50 Pen and Paper Exercises on Machine Learning (with Solutions)

Tanujit Chakraborty (Blog)

2024

Topics to be covered (Exercises and Solutions):

1. Mathematics for Machine learning - Optimization and Probability
2. Linear Models
3. Decision Trees and Nearest Neighbors
4. Model Selection and Cross-Validation
5. Kernel machines (Support Vector Machines)
6. Probabilistic Machine Learning
7. EM Algorithm
8. Generative models
9. Unsupervised Learning
10. Neural Networks

Acknowledgment: These problems and their solutions are largely based on some extraordinary books on Machine Learning and several outstanding university courses available online, with some minor additions by the author. A non-exhaustive list is given below. Happy Learning!

References (Books):

1. Deisenroth, Marc Peter, A. Aldo Faisal, and Cheng Soon Ong. Mathematics for machine learning. Cambridge University Press, 2020.
2. Bishop, Christopher M., and Nasser M. Nasrabadi. Pattern recognition and machine learning. Vol. 4. No. 4. New York: Springer, 2006.
3. Murphy, Kevin P. Machine learning: a probabilistic perspective. MIT Press, 2012.

References (Online Resources):

1. CS229: Machine Learning Course at Stanford University.
2. CS771A: Introduction to Machine Learning Course at IIT Kanpur.
3. 10-701 Machine Learning Course at CMU, USA.

Machine Learning - Pen and Paper Exercises (with Solutions)

Problem 1. (Placement Woes)

Tanujit wants to set up a shop at a point in $\mathbb{R}^2$. The consumer density is such that if he opens shop at a point $(x, y) \in \mathbb{R}^2$, his daily income will be $2x+4y$. However, since his supplier is situated at $(2, -1)$, he will incur a daily cost of $(x-2)^2+(y+1)^2$ for transporting goods from his supplier to his shop. Where should Tanujit open his shop to get the maximum daily profit, and what is that maximum profit value?

Solution:

We cast the above problem as an optimization problem. Let $\mathbf{a} = (2, 4), \mathbf{b} = (2, -1) \in \mathbb{R}^2$. We wish to minimize $f(\mathbf{x}) = \|\mathbf{x} - \mathbf{b}\|_2^2 - \mathbf{a}^\top \mathbf{x}$ (minimizing loss is the same as maximizing profit). We have $\nabla f(\mathbf{x}) = 2(\mathbf{x} - \mathbf{b}) - \mathbf{a}$ and $\nabla^2 f(\mathbf{x}) = 2I > 0$, i.e., f is a convex differentiable function. Using first-order optimality to get the global minimum gives us $\mathbf{x}^{\text{opt}} = \mathbf{b} + \mathbf{a}/2 = (3, 1)$ and the profit earned by opening up a shop at $\mathbf{x}^{\text{opt}}$ is $-f(\mathbf{x}^{\text{opt}}) = 5$ units.

Problem 2. (Bayes Rule)

Suppose that 1% of Olympic athletes use performance-enhancing drugs and that a particular drug test has a 5% false positive rate and a 2% false negative rate.

(a) Athlete A tests positive for drug use. What is the probability that Athlete A is not using drugs?

(b) Athlete B tests negative for drug use. What is the probability that Athlete B is using drugs?

Solution:

Some calculations about joint probabilities.

$$\begin{aligned}
P(D = 0) &= 1 - 0.01 = 0.99 \\
P(D = 1) &= 0.01 \\
P(D = 0, T = 0) &= 0.99 \times (1 - 0.05) = 0.9405 \\
P(D = 0, T = 1) &= 0.99 \times 0.05 = 0.0495 \\
P(D = 1, T = 0) &= 0.01 \times 0.02 = 0.0002 \\
P(D = 1, T = 1) &= 0.01 \times (1 - 0.02) = 0.0098
\end{aligned}$$

(a) $P(D = 0 \mid T = 1) = \frac{P(D=0,T=1)}{P(D=0,T=1)+P(D=1,T=1)} = \frac{0.0495}{0.0495+0.0098} \approx 0.8347$

(b) $P(D = 1 \mid T = 0) = \frac{P(D=1,T=0)}{P(D=0,T=0)+P(D=1,T=0)} = \frac{0.0002}{0.9405+0.0002} \approx 0.0002$

Problem 3. (Derivatives)

Let $\mathbf{a} \in \mathbb{R}^d$ be a constant vector and $b \in \mathbb{R}$ be a constant scalar. For $\mathbf{x} \in \mathbb{R}^d$, let us define the function $f(\mathbf{x}) = \ln\left(1 + \exp\left(-b \cdot \mathbf{a}^\top \mathbf{x}\right)\right)$ (where ln is the natural logarithm). Find $\nabla f(\mathbf{x})$ and briefly show all major steps in your derivation.

Solution:

We have $f(t) = \ln(t)$ where $t(s) = 1 + \exp(-s)$ where $s(\mathbf{x}) = b \cdot \mathbf{a}^\top \mathbf{x}$. We have $f'(t) = \frac{1}{t'}$, $t'(s) = -\exp(-s)$, and $\nabla s(\mathbf{x}) = b \cdot \mathbf{a}$. Thus, applying the chain rule gives us

$$\nabla f(\mathbf{x}) = \frac{-b \cdot \exp\left(-b \cdot \mathbf{a}^\top \mathbf{x}\right)}{1 + \exp\left(-b \cdot \mathbf{a}^\top \mathbf{x}\right)} \cdot \mathbf{a}.$$

Problem 4. (Optimization Problem)

Consider the optimization problem below.

$$\begin{array}{ll} \min\limits_{x\in\mathbb{R}} & \frac{1}{2}x^2 \\ \text{s.t.} & x \geq 2 \end{array}$$

Write down the Lagrangian of the problem. Then write down the dual of the problem. Eliminate the primal variable and write down the simplified dual problem with the primal variable eliminated. Then solve the dual problem and write down the optimal values of the primal and dual variables that you have obtained.

