D(1-\epsilon))}{\mathrm{Vol}(B_D(1))} = (1-\varepsilon)^D.$$

This ratio ϱ_D corresponds to the proportion of the D-dimensional unit ball's volume that is occupied by the slightly smaller ball.
Interpretation: As D increases, $(1-\varepsilon)^D$ decreases rapidly since $0 < \varepsilon < 1$. This means, that even a small reduction in the radius (by ε) leads to a significant reduction in the ratio ϱ_D. This is again the curse of dimensionality! Small changes in high-dimensional spaces can lead to large effects on volume.
Numerical value. For $D = 500$ and $\varepsilon = 0.01$,

$$\varrho_{500}(0.01) = (1-0.01)^{500} = (0.99)^{500} \approx 0.007.$$

Interpretation: When you reduce radius of a 500-dimensional sphere by just 1% ($1 \to 0.99$), the volume of the slightly smaller sphere is only 0.7% of the volume of the original sphere.
→ In high- dimensional spaces, most of the volume of the sphere is concentrated near its surface.

Exercise 2 (Distances and the CLT ⋆⋆⋆)

We want to compute the distribution of distances between random points in a D-dimensional cube. This is not easy if we want to do it precisely, but for large D we can get a good approximation using the central limit theorem. Let $\boldsymbol{X} = [X_1, \dots, X_D]^T$ and $\boldsymbol{Y} = [Y_1, \dots, Y_D]^T$ be two points picked independently uniformly at random in $[0,1]^D$. Thus, the coordinates $X_1, \dots, X_D$ and $Y_1, \dots, Y_D$ are independent and identically distributed (iid) random variables, each following a Uniform($[0,1]$) distribution. We will consider the squared distance

$$\|\boldsymbol{X} - \boldsymbol{Y}\|^2 = \sum_{d=1}^{D} (X_d - Y_d)^2.$$

1. What is the probability density function of $Z_d = X_d - Y_d$? Compute it and make a plot.
2. Compute the mean μ and the variance σ^2 of Z_d^2.
3. Use the Central Limit Theorem (CLT) to approximate the distribution of $\|\boldsymbol{X} - \boldsymbol{Y}\|^2$ by a normal distribution. *Hint*: You might want to multiply both sides by $\sqrt{D}$ in the usual statement of the CLT.
4. The most interesting statement (and the one compatible with the figures shown in class) is relative to the maximum distance in the D-dimensional cube. What is the squared length of the longest diagonal in this hypercube?
5. Denoting this squared length by ℓ_D^2, what are the mean and the variance of $\dfrac{\|\boldsymbol{X} - \boldsymbol{Y}\|^2}{\ell_D^2}$?

1. X_d and Y_d are iid random variables with uniform distribution.

$$X_d \text{ and } Y_d \text{ iid}, \qquad X_d \sim \text{Unif}(0,1), \qquad Y_d \sim \text{Unif}(0,1)$$

$$\text{goal: pdf of } \; Z_d = X_d - Y_d$$

$$f_{X_d}(x) = \begin{cases} 1, & 0 \le x \le 1, \\ 0, & \text{otherwise} \end{cases} \qquad f_{Y_d}(y) = \begin{cases} 1, & 0 \le y \le 1, \\ 0, & \text{otherwise} \end{cases}$$

$$f_{-Y_d}(y) = \begin{cases} 1, & -1 \le y \le 0, \\ 0, & \text{otherwise} \end{cases}$$

Convolution formula for two iid random variables $W = U + V$:

$$f_W(w) = \int_{-\infty}^{\infty} f_U(u)\, f_V(w-u)\, du$$

In our case: $\;U = X_d,\; V = -Y_d,\; W = Z_d$

$$\Rightarrow \quad f_{Z_d}(z) = \int_{-\infty}^{\infty} f_{X_d}(x)\, f_{-Y_d}(z-x)\, dx \qquad \{\text{find regions in which } f_{X_d} \text{ and } f_{-Y_d} \text{ are both } 1\}$$

Since $X, Y \in [0,1]$,

$$\min Z = 0 - 1 = -1 \quad (\text{at } X=0,\, Y=1), \qquad \max Z = 1 - 0 = 1 \quad (\text{at } X=1,\, Y=0).$$