Solution:

If we introduce a dual variable α, the Lagrangian is

$$\mathcal{L}(x,\alpha) = \frac{1}{2}x^2 + \alpha(2-x).$$

The dual problem is

$$\max_{\alpha\geq 0}\min_{x} \frac{1}{2}x^2 + \alpha(2-x).$$

Using first-order optimality gives us $x - \alpha = 0$ or in other words, $x = \alpha$. Substituting this into the dual problem gives us $\max_{\alpha\geq 0} 2\alpha - \frac{1}{2}\alpha^2$ as the simplified dual problem.

If suppose we ignore the constraint $\alpha \geq 0$ for a moment, then applying the first order optimality condition tells us that the function $2\alpha - \frac{1}{2}\alpha^2$ achieves its maximum where $2 - \alpha = 0$ or in other words at $\alpha = 2$. However, this point does satisfy $\alpha \geq 0$ which means $\alpha = 2$ is the maximum for $2\alpha - \frac{1}{2}\alpha^2$ even with the constraint $\alpha \geq 0$ and thus, the solution of the dual problem. Using $x = \alpha$ and strong duality tells us that $x = 2$ is the solution to the primal problem.

Problem 5. (Sub-differential)

Let $\mathbf{a} \in \mathbb{R}^d$ be a constant vector and $b \in \mathbb{R}$ be a constant scalar. Let $f : \mathbb{R}^d \to \mathbb{R}$ be a function defined as $f(\mathbf{x}) = \max\left\{\mathbf{a}^\top\mathbf{x} + b, 0\right\}$. Find an expression for subdifferential of f at any $\mathbf{x} \in \mathbb{R}^d$. Show all the main steps of your derivation briefly in the space provided.

Solution:

We have $f(\mathbf{x}) = \max\{g(\mathbf{x}), h(\mathbf{x})\}$ where $g(\mathbf{x}) = \mathbf{a}^\top\mathbf{x} + b$ and $h(\mathbf{x}) = 0$. Note that both $g(\mathbf{x})$ and $h(\mathbf{x})$ are differentiable functions. Thus, applying the max rule for subgradients gives us

$$\begin{cases} \mathbf{a} & \text{if } \mathbf{a}^\top\mathbf{x} + b > 0 \\ \mathbf{0} & \text{if } \mathbf{a}^\top\mathbf{x} + b < 0 \\ \lambda\cdot\mathbf{a} & \text{if } \mathbf{a}^\top\mathbf{x} + b = 0 \end{cases}$$

where $\lambda \in [0,1]$ and $\mathbf{0} \in \mathbb{R}^d$ is the d-dimensional all zeros vector. Note that in the second case, the subgradient is the zero vector, not the number/ scalar zero.

Problem 6. (Chain Rule)

Let $\mathbf{x} = [1,-1]^\top, \mathbf{y} = [-1,1]^\top \in \mathbb{R}^2$. Define the function $f : \mathbb{R}^2 \to \mathbb{R}^2$ as $f(\mathbf{z}) = z_1\cdot\mathbf{x} + z_2\cdot\mathbf{y}$ for any $\mathbf{z} = [z_1, z_2] \in \mathbb{R}^2$. Define another function $g : \mathbb{R} \to \mathbb{R}^2$ as $g(r) = \left[r, r^2\right]$ where $r \in \mathbb{R}$. Let $h : \mathbb{R} \to \mathbb{R}^2$ be defined as $h(r) = f(g(r))$. Derive a general expression for $\frac{dh}{dr}$ using the chain rule giving major steps of derivation and then evaluate $\frac{dh}{dr}$ at $r = 3$.

Solution:

Let $A = \left[\mathbf{x}^\top, \mathbf{y}^\top\right] = \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \in \mathbb{R}^2$ so that we have $f(\mathbf{z}) = A\mathbf{z}$ which gives us $J^f = \nabla f = A$ (to see that the answer is indeed A and not $A^\top$, think of a hypothetical example where we have $\mathbf{z} = [z_1, z_2, z_3] \in \mathbb{R}^3$ and $f(\mathbf{z}) = z_1\cdot\mathbf{x} + z_2\cdot\mathbf{y} + z_3\cdot\mathbf{p}$ for $\mathbf{p} = [1,1]^\top$).

Next, we calculate $J^g = [1, 2r]^\top$ (notice that this is a column vector since this is not a gradient of a real-valued function but rather the Jacobian of a vector-valued function). Thus, we have

$$\frac{dh}{dr} = J^h = J^f \cdot J^g = \left[\begin{array}{cc} 1 & -1 \\ -1 & 1 \end{array}\right] \left[\begin{array}{c} 1 \\ 2r \end{array}\right] = \left[\begin{array}{c} 1-2r \\ 2r-1 \end{array}\right].$$

Note the dimensionality of J^h which fits our convention of Jacobians being of dimensionality o/p dims ×i/p dims since $h : \mathbb{R} \to \mathbb{R}^2$. At $r = 3$ we have $J^h = \left[\begin{array}{c} -5 \\ 5 \end{array}\right]$.

Problem 7. (Squared Hinge Loss)

Let $\mathbf{a} \in \mathbb{R}^d$ and $b \in \mathbb{R}$ be constants. For $\mathbf{x} \in \mathbb{R}^d$, define the function $f(\mathbf{x}) = \left(\left[1 - b \cdot \mathbf{a}^\top \mathbf{x}\right]_+\right)^2$. Find $\nabla f(\mathbf{x})$.

Solution:

The function $g(t) = ([1-t]_+)^2$ is actually differentiable. The function is clearly differentiable for $t \neq 1$ with

$$g'(t) = \begin{cases} 0 & t > 1 \\ 2(t-1) & t < 1 \end{cases}$$

However, it is clear that $\lim_{t \to 1^+} g'(t) = \lim_{t \to 1^-} g'(t) = 0$ which shows that $g(t)$ is differentiable. Now, we simply have $f(\mathbf{x}) = g\left(b \cdot \mathbf{a}^\top \mathbf{x}\right)$ and thus, on applying chain rule we get

$$\nabla f(\mathbf{x}) = g'\left(b \cdot \mathbf{a}^\top \mathbf{x}\right) \cdot b \cdot \mathbf{a} = \begin{cases} \mathbf{0} & b \cdot \mathbf{a}^\top \mathbf{x} \geq 1 \\ 2b\left(b \cdot \mathbf{a}^\top \mathbf{x} - 1\right) \cdot \mathbf{a} & b \cdot \mathbf{a}^\top \mathbf{x} < 1 \end{cases}$$

Note that the $\mathbf{0}$ above is the zero vector and not the zero real number.

Problem 8. (Model Exfiltration)

Doraemon has a secret function $f : \mathbb{R}^4 \to \mathbb{R}$. We know that f thresholds one of the 4 coordinates of the input at a value, i.e., for all $\mathbf{x} \in \mathbb{R}^4, f(\mathbf{x}) = \mathbf{x}_j - c$ where $j \in \{1, 2, 3, 4\}$ and $c \in \mathbb{R}$. E.g. if $j = 2, c = 1.5$, then for $\mathbf{x} = (1, 5, 2, 2), f(\mathbf{x}) = 5 - 1.5 = 3.5$. I want to steal Doraemon's model and find out what value of j, c is Doraemon using. I can send Doraemon any number of inputs $\mathbf{x}^1, \mathbf{x}^2, \ldots \in \mathbb{R}^4$ and Doraemon will return $f\left(\mathbf{x}^1\right), f\left(\mathbf{x}^2\right), \ldots \in \mathbb{R}$ back to me. Design an algorithm below that asks Doraemon function values on one or more 4D vectors and uses Doraemon's responses to find the value of j and c being used. You must give explicit descriptions of the 4D vectors you are querying Doraemon (e.g., you may say that you wish to query only three vectors $(1, -0.5, 1, -3), (8, -0.1, 1, 5)$ and $(9, 1, 5, 6)$). The fewer vectors you query Doraemon to correctly find j, c, the better your answer.

Solution:

Step 1: Query $\mathbf{x}^1 = (0, 0, 0, 0)$. Note that $f\left(\mathbf{x}^1\right) = -c$ so $c = -f\left(\mathbf{x}^1\right)$.

Step 2: Query $\mathbf{x}^2 = (1, 2, 3, 4)$. Note that $f\left(\mathbf{x}^2\right) = \mathbf{x}_j^2 - c = \mathbf{x}_j^2 + f\left(\mathbf{x}^1\right)$ so $\mathbf{x}_j^2 = f\left(\mathbf{x}^2\right) - f\left(\mathbf{x}^1\right)$. Since we have deliberately set $\mathbf{x}_i^2 = i$ for all $i \in \{1, 2, 3, 4\}$, we get $j = f\left(\mathbf{x}^2\right) - f\left(\mathbf{x}^1\right)$.

Step 3: Output $c = -f\left(\mathbf{x}^1\right),\ j = f\left(\mathbf{x}^2\right) - f\left(\mathbf{x}^1\right)$.

Note that the only thing required in step 2 is for $\mathbf{x}^2$ to have different values for all its four coordinates. This helps us figure out which coordinate is being thresholded. Setting $\mathbf{x}_i^2 = i$ merely makes the algorithm more aesthetically pleasing.

Also note that the above scheme uses two queries to figure out the model completely. This is, in general, optimal, i.e., there is no way an algorithm can figure out both j, c using just one query. Suppose an algorithm makes a single query on a vector $\mathbf{x} = (p, q, r, s)$ to see why. Upon receiving the response, such an algorithm will still be unsure whether $j = 1$ and $c = p - f(\mathbf{x})$ or whether $j = 2$ and $c = q - f(\mathbf{x})$ or $j = 3$ and $c = r - f(\mathbf{x})$ or $j = 4$ and $c = s - f(\mathbf{x})$. The algorithm will necessarily need to make another query to zero in on one of the above 4 cases.

Problem 9. (Normal Random Variable)

Let X be a Normal random variable with mean $\mu \in \mathbb{R}$ and variance $\tau^2 > 0$, i.e. $X \sim \mathcal{N}\left(\mu, \tau^2\right)$. Recall that the probability density of X is given by

$$f_X(x) = \frac{1}{\sqrt{2\pi}\tau} e^{-(x-\mu)^2/2\tau^2}, \quad -\infty < x < \infty.$$

Furthermore, the random variable Y given $X = x$ is normally distributed with mean x and variance σ^2, i.e.,

$$Y|\, X = x \sim \mathcal{N}\left(x, \sigma^2\right).$$

(a) Derive the marginal distribution of Y.

(b) Use Bayes' theorem to derive the conditional distribution of X given $Y = y$.

[Hint: For both tasks derive the density up to a constant factor and use this to identify the distribution.]

Solution:

Before starting calculations, it is good to mention that one can easily compute the following integral for $a > 0$ by creating complete squares:

$$\begin{aligned}\int_{\mathbb{R}} e^{-\left(ax^2+2bx+c\right)} dx &= \int_{\mathbb{R}} \exp\left(-a\left[\left(x+\frac{b}{a}\right)^2 - \frac{b^2-ac}{a^2}\right]\right) dx \\ &= \exp\left(\frac{b^2-ac}{a}\right) \cdot \int_{\mathbb{R}} \exp\left(-\frac{1}{2}\frac{\left(x+\frac{b}{a}\right)^2}{1/2a}\right) dx \\ &= \exp\left(\frac{b^2-ac}{a}\right)\sqrt{\pi/a}\end{aligned}$$

As a prelude to both (a) and (b) we consider the joint density function $f_{X,Y}(x,y)$ of X and Y

$$f_{X,Y}(x,y) = f_{Y|X}(y \mid X = x) f_X(x) = \frac{1}{2\pi\sigma\tau} \exp\left(-\frac{1}{2}\underbrace{\left[\frac{(x-\mu)^2}{\tau^2} + \frac{(y-x)^2}{\sigma^2}\right]}_{\text{(A)}}\right).$$

For brevity, let us define

$$\begin{aligned}a &:= \frac{\sigma^2+\tau^2}{2\sigma^2\tau^2}, \\ b &:= -\frac{\sigma^2\mu+\tau^2 y}{2\sigma^2\tau^2}, \\ c &:= \frac{\sigma^2\mu^2+\tau^2 y^2}{2\sigma^2\tau^2}.\end{aligned}$$

Using simple algebraic operations, we obtain that (A) $= ax^2 + 2bx + c$.