Hence $Z \in [-1,1]$. For the convolution $f_Z(z) = \int f_X(x)\, f_{-Y}(z-x)\, dx$, the integrand is 1 only if $x \in [0,1]$ and $z - x \in [-1,0]$. We need to look at two cases:

Case 1: $\quad -1 \le z < 0$

$$\text{For } f_{-Y_d}(y) = f_{-Y_d}(z-x) = 1, \text{ we need } y = z - x \in [-1,0]$$

$$\Rightarrow -1 \leq z - x \leq 0 \Leftrightarrow x \leq z + 1$$

and

$$x \geq 0 \text{ (since uniform)}$$

$$\Rightarrow \quad \int_0^{z+1} 1 \cdot 1 \, dx = [x]_0^{z+1} = z + 1$$

Case 2: $0 \leq z \leq 1$

$$\text{For } f_{-Y_d}(y) = f_{-Y_d}(z - x) = 1, \text{ we need } y = z - x \in [-1, 0]$$

$$\Rightarrow -1 \leq z - x \leq 0 \Leftrightarrow x \geq z$$

and

$$x \leq 1 \text{ (since uniform)}$$

$$\Rightarrow \quad \int_z^1 1 \cdot 1 \, dx = [x]_z^1 = 1 - z$$

$\Rightarrow$ The pdf of Z_d is

$$f_{Z_d}(z) = \begin{cases} 1 + z, & -1 \leq z < 0, \\ 1 - z, & 0 \leq z \leq 1, \\ 0, & \text{otherwise.} \end{cases}$$

2. General formula to compute the mean of a random variable Z_d:

$$\mu_Z = \mathbb{E}[Z] = \int_{-\infty}^{\infty} z \, f_Z(z) \, dz \qquad \text{where } f_Z(z) \text{ is probability density function.}$$

$$\Rightarrow \mu_{Z_d^2} = \mathbb{E}[Z_d^2] = \int_{-\infty}^{\infty} z_d^2 \, f_{Z_d}(z_d) \, dz_d \qquad \text{with pdf}(z_d) \text{ from part 1 of this problem}$$

$$\Rightarrow \quad \mu_{Z_d^2} = \mathbb{E}[Z_d^2] = \int_{-1}^{0} z_d^2 (1 + z_d) \, dz_d \; + \; \int_0^1 z_d^2 (1 - z_d) \, dz_d$$

Multiply out

$$= \int_{-1}^{0} z_d^2 \, dz_d \; + \; \int_{-1}^{0} z_d^3 \, dz_d \; + \; \int_0^1 z_d^2 \, dz_d \; - \; \int_0^1 z_d^3 \, dz_d$$

Find primitive functions

$$= \left[\tfrac{1}{3} z_d^3\right]_{-1}^{0} \; + \; \left[\tfrac{1}{4} z_d^4\right]_{-1}^{0} \; + \; \left[\tfrac{1}{3} z_d^3\right]_0^1 \; - \; \left[\tfrac{1}{4} z_d^4\right]_0^1$$

Just compute

$$= \tfrac{1}{3} \; - \; \tfrac{1}{4} \; + \; \tfrac{1}{3} \; - \; \tfrac{1}{4} = \tfrac{2}{3} \; - \; \tfrac{2}{4} = \tfrac{1}{6} \quad \Rightarrow \quad \mu_{Z_d^2} = \tfrac{1}{6}$$

Calculate variance $\sigma_{Z_d^2}^2$ via:

$\sigma_{Z_d^2}^2 = \mathbb{E}\left[(Z_d^2)^2\right] - \left(\mathbb{E}[Z_d^2]\right)^2$ We already know $\mathbb{E}[Z_d^2] = \mu_{Z_d^2} = \frac{1}{6} \Rightarrow \left(\mathbb{E}[Z_d^2]\right)^2 = \left(\frac{1}{6}\right)^2 = \frac{1}{36}$.

We need to calculate $\mathbb{E}[Z_d^4]$:

$$\mathbb{E}[Z_d^4] = \int_{-1}^{0} z^4 (1 + z) \, dz \; + \; \int_0^1 z^4 (1 - z) \, dz = \tfrac{1}{30} \; + \; \tfrac{1}{30} = \tfrac{2}{30} = \tfrac{1}{15}.$$

$$\Rightarrow \quad \sigma_{Z_d^2}^2 = \tfrac{1}{15} - \tfrac{1}{36} = \tfrac{7}{180} \; .$$

3. Use CLT to approximate the distribution of $\|\mathbf{X}-\mathbf{Y}\|^2$ by a normal distribution.

$$\|\mathbf{X}-\mathbf{Y}\|^2 \;=\; \sum_{d=1}^{D}(X_d-Y_d)^2 \;=\; \sum_{d=1}^{D} Z_d^2 := S_D \qquad \textit{This is a sum of iid random variables.}$$

CLT: as $D \to \infty$, the sum of these iid random variables will be normally distributed. Let's calculate the mean and variance of the corresponding normal distribution. *Note: we know $\mu_{Z_d^2}$ and $\sigma^2_{Z_d^2}$ from the second part of this problem.*

$$\Rightarrow \quad \mu_{S_D} = \mathbb{E}[\|\mathbf{X}-\mathbf{Y}\|^2] = \mathbb{E}\left[\sum_{d=1}^{D} Z_d^2\right] = \mathbb{E}[Z_1^2 + \cdots + Z_D^2]$$

$$\overset{\text{independence of } Z_d\text{'s}}{=} \quad \mathbb{E}[Z_1^2] + \cdots + \mathbb{E}[Z_D^2]$$

$$\overset{\text{identical distribution of } Z_d\text{'s}}{=} \quad D \cdot \mathbb{E}[Z_d^2] = \; D \cdot \tfrac{1}{6} \quad \Rightarrow \quad \boxed{\mu_{S_D} = \frac{D}{6}}$$

The same ideas apply to the variance:

$$\Rightarrow \quad \sigma^2_{S_D} = \mathrm{Var}(\|\mathbf{X}-\mathbf{Y}\|^2) = \mathrm{Var}\left(\sum_{d=1}^{D} Z_d^2\right) = \mathrm{Var}(Z_1^2 + \cdots + Z_D^2)$$

$$\overset{\text{independence of } Z_d\text{'s}}{=} \mathrm{Var}(Z_1^2) + \cdots + \mathrm{Var}(Z_D^2)$$

$$\overset{\text{identical distribution of } Z_d\text{'s}}{=} D \cdot \mathrm{Var}(Z_d^2) = D \cdot \tfrac{7}{180} \quad \Rightarrow \quad \boxed{\sigma^2_{S_D} = \frac{7D}{180}}$$

$\Rightarrow$ Distribution of $\|\mathbf{X}-\mathbf{Y}\|^2$ can be approximated by a normal distribution for $D \to \infty$ with

$$\boxed{\|\mathbf{X}-\mathbf{Y}\|^2 \;\approx\; \mathcal{N}\left(\frac{D}{6}, \frac{7D}{180}\right).}$$

We can also express this in the standardized form of the CLT:

$$\frac{\|\mathbf{X}-\mathbf{Y}\|^2 - \mu_{S_D}}{\sqrt{\sigma^2_{S_D}}} = \frac{\|\mathbf{X}-\mathbf{Y}\|^2 - \frac{D}{6}}{\sqrt{\frac{7D}{180}}} \quad \xrightarrow{d} \quad \mathcal{N}(0,1).$$

In the above equation, the d over the arrow means convergence in distribution, i.e. that the distribution of this standardized sum converges to a normal distribution for $D \to \infty$.

4. What is squared length of the longest diagonal in this hypercube? (side length is 1.) It follows from the generalized Pythagorean theorem, that:

- For 1D cube ($\widehat{=}$ line): longest diagonal $= \sqrt{1^1} = 1$
- 2D cube ($\widehat{=}$ square): longest diagonal $= \sqrt{1^1 + 1^1} = \sqrt{2}$
- 3D cube ($\widehat{=}$ cube): longest diagonal $= \sqrt{\sqrt{2}^2 + 1^2} = \sqrt{3}$

$$\vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots$$

- D dimensions: longest diagonal $= \sqrt{D}$

$\Rightarrow$ **Squared length of longest diagonal:** $(\sqrt{D})^2 = \boxed{D}.$

5. Squared length of longest diagonal in hypercube: ℓ_D^2. What are μ and σ^2 of $\dfrac{\|\mathbf{X}-\mathbf{Y}\|^2}{\ell_D^2}$?