(a) The marginal density of Y is given by

$$f_Y(y) = \int_{\mathbb{R}} f_{X,Y}(x,y) dx = \int_{\mathbb{R}} f_{Y|X}(y \mid X = x) f_X(x) dx.$$

Using the formula discussed at the beginning of the solution, we can compute this integral by just putting in the values of a, b, and c:

$$\begin{aligned}f_Y(y) &= \int_{\mathbb{R}} f_{X,Y}(x,y) dx \\ &= \int_{\mathbb{R}} \frac{1}{2\pi\sigma\tau} e^{-\left(ax^2+2bx+c\right)} dx \\ &= \frac{1}{2\pi\sigma\tau} \exp\left(\frac{b^2-ac}{a}\right)\sqrt{\pi/a} \\ &\propto \exp\left(\frac{b^2-ac}{a}\right) \qquad (a \text{ does not depend on } y)\end{aligned}$$

Now we try to write $\left(b^2-ac\right)/a$ as a complete square:

$$\begin{aligned}
\frac{b^2-ac}{a} &= \frac{1}{a}\left\{\left(\frac{\sigma^2\mu+\tau^2 y}{2\sigma^2\tau^2}\right)^2 - \frac{\left(\sigma^2+\tau^2\right)\left(\sigma^2\mu^2+\tau^2y^2\right)}{\left(2\sigma^2\tau^2\right)^2}\right\} \\
&= -\frac{1}{a}\cdot\frac{1}{\left(2\sigma^2\tau^2\right)^2}\cdot\left(\sigma^2\tau^2y^2-2\tau^2\sigma^2\mu y+\sigma^2\tau^2\mu^2\right) \\
&= -\frac{1}{a}\cdot\frac{\sigma^2\tau^2}{\left(2\sigma^2\tau^2\right)^2}\cdot\left((y-\mu)^2+\cdots\right) \\
&= -\frac{1}{2}\frac{1}{\left(\sigma^2+\tau^2\right)}\cdot\left((y-\mu)^2+\cdots\right)
\end{aligned}$$

Putting everything together yields

$$f_Y(y) \propto \exp\left[-\frac{1}{2}\frac{(y-\mu)^2}{\left(\sigma^2+\tau^2\right)}\right]$$

meaning that Y has a Gaussian distribution with mean μ and variance $\sigma^2+\tau^2$.

(b) The conditional density of X given $Y=y$ is proportional to the joint density function, i.e.

$$f_{X|Y}(x \mid Y=y) = \frac{f_{X,Y}(x,y)}{f_Y(y)} \propto f_{X,Y}(x,y).$$

By the discussion at the beginning of the solution, $f_{X,Y}(x,y) \propto \exp\left(-\left(ax^2+2bx+c\right)\right)$. Since c does not depend on x (and y is considered as fixed/given), we can say:

$$f_{X|Y}(x \mid Y=y) \propto \exp\left(-\frac{1}{2}\frac{\left(x+\frac{b}{a}\right)^2}{1/2a}\right)$$

So the mean would be $-b/a$, and the variance would be $1/2a$. Concretely:

$$\text{mean} = -\frac{b}{a} = \frac{\sigma^2\mu+\tau^2y}{\sigma^2+\tau^2} = \frac{\sigma^2}{\sigma^2+\tau^2}\mu + \frac{\tau^2}{\sigma^2+\tau^2}y$$

Note that the mean is a convex combination of μ and the observation y. Also

$$\text{variance } = \frac{1}{2a} = \frac{\sigma^2\tau^2}{\sigma^2+\tau^2}.$$

Problem 10. (Bivariate Normal Random Variable)

Let X be a bivariate Normal random variable (taking on values in $\mathbb{R}^2$) with mean $\mu=(1,1)$ and covariance matrix $\Sigma=\left(\begin{array}{cc} 3 & 1 \\ 1 & 2 \end{array}\right)$. The density of X is then given by

$$f_X(\mathbf{x}) = \frac{1}{\sqrt{(2\pi)^2\det(\Sigma)}}\exp\left(-\frac{1}{2}(\mathbf{x}-\mu)^T\Sigma^{-1}(\mathbf{x}-\mu)\right).$$

Find the conditional distribution of $Y=X_1+X_2$ given $Z=X_1-X_2=0$.

Solution:

We present two approaches for solving this exercise:

Approach 1. Note that $Z=0$ implies $X_1=X_2$. Furthermore by the definition of Y, we have $X_1=X_2=Y/2$ given $Z=0$. Hence the marginal density of Y given $Z=0$ is proportional to

$$f_{Y|Z}(y \mid Z=0) = \frac{f_{Y,Z}(y,0)}{f_Z(0)} \propto f_{Y,Z}(y,0) \propto f_X\left[\left(\begin{array}{c} y/2 \\ y/2 \end{array}\right)\right].$$

We then have

$$\begin{aligned}
f_X\left[\begin{pmatrix} y/2 \\ y/2 \end{pmatrix}\right] &\propto \exp\left(-\frac{1}{2}\begin{pmatrix} \frac{y}{2}-1 \\ \frac{y}{2}-1 \end{pmatrix}^T \begin{pmatrix} 3 & 1 \\ 1 & 2 \end{pmatrix}^{-1} \begin{pmatrix} \frac{y}{2}-1 \\ \frac{y}{2}-1 \end{pmatrix}\right) \\
&= \exp\left(-\frac{1}{2}\begin{pmatrix} \frac{y}{2}-1 \\ \frac{y}{2}-1 \end{pmatrix}^T \frac{1}{5}\begin{pmatrix} 2 & -1 \\ -1 & 3 \end{pmatrix} \begin{pmatrix} \frac{y}{2}-1 \\ \frac{y}{2}-1 \end{pmatrix}\right) \\
&= \exp\left(-\frac{1}{2}\frac{(y-2)^2}{\frac{20}{3}}\right).
\end{aligned}$$

Clearly, the conditional distribution of Y given $Z = 0$ is hence Normal with mean 2 and variance $\frac{20}{3}$.