Mean:

$$\mu = \mathbb{E}\left[\frac{\|\mathbf{X}-\mathbf{Y}\|^2}{\ell_D^2}\right] = \mathbb{E}\left[\frac{\|\mathbf{X}-\mathbf{Y}\|^2}{D}\right] = \frac{1}{D}\,\mathbb{E}\left[\|\mathbf{X}-\mathbf{Y}\|^2\right] \quad \text{(we know this from Q. 3)}$$

$$= \frac{1}{D}\cdot\frac{D}{6} = \boxed{\frac{1}{6}} \quad \Rightarrow \quad \boxed{\mu = \tfrac{1}{6}}$$

Variance:

$$\sigma^2 = \mathrm{Var}\left[\frac{\|\mathbf{X}-\mathbf{Y}\|^2}{\ell_D^2}\right] = \mathrm{Var}\left[\frac{\|\mathbf{X}-\mathbf{Y}\|^2}{D}\right]$$

$$\mathrm{Var}(aX) = a^2\mathrm{Var}(X) \ \Rightarrow = \frac{1}{D^2}\,\mathrm{Var}\left[\|\mathbf{X}-\mathbf{Y}\|^2\right] = \frac{1}{D^2}\cdot\frac{7D}{180} \ \Rightarrow \ \boxed{\sigma^2 = \frac{7}{180D}}$$

Interpretation. As $D \to \infty$, the variance tends to 0, so these normalized squared distances become almost constant, and the corresponding (non-squared) distances concentrate around $\sqrt{1/6}$. In practical terms, in high dimensions most random pairs of points lie at nearly the same normalized distance, so the geometric contrast between "near" and "far" largely disappears. This illustrates the concentration of measure phenomenon.

Exercise 3 (conditional probability, and optimal classification ⋆)

From a panel of 60 patients from the hospital, we are given the following survey table counting the number of patients smoking and/or having cancer.

	not smoke ($X = 0$)	smoke ($X = 1$)
not cancer ($Y = 0$)	40	10
cancer ($Y = 1$)	7	3

Let $Y \in \{0, 1\}$ indicate cancer (1 if the patient has cancer, 0 otherwise) and $X \in \{0, 1\}$ indicate smoking (1 if the patient smokes, 0 otherwise). Throughout, treat probabilities as empirical proportions from the table ($\hat{P} = \text{count}/60$).

1. Compute the empirical joint probabilities $\hat{P}(X = 1, Y = 1)$ and $\hat{P}(X = 0, Y = 0)$ from the table.

2. Does the data provide evidence of an (in)dependence between X and Y? Justify quantitatively.

3. Compute the conditional probabilities $\hat{P}(X = 1 \mid Y = 1)$ and $\hat{P}(X = 0 \mid Y = 1)$.

4. Verify the identity $\hat{P}(X = 1 \mid Y = 1) = \dfrac{\hat{P}(X = 1, Y = 1)}{\hat{P}(Y = 1)}$.

5. Derive a Bayes optimal classifier for cancer $\hat{Y}(x)$. Making an error in the prediction can be rather dramatic, depending on the situation. The following table gives the cost of each prediction depending on the true state of the variable Y. Use this cost table to build the Bayes optimal classifier.

	predict not cancer ($\hat{Y} = 0$)	predict cancer ($\hat{Y} = 1$)
not cancer ($Y = 0$)	0	500
cancer ($Y = 1$)	10,000	−100,000

1. By the definition of conditional probability distributions

$$\begin{aligned} P(X=x, Y=y) &= P(Y=y|X=x)\cdot P(X=x) \\ &= P(X=x|Y=y)\cdot P(Y=y) \end{aligned}$$

we have:

$$\hat{P}(X=1,\, Y=1) = \hat{P}(X=1 \mid Y=1)\cdot \hat{P}(Y=1)$$

$$= \frac{3}{10}\cdot\frac{10}{60} = \frac{3}{60} = \boxed{\frac{1}{20}}$$

$$\hat{P}(X=0,\, Y=0) = \hat{P}(X=0 \mid Y=0)\cdot \hat{P}(Y=0)$$

$$= \frac{40}{50}\cdot\frac{50}{60} = \frac{40}{60} = \frac{20}{30} = \boxed{\frac{2}{3}}$$

True vs. empirical probabilities (why is it sometimes P and sometimes $\hat{P}$?) All theoretical identities we use in this exercise (conditional probabilities, Bayes' rule, optimal decision rules,..) are stated for the *true* population quantities $P(\cdot)$. In practice, we usually don't know them. With a finite sample (the 60 patients), we *estimate* them by empirical frequencies:

$$\hat{P}(\cdot) = \frac{\#(\cdot)}{60}, \qquad \hat{P}(X=x \mid Y=y) = \frac{\#\{X=x, Y=y\}}{\#\{Y=y\}}.$$

Thus, formulas remain the same, but with hats when we plug in counts. As sample size increases, $\hat{P} \to P$ (law of large numbers), so the plug-in rules approach the theoretical ones.