Approach 2. We define the random variable $\mathbf{R}$ as

$$\mathbf{R} = \begin{pmatrix} Y \\ Z \end{pmatrix} = \underbrace{\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}}_{=\mathbf{A}} \mathbf{X}.$$

By linearity of expectation, the mean $\mu_\mathbf{R}$ of $\mathbf{R}$ is

$$\mathbb{E}[\mathbf{R}] = \mathbf{A}\mathbb{E}[\mathbf{X}] = \mathbf{A}\mu = \begin{pmatrix} 2 \\ 0 \end{pmatrix}.$$

The covariance matrix $\mathbf{\Sigma_R}$ of $\mathbf{R}$ is given by

$$\begin{aligned}
\mathbf{\Sigma_R} &= \mathbb{E}\left[(\mathbf{R}-\mathbb{E}[\mathbf{R}])(\mathbf{R}-\mathbb{E}[\mathbf{R}])^T\right] = \mathbb{E}\left[\mathbf{A}(\mathbf{X}-\mathbb{E}[\mathbf{X}])(\mathbf{X}-\mathbb{E}[\mathbf{X}])^T\mathbf{A}^T\right] \\
&= \mathbf{A}\mathbb{E}\left[(\mathbf{X}-\mathbb{E}[\mathbf{X}])(\mathbf{X}-\mathbb{E}[\mathbf{X}])^T\right]\mathbf{A}^T = \mathbf{A\Sigma A}^T \\
&= \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}\begin{pmatrix} 3 & 1 \\ 1 & 2 \end{pmatrix}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \\
&= \begin{pmatrix} 4 & 3 \\ 2 & -1 \end{pmatrix}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \\
&= \begin{pmatrix} 7 & 1 \\ 1 & 3 \end{pmatrix}
\end{aligned}$$

Since $\mathbf{X}$ is multivariate Gaussian and $\mathbf{R}$ is an affine transformation of $\mathbf{X}, \mathbf{R}$ is a bivariate Normal random variable with mean $\mu_\mathbf{R}$ and covariance matrix $\mathbf{\Sigma_R}$.[1]

The conditional density of Y given $Z = 0$ is then given by

$$\begin{aligned}
f_{Y|Z}(y \mid Z=0) &= \frac{f_{Y,Z}(y,0)}{f_Z(0)} \propto f_{Y,Z}(y,0) \\
&\propto \exp\left(-\frac{1}{2}\begin{pmatrix} y-2 \\ 0 \end{pmatrix}^T \begin{pmatrix} 7 & 1 \\ 1 & 3 \end{pmatrix}^{-1} \begin{pmatrix} y-2 \\ 0 \end{pmatrix}\right) \\
&= \exp\left(-\frac{1}{2}\begin{pmatrix} y-2 \\ 0 \end{pmatrix}^T \frac{1}{20}\begin{pmatrix} 3 & -1 \\ -1 & 7 \end{pmatrix} \begin{pmatrix} y-2 \\ 0 \end{pmatrix}\right) \\
&= \exp\left(-\frac{1}{2}\frac{(y-2)^2}{\frac{20}{3}}\right).
\end{aligned}$$

Clearly, the conditional distribution of Y given $Z = 0$ is hence Normal with mean 2 and variance $\frac{20}{3}$.

[1] This result can be easily derived from the characteristic function of the multivariate Normal distribution. $\mathbf{R}$ is bivariate Normal if and only if for any $\mathbf{t} \in \mathbb{R}^2$.

$$\mathbb{E}\left[e^{i\mathbf{t}^T\mathbf{R}}\right] = e^{i\mathbf{t}^T\mu_\mathbf{R} - \mathbf{t}^T\mathbf{\Sigma_R}\mathbf{t}/2}.$$

This holds since the corresponding property holds for $\mathbf{X}$ with $\mathbf{s} = \mathbf{t}^T\mathbf{A}$, i.e.

$$\mathbb{E}\left[e^{i\mathbf{t}^T\mathbf{R}}\right] = \mathbb{E}\left[e^{i\mathbf{t}^T\mathbf{AX}}\right] = \mathbb{E}\left[e^{i\mathbf{s}^T\mathbf{X}}\right] = e^{i\mathbf{s}^T\mu - \mathbf{s}^T\mathbf{\Sigma s}/2} = e^{i\mathbf{t}^T\mathbf{A}\mu - \mathbf{t}^T\mathbf{A\Sigma A}^T\mathbf{t}/2} = e^{i\mathbf{t}^T\mu_\mathbf{R} - \mathbf{t}^T\mathbf{\Sigma_R}\mathbf{t}/2}.$$

Problem 11. (Linear Regression and Ridge Regression)

Let $D = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots (\mathbf{x}_n, y_n)\}$ where $\mathbf{x}_i \in \mathbb{R}^d$ and $y_i \in \mathbb{R}$ be the training data that you are given. As you have to predict a continuous variable, one of the simplest possible models is linear regression, i.e., to predict y as $\mathbf{w}^T\mathbf{x}$ for some parameter vector $\mathbf{w} \in \mathbb{R}^d$ [2]. We thus suggest minimizing the following loss

$$\underset{\mathbf{w}}{\operatorname{argmin}} \hat{R}(\mathbf{w}) = \underset{\mathbf{w}}{\operatorname{argmin}} \sum_{i=1}^{n} \left(y_i - \mathbf{w}^T\mathbf{x}_i\right)^2 . \tag{1}$$

Let us introduce the $n \times d$ matrix $\mathbf{X} \in \mathbb{R}^{n \times d}$ with the $\mathbf{x}_i$ as rows, and the vector $\mathbf{y} \in \mathbb{R}^n$ consisting of the scalars y_i. Then, Eqn. (1) can be equivalently re-written as

$$\underset{\mathbf{w}}{\operatorname{argmin}} \|\mathbf{X}\mathbf{w} - \mathbf{y}\|^2 .$$

We refer to any $\mathbf{w}^*$ that attains the above minimum as a solution to the problem.

(a) Show that if $\mathbf{X}^T\mathbf{X}$ is invertible, then there is a unique $\mathbf{w}^*$ that can be computed as $\mathbf{w}^* = \left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\mathbf{y}$.

(b) Show for $n < d$ that Eqn. (1) does not admit a unique solution. Intuitively explain why this is the case.

(c) Consider the case $n \geq d$. Under what assumptions on $\mathbf{X}$ does Eqn. (1) admit a unique solution $\mathbf{w}^*$? Give an example with $n = 3$ and $d = 2$ where these assumptions do not hold.

The ridge regression optimization problem with parameter $\lambda > 0$ is given by

$$\underset{\mathbf{w}}{\operatorname{argmin}} \hat{R}_{\text{Ridge}}(\mathbf{w}) = \underset{\mathbf{w}}{\operatorname{argmin}} \left[\sum_{i=1}^{n} \left(y_i - w^T\mathbf{x}_i\right)^2 + \lambda\mathbf{w}^T\mathbf{w}\right] . \tag{2}$$

(d) Show that $\hat{R}_{\text{Ridge}}(\mathbf{w})$ is convex with regards to $\mathbf{w}$. You can use the fact that a twice differentiable function is convex if and only if it's Hessian $\mathbf{H} \in \mathbb{R}^{d \times d}$ satisfies $\mathbf{w}^T\mathbf{H}\mathbf{w} \geq 0$ for all $\mathbf{w} \in \mathbb{R}^d$ (is positive semi-definite).

(e) Derive the closed form solution $\mathbf{w}^*_{\text{Ridge}} = \left(\mathbf{X}^T\mathbf{X} + \lambda I_d\right)^{-1}\mathbf{X}^T\mathbf{y}$ to Eqn. (2) where I_d denotes the identity matrix of size $d \times d$.

(f) Show that Eqn. (2) admits the unique solution $\mathbf{w}^*_{\text{Ridge}}$ for any matrix $\mathbf{X}$. Show that this even holds for the cases in (b) and (c) where Eqn. (1) does not admit a unique solution $\mathbf{w}^*$.

(g) What is the role of the term $\lambda\mathbf{w}^T\mathbf{w}$ in $\hat{R}_{\text{Ridge}}(\mathbf{w})$? What happens to $\mathbf{w}^*_{\text{Ridge}}$ as $\lambda \to 0$ and $\lambda \to \infty$?

Solution:

(a) Note that

$$\hat{R}(\mathbf{w}) = \|\mathbf{X}\mathbf{w} - \mathbf{y}\|^2 = (\mathbf{X}\mathbf{w} - \mathbf{y})^T(\mathbf{X}\mathbf{w} - \mathbf{y}) = \mathbf{w}^T\mathbf{X}^T\mathbf{X}\mathbf{w} - 2\mathbf{w}^T\mathbf{X}^T\mathbf{y} + \mathbf{y}^T\mathbf{y}.$$

The gradient of this function is equal to (see Lemma 1 in Supplementary Material)

$$\nabla\hat{R}(\mathbf{w}) = 2\mathbf{X}^T\mathbf{X}\mathbf{w} - 2\mathbf{X}^T\mathbf{y}.$$

Because $\hat{R}(\mathbf{w})$ is convex (formally proven in (d)), its optima are exactly those points that have a zero gradient, i.e. those $\mathbf{w}^*$ that satisfy $\mathbf{X}^T\mathbf{X}\mathbf{w}^* = \mathbf{X}^T\mathbf{y}$. Under the given assumption, the unique minimizer is indeed equal to $\mathbf{w}^* = \left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\mathbf{y}$.

(b) Consider the singular value decomposition $\mathbf{X} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$ where $\mathbf{U}$ is a unitary $n \times n$ matrix, $\mathbf{V}$ is a unitary $d \times d$ matrix and $\mathbf{\Sigma}$ is a diagonal $n \times d$ matrix with the singular values of $\mathbf{X}$ on the diagonal. We then have

$$\underset{\mathbf{w}}{\operatorname{argmin}} \hat{R}(\mathbf{w}) = \underset{\mathbf{w}}{\operatorname{argmin}} \left[\mathbf{w}^T\mathbf{V}\mathbf{\Sigma}^2\mathbf{V}^T\mathbf{w} - 2\mathbf{y}^T\mathbf{U}\mathbf{\Sigma}\mathbf{V}^T\mathbf{w}\right]$$

[2] Without loss of generality, we assume that both $\mathbf{x}_i$ and y_i are centered, i.e., they have zero empirical means. Hence we can neglect the otherwise necessary bias term b.