2. Two random variables are independent, if following is true:

$$P(X=x \mid Y=y) = P(X=x) \qquad \forall \text{ values of } X, Y$$

$$\begin{aligned} \hat{P}(X=0) &= \frac{47}{60}, & \hat{P}(X=1) &= \frac{13}{60} \\ \hat{P}(Y=0) &= \frac{50}{60}, & \hat{P}(Y=1) &= \frac{10}{60} \\ \hat{P}(X=0 \mid Y=0) &= \frac{40}{50}, & \hat{P}(X=0 \mid Y=1) &= \frac{7}{10} \\ \hat{P}(X=1 \mid Y=0) &= \frac{10}{50}, & \hat{P}(X=1 \mid Y=1) &= \frac{3}{10} \end{aligned}$$

$$\Rightarrow\ \hat{P}(X=x \mid Y=y) \neq \hat{P}(X=x) \ \ \forall \text{ values of } X \text{ and } Y$$

$$\Rightarrow \quad \boxed{\text{The data does provide evidence that } X, Y \text{ are dependent.}}$$

3. We can directly read this off from the table.

$$\hat{P}(X=1 \mid Y=1) = \frac{3}{10}, \qquad \hat{P}(X=0 \mid Y=1) = \frac{7}{10}$$

4. We can directly read off $\hat{P}(X=1 \mid Y=1) = \frac{3}{10}$ from the table. But we can also calculate it via the given relation:

$$\hat{P}(X=1 \mid Y=1) = \frac{\hat{P}(X=1, Y=1)}{\hat{P}(Y=1)} = \frac{\frac{3}{60}}{\frac{10}{60}} = \frac{3}{10} \quad \Rightarrow \quad \text{verified.}$$

5. The Bayes optimal classification is given via:

$$\hat{Y}(x) = \mathbb{1}\left\{\frac{\hat{P}(X = x \mid Y = 1)}{\hat{P}(X = x \mid Y = 0)} \geq \frac{\hat{P}(Y = 0) \cdot \big(\text{loss}(\hat{Y} = 1, Y = 0) - \text{loss}(\hat{Y} = 0, Y = 0)\big)}{\hat{P}(Y = 1) \cdot \big(\text{loss}(\hat{Y} = 0, Y = 1) - \text{loss}(\hat{Y} = 1, Y = 1)\big)}\right.$$

$$= \mathbb{1}\left\{\frac{\hat{P}(X = x \mid Y = 1)}{\hat{P}(X = x \mid Y = 0)} \geq \frac{\frac{50}{60}}{\frac{10}{60}} \cdot \frac{500}{10,000 - (-100,000)}\right.$$

$$= \mathbb{1}\left\{\frac{\hat{P}(X = x \mid Y = 1)}{\hat{P}(X = x \mid Y = 0)} \geq 0.0227\right.$$

If $X = 0$, then

$$\hat{Y}(X = 0) = \mathbb{1}\left\{\frac{\hat{P}(X = 0 \mid Y = 1)}{\hat{P}(X = 0 \mid Y = 0)} \geq 0.0227\right.$$

$$= \mathbb{1}\left\{\frac{\frac{7}{10}}{\frac{40}{50}} = 0.875 \geq 0.0227\right.$$

$$\Rightarrow 0.875 \geq 0.0227 \text{ is true } \Rightarrow \hat{Y}(X = 0) = 1$$

If $X = 1$, then

$$\hat{Y}(X = 1) = \mathbb{1}\left\{\frac{\hat{P}(X = 1 \mid Y = 1)}{\hat{P}(X = 1 \mid Y = 0)} \geq 0.0227\right.$$

$$= \mathbb{1}\left\{\frac{\frac{3}{10}}{\frac{10}{50}} = 1.5 \geq 0.0227\right.$$

$$\Rightarrow 1.5 \geq 0.0227 \text{ is true } \Rightarrow \hat{Y}(X = 1) = 1$$

$\Rightarrow$ **Bayes optimal predictor:**

$$\boxed{\hat{Y}(x) = \begin{cases} 1 & \text{for } X = 0 \\ 1 & \text{for } X = 1 \end{cases}}$$