Since $\mathbf{V}$ is unitary (and hence it is a bijection), we may rotate $\mathbf{w}$ using $\mathbf{V}$ to $\mathbf{z} = \mathbf{V}^T\mathbf{w}$ and formulate the optimization problem in terms of $\mathbf{z}$, i.e.

$$\underset{\mathbf{z}}{\operatorname{argmin}} \left[\mathbf{z}^T\mathbf{\Sigma}^2\mathbf{z} - 2\mathbf{y}^T\mathbf{U}\mathbf{\Sigma}\mathbf{z}\right] = \underset{\mathbf{z}}{\operatorname{argmin}} \sum_{i=1}^{d} \left[z_i^2\sigma_i^2 - 2\left(\mathbf{U}^T\mathbf{y}\right)_i z_i\sigma_i\right]$$

where σ_i is the i entry in the diagonal of $\mathbf{\Sigma}$. Note that this problem decomposes into d independent optimization problems of the form

$$z_i = \underset{z}{\operatorname{argmin}} \left[z^2\sigma_i^2 - 2\left(\mathbf{U}^T\mathbf{y}\right)_i z\sigma_i\right]$$

for $i = 1, 2, \ldots, d$. Since each problem is quadratic with a positive coefficient and thus convex, we may obtain the solution by finding the root of the first derivative. For $i = 1, 2, \ldots d$ we require that z_i satisfies

$$z_i\sigma_i^2 - \left(\mathbf{U}^t\mathbf{y}\right)_i \sigma_i = 0.$$

For all $i = 1, 2, \ldots d$ such that $\sigma_i \neq 0$, the solution z_i is thus given by

$$z_i = \frac{(\mathbf{U}^t\mathbf{y})_i}{\sigma_i}.$$

For the case $n < d$, however, $\mathbf{X}$ has at most rank n as it is a $n \times d$ matrix and hence at most n of its singular values are nonzero. This means that there is at least one index j such that $\sigma_j = 0$ and hence any $z_j \in \mathbb{R}$ is a solution to the optimization problem. As a result the set of optimal solutions for $\mathbf{z}$ is a linear subspace of at least one dimension. By rotating this subspace back using $\mathbf{V}$, i.e., $\mathbf{w} = \mathbf{V}\mathbf{z}$, it is evident that the optimal solution to the optimization problem in terms of $\mathbf{w}$ is also a linear subspace of at least one dimension and that thus no unique solution exists. Furthermore, since $\mathbf{X}$ has at most rank $n, \mathbf{X}^T\mathbf{X}$ is not of full rank (see Lemma 2 of Supplementary Section). As a result $\left(\mathbf{X}^T\mathbf{X}\right)^{-1}$ does not exist and $\mathbf{w}^*$ is ill-defined.

The intuition behind these results is that the "linear system" $\mathbf{X}\mathbf{w} \approx \mathbf{y}$ is under-determined as there are fewer data points than parameters that we want to estimate.

(c) We showed in (b) that the optimization problem admits a unique solution only if all the singular values of $\mathbf{X}$ are nonzero. For $n \geq d$, this is the case if and only if $\mathbf{X}$ is of full rank, i.e., all the columns of $\mathbf{X}$ are linearly independent. As an example of a matrix not satisfying these assumptions, any matrix with linearly dependent suffices, e.g.

$$\mathbf{X}_{\text{degenerate}} = \begin{pmatrix} 1 & -2 \\ 0 & 0 \\ -2 & 4 \end{pmatrix}.$$

(d) Because convex functions are closed under addition, we will show that each term in the objective is convex, from which the claim will follow. Each data term $\left(y_i - \mathbf{w}^T\mathbf{x}_i\right)^2$ has a Hessian $\mathbf{x}_i\mathbf{x}_i^T$, which is positive semi-definite because for any $\mathbf{w} \in \mathbf{R}^d$ we have $\mathbf{w}^T\mathbf{x}_i\mathbf{x}_i^T\mathbf{w} = \left(\mathbf{x}_i^T\mathbf{w}_i\right)^2 \geq 0$ (note that $\mathbf{x}_i^T\mathbf{w} = \mathbf{w}^T\mathbf{x}_i$ are scalars). The regularizer $\lambda\mathbf{w}^T\mathbf{w}$ has the identity matrix λI_d as a Hessian, which is also positive semi-definite because for any $\mathbf{w} \in \mathbf{R}^d$ we have $\mathbf{w}^T\lambda I_d\mathbf{w} = \lambda\|\mathbf{w}\|^2 \geq 0$, and this completes the proof.

(e) The gradient of $\hat{R}_{\text{Ridge}}\left(\mathbf{w}\right)$ with respect to $\mathbf{w}$ is given by

$$\nabla\hat{R}_{\text{Ridge}}\left(\mathbf{w}\right) = 2\mathbf{X}^T(\mathbf{X}\mathbf{w} - \mathbf{y}) + 2\lambda\mathbf{w}.$$

Similar to (a), because $\hat{R}_{\text{Ridge}}\left(\mathbf{w}\right)$ is convex, we only have to find a point $\mathbf{w}^*_{\text{Ridge}}$ such that

$$\nabla\hat{R}_{\text{Ridge}}\left(\mathbf{w}^*_{\text{Ridge}}\right) = 2\mathbf{X}^T\left(\mathbf{X}\mathbf{w}^*_{\text{Ridge}} - \mathbf{y}\right) + 2\lambda\mathbf{w}^*_{\text{Ridge}} = 0.$$

This is equivalent to

$$\left(\mathbf{X}^T\mathbf{X} + \lambda I_d\right)\mathbf{w}^*_{\text{Ridge}} = \mathbf{X}^T\mathbf{y}$$

which implies the required result

$$\mathbf{w}^*_{\text{Ridge}} = \left(\mathbf{X}^T\mathbf{X} + \lambda I_d\right)^{-1}\mathbf{X}^T\mathbf{y}$$

(f) Note that $\mathbf{X}^T\mathbf{X}$ is a positive semi-definite matrix[3] since $\forall \mathbf{w} \in \mathbb{R}^d : \mathbf{w}^T\mathbf{X}^T\mathbf{X}\mathbf{w} = \sum_{i=1}^n \left[(\mathbf{X}\mathbf{w})_i\right]^2 \geq 0$, which implies that it has non-negative eigenvalues. But then, $\mathbf{X}^T\mathbf{X} + \lambda I_d$ has eigenvalues bounded from below by $\lambda > 0$, which means that it is invertible and thus the optimum is uniquely defined.