Exercise 4 (Distance metrics and KNN classifier $\star$)

You are given a training set in $\mathbb{R}^2$ with binary labels:

- Class 0: $a_1 = (3, 0)$
- Class 1: $b_1 = (2, 2)$

Consider a test instance located at $x = (0, 0)$. We want to classify this instance using a K-Nearest Neighbors (KNN) classifier with the following assumption: the classifier is unweighted, meaning that each neighbor has an equal vote. Use the Manhattan distance ℓ_1 and the Euclidean distance ℓ_2.

1. Compute $d_1(x, a_1)$, $d_1(x, b_1)$, $d_2(x, a_1)$, $d_2(x, b_1)$.
2. For $k = 1$, determine the predicted class under the Manhattan distance ℓ_1 and under the Euclidean distance ℓ_2.
3. Briefly explain why changing the metric can change the 1-NN decision.

Reminder: For two points (x_1, y_1) and (x_2, y_2):

- The Manhattan distance is given by $\ell_1 = d((x_1, y_1), (x_2, y_2)) = |x_1 - x_2| + |y_1 - y_2|$

- The Euclidean distance is given by $\ell_2 = d((x_1, y_1), (x_2, y_2)) = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$

1. We use the provided formulas to compute the distances

$$\begin{aligned} d_1(x, a_1) &= |0-3| + |0-0| = 3, \\ d_1(x, b_1) &= |0-2| + |0-2| = 4, \\ d_2(x, a_1) &= \sqrt{(0-3)^2 + (0-0)^2} = \sqrt{9} = 3, \\ d_2(x, b_1) &= \sqrt{(0-2)^2 + (0-2)^2} = \sqrt{8} = 2\sqrt{2} \approx 2.828. \end{aligned}$$

2. 1-NN prediction
   - With ℓ_1: $d_1(x, a_1) = 3 < 4 = d_1(x, b_1) \Rightarrow$ nearest neighbor is a_1 (Class 0). **Predicted class: 0.**
   - With ℓ_2: $d_2(x, b_1) = 2\sqrt{2} < 3 = d_2(x, a_1) \Rightarrow$ nearest neighbor is b_1 (Class 1). **Predicted class: 1.**

3. Different metrics induce different notions of "closeness." The rankings of distances to candidate neighbors can change between ℓ_1 and ℓ_2. Here, x is Manhattan-closer to $(3, 0)$ but Euclidean-closer to $(2, 2)$, so the 1-NN decision flips.

Coding [You have the option to use Gradescope to check the correctness of your code]

Exercise 5 (K-nearest neighbour classifier ⋆)

The K-Nearest Neighbour (K-NN) classifier assigns a class to a test point based on the majority class among its K nearest neighbors in the training set. Your task is to implement a simple K-NN classifier by hand. We have provided a skeleton script `knn.py`.

We use the ℓ_2 norm to define distances between points, and consequently, the neighbors. The script `knn.py` contains several functions that need to be completed to create a viable working classifier:

1. `train()`: takes the training data and the corresponding labels as input. Fortunately, you only need to store this data in the class, as the K-NN classifier uses the data without any additional processing.

2. `predict()` and `predict_single_data()`: the former takes the entire dataset as input, while the latter takes a single data point and predicts its class label.

3. `compute_accuracy()`: first predicts the labels for the data and then computes the accuracy using the true labels. The accuracy can be computed as follows:

$$\text{accuracy} = \frac{1}{N}\sum_{j=1}^{N} \mathbb{1}_{y_j = y_j^{\text{predict}}},$$

where N is the total number of data points. y_j is the true label of the j^{th} data. y_j^{predict} is the corresponding predicted label.

Exercise 6 (Document classification ⋆⋆⋆)

Text classification is a key application of machine learning. It classifies documents into predefined topics. Since most machine learning algorithms operate on numerical data (vector of real numbers), we must represent documents by such “feature” vectors before performing classification. This should be done in such a way that the set of vectors corresponding to the set of documents has some sort of “linear” structure so as to make the application of standard algorithms meaningful. (Though we do not worry about it here.) The most common feature representation is a Bag of Words (BoW): it represents a document with a vectors containing the frequencies of a predefined set of words.

Bag of Words (BoW) In the Bag of Words (BoW) approach we treat predefined words as different dimensions or coordinates in the vector space, with the number of occurrences of each word serving as the value along that dimension. To apply BoW, we first generate a vocabulary of words from the dataset and then convert sentences into vectors based on word frequencies. For example, consider the following sentences:

1. ”The quick brown fox.”

2. ”The cat chased the mouse.”

3. ”Jumps over the lazy dog.”

4. ”The dog barks loudly”

The vocabulary derived from these sentences is:

$V =$ [”barks”,”brown”,”cat”,”chased”,”dog”, ”fox”,”jumps” ,”lazy”,”loudly”,”mouse”,”over”,”quick”,”the”]

Note that in this example, we consider words with different capitalization as the same. Depending on the application, vocabulary construction may vary. Once we have established the vocabulary, we can represent the sentences as vectors:

1. ”The quick brown fox” - $[0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1]$
2. ”The cat chased the” - $[0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 2]$
3. ”Jumps over the lazy dog” - $[0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 1]$
4. ”The dog barks loudly” - $[1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1]$

Now that we have a vector representation, we can employ various machine learning methods. Remember: while BoW is effective, there exist more sophisticated and more effective representations.

Your task is to build a K-NN classifier that takes research abstracts as input and predicts the domain to which they belong. You are provided with two files: `WebofScience_train.csv` (train data) and `WebofScience_test.csv` (test data), which contains data-sets with research abstracts, keywords, domain and sub domain. This dataset is preprocessed from a subset of the Web of Science [1] dataset. We break the task down as follows:

1. Implement the following functions in the `document_classification.py` file:
   (a) `extract_labels_and_text()`: Extract labels from the ’Domain’ column and input data from the ’Abstract’ column of the pandas DataFrame. No further processing is required here.

   Note: pandas is a powerful python library which is used for many exploratory data analysis tasks; well worth getting acquainted with if you’re not.

   (b) `generate_labels()`: Given a list of domain names, find all unique domain names and store them in the `self.class_labels` variable as a sorted list.

   (c) `preprocess_labels()`: Convert the domain names and class labels into an array where classes are represented as integers from 0 to #classes $-$ 1.

   (d) `preprocess()`: Convert text into a list of words by:
      i. Removing special characters and punctuation.
      ii. Removing words with a length of 1.
      iii. Converting all words to lowercase.

   (e) `bag_words()`: Generate the vocabulary from the training data and convert the training data into a matrix of features and store it in `self.X_train`.

   (f) `transform()`: Given abstracts, output their corresponding BoW representations in matrix form.

2. Use your implementation from `knn.py` to experiment with hyperparameters and, if possible, subsets of the training data. Report the accuracies achieved and recommend the best hyper-parameter settings and training data subsets that enhance accuracy on the test data. Provide the following in your report and submit it along with the math part of the problem set:
   (a) The plot of accuracy vs # neighbours
   (b) Hyperparameters (i.e. number of neighbours and subset of the training dataset) which worked best for you

3. Tf-idf is another popular representation for converting sentences into feature vectors. Instead of using Bag of Words (BoW) as in the previous question, use the tf-idf representation from the sklearn library. Report the accuracy versus the number of neighbors in a plot, along with the hyperparameters you used. Do you observe any improvements compared to BoW?

   Note:

   (a) tf-idf documentation is present here

Hints:

1. Use the `.split()` to convert the string to words.
2. You can use the helper function `remove_special_characters()` provided, to remove special characters and punctuations' present in the string.
3. To verify your vocabulary output, compare it with sklearn's "CountVectorizer" function.

Exercise 7 (Multi-class perceptron ⋆)

In this exercise, you will implement a multi-class version of the perceptron introduced in the lecture. Extending the two-class perceptron to a K-class perceptron involves the use of a weight matrix W with dimensions $K \times D$, where K represents the number of classes, and D denotes the dimensionality of the input data vector x. Each row in the weight matrix W corresponds to a unique class, while the columns represent the individual features or dimensions of the input data. The prediction for the class in the k-class perceptron is defined as:

$$y^{\text{predict}} = \arg\max_i \langle W[i,:], x\rangle$$

The training process is obtained as follows:

Algorithm 1 Training the perceptron

```
Input: , Dataset =$\{x_1, x_2, \dots x_N\}$, labels = $\{y_1, y_2, \dots, y_N\}$,$W_{init}$, $n_{\text{iter}}$
Initialize: $n \leftarrow 0, W \leftarrow W_{init}$
while $n \leq n_{\text{iter}}$ do
    for $j \leftarrow 1$ to N do
        $y_j^{\text{predict}} = \arg\max_i \langle W[i,:], x_j\rangle$
    end for
    for $j \leftarrow 1$ to N do
        if $y_j^{\text{predict}} = y_j$ then
            $W \leftarrow W$
        else if $y_{\text{predict}} \neq y$ then
            $W[y,:] \leftarrow W[y,:] + x_j$
            $W[y_j^{\text{predict}},:] \leftarrow W[y_j^{\text{predict}},:] - x_j$
        end if
    end for
    $n \leftarrow n + 1$
end while
```

Please note that the predicted values are integers in the multi-class setting. Since each value represents a specific class in the data, it is considered a categorical value. There is no numerical relationship between Class 1 and Class 2. Therefore, it is often meaningful to represent them as a K-dimensional vector of 1s and 0s. For the data point x, if the label is k, we represent it as follows:

$$\tilde{y} = [\underbrace{0, \dots, 0}_{k-1 \text{ times}}, 1, 0, \dots]^{\mathrm{T}}$$

This conversion of labels from categorical representation to a vector representation is called one-hot encoding.

Using these definitions, your goal is to train a multi-class perceptron on the popular MNIST dataset, which contains 28×28 pixel images of handwritten digits from 0 to 9. You have been provided with a skeleton code for the task and the MNIST dataset in the data folder. You need to complete the following functions:

1. `one_hot_encode()`: This function converts integer class labels into one-hot encoded vectors.

2. `predict()`: This function predicts the label using the weight matrix and the dataset and outputs the one-hot encoded version of the labels.

   Hint: Using a loop to process each point is too slow. Python and numpy can handle such operations simultaneously.

3. `predict_score()`: It calculates the matrix product of the dataset and the weight matrix W. The score for a single data x is defined as Wx.

   Hint: Same as the previous hint.

4. `loss()`: Computes the total number of times the prediction differs from the true label. This is defined as:

$$\text{loss} = \sum_{j=1}^{N} \mathbb{1}_{y_j \neq y_j^{predict}}$$

5. `train()`: This function updates the weight matrix for a specified number of iterations and computes the loss at each iteration. Use the provided update function to update the weight matrix and utilize the completed functions to compute the loss.

Exercise 8 (Classifying random patterns ⋆)

In this exercise, you will explore the behavior and performance of the previously implemented perceptron on a random dataset for binary classification. The goal is to vary the dimension of the input data D while keeping the number of samples N fixed, and analyze the perceptron's performance.

You have been provided with a skeleton code in `random_classifier.py`, with all the necessary libraries preloaded. Your task is outlined as follows:

1. Set the number of samples to $N = 100$ and the input dimension to $D = 10$. Generate N samples of Gaussian random vectors, denoted as $\mathbf{x} \sim \mathcal{N}(0, I_D)$. Generate the output class values (y) for each vector by sampling a Bernoulli random variable $y \sim \text{Ber}(p = 0.5)$. This will serve as our training set. Similarly, create another set of N samples of x vectors and their corresponding labels for the test set.

2. Train the perceptron using the training set for a fixed number of iterations. Compute the accuracy on both the training and test set. The accuracy is defined as follows:

$$\text{accuracy} = \frac{1}{N} \sum_{i=1}^{N} \mathbb{1}_{y = y_{\text{predict}}}$$

3. Maintain the same setup and resample the test and train data while varying the input dimension from $D = 10$ to $D = 200$ in increments of 10. Plot the train and test accuracies as functions of D. Provide an explanation for the observed trends in the plot.

   Hint: To obtain a smooth curve, repeat the experiment and average the results.

References

[1] Kamran Kowsari, Donald E. Brown, Mojtaba Heidarysafa, Kiana Jafari Meimandi, Matthew S. Gerber, Laura E. Barnes, *Hdltex: Hierarchical deep learning for text classification*, In *2017 16th IEEE international conference on machine learning and applications (ICMLA)*, pages 364–371, Year 2017, Organization IEEE