Note. Since $\mathbf{X}^T\mathbf{X}$ is symmetric, all of its eigenvalues are real, and it is clear that μ is an eigenvalue of $\mathbf{X}^T\mathbf{X}$ if and only if $\mu + \lambda$ is an eigenvalue of $\mathbf{X}^T\mathbf{X} + \lambda I$. Also note that if a linear function is injective, its kernel is $\{\mathbf{0}\}$, meaning it does not have a zero eigenvalue. The converse is also true.

(g) The term $\lambda \mathbf{w}^T\mathbf{w}$ "biases" the solution towards the origin, i.e., there is a quadratic penalty for solutions $\mathbf{w}$ that is far from the origin. The parameter λ determines the extend of this effect: As $\lambda \to 0$, $\hat{R}_{\text{Ridge}}\left(\mathbf{w}\right)$ converges to $\hat{R}(\mathbf{w})$. As a result the optimal solution $\mathbf{w}^*_{\text{Ridge}}$ approaches the solution of (1). As $\lambda \to \infty$, only the quadratic penalty $\mathbf{w}^T\mathbf{w}$ is relevant and $\mathbf{w}^*_{\text{Ridge}}$ hence approaches the null vector $(0, 0, \ldots, 0)$.

One can also pose this interesting question: Assume $n < d$ (as the situation discussed in (b)). Then w* for linear regression is not unique. Denote by $\mathbf{w}^*_\lambda$ the unique solution to the Ridge regression problem for $\lambda > 0$. Does the limit $\lim_{\lambda \to 0} \mathbf{w}^*_\lambda$ exist? If yes, because of the completeness of $\mathbb{R}^d$, the limit point should fall inside the space of solutions to linear regression problems. What is this solution?

Supplementary Material:

Lemma 1. *Let $A \in \mathbb{R}^{n\times n}$ be a real matrix and define $f(\mathbf{x}) = \mathbf{x}^\top A\mathbf{x}$ to be the quadratic form defined via A. Then we have $\nabla f(\mathbf{x}) = \left(A + A^\top\right)\mathbf{x}$. Moreover, if A is symmetric, then $\nabla f(\mathbf{x}) = 2A\mathbf{x}$.*

Proof. Let us compute the derivative of f at point $\mathbf{x}$. We know

$$f(\mathbf{x}+\mathbf{h}) - f(\mathbf{x}) = \mathbf{h}^\top A\mathbf{h} + \mathbf{h}^\top A\mathbf{x} + \mathbf{x}^\top A\mathbf{h} = \left(\mathbf{h}^\top A + \mathbf{x}^\top A^\top + \mathbf{x}^\top A\right)\mathbf{h}.$$

By taking the limit $\|\mathbf{h}\| \to 0$, the linear operator $\left(\mathbf{x}^\top A^\top + \mathbf{x}^\top A\right)$ would be the derivative. So the gradient would be

$$\nabla f(\mathbf{x}) = \left(A + A^\top\right)\mathbf{x}$$

□

Lemma 2. *Let $A \in \mathbb{R}^{m\times n}$ and $B \in \mathbb{R}^{n\times k}$ be two matrices. Then*

$$\operatorname{rank}(AB) \leq \operatorname{rank}(A)$$

Proof. If we denote the columns of B by $\mathbf{b}_1, \ldots, \mathbf{b}_k$, then we can write $AB = [A\mathbf{b}_1, \ldots, A\mathbf{b}_k]$. Now $A\mathbf{b}_i$ is a linear combination of columns of A, so the columns of AB are all linear combinations of columns of A. It follows that the subspace spanned by the columns of AB is included in the span of columns of A. Hence we will have the desired inequality.

□

Problem 12. (Multi-output Regression with Reduced Number of Parameters)

Consider the multi-output regression in which each output $\boldsymbol{y}_n \in \mathbb{R}^M$ in a real-valued vector rather than a scalar. Assuming a linear model, we can model the outputs as $\mathbf{Y} = \mathbf{X}\mathbf{W}$, where $\mathbf{X}$ is the $N \times D$ feature matrix and $\mathbf{Y}$ is $N \times M$ response matrix with row n being $\boldsymbol{y}_n^\top$ (note that each column of $\mathbf{Y}$ denotes one of the M responses), and $\mathbf{W}$ is the $D \times M$ weight matrix, with its M columns containing the M weight vectors $\boldsymbol{w}_1, \boldsymbol{w}_2, \ldots, \boldsymbol{w}_M$. Let's define a squared error loss function $\sum_{n=1}^N \sum_{m=1}^M \left(y_{nm} - \boldsymbol{w}_m^\top \boldsymbol{x}_n\right)^2$, which is just the usual squared error but summed over all the M outputs. Firstly, verify that this can also be written in a more compact notation as TRACE $\left[(\mathbf{Y} - \mathbf{X}\mathbf{W})^\top(\mathbf{Y} - \mathbf{X}\mathbf{W})\right]$.

Note: Here we will assume that the weight matrix W can be written as a product of two matrices, i.e., $\mathbf{W} = \mathbf{B}\mathbf{S}$ where $\mathbf{B}$ is $D \times K$ and $\mathbf{S}$ is $K \times M$ (assume $K < \min\{D, M\}$). Note that there is a benefit of modeling W this way since now we need to learn only $K \times (D + M)$ parameters as opposed to $D \times M$ parameters. If K is small, this can significantly reduce the number of parameters (in fact, reducing the effective number of parameters to be learned is another way of regularizing a machine learning model). Note (you can verify) that each w_m can be written as a linear combination of K columns of $\mathbf{B}$ in this formulation.

[3] An equivalent notion for a matrix A being positive semi-definite is that for all $\mathbf{x} \in \mathbb{R}^n$ we have $\mathbf{x}^\top A\mathbf{x} \geq 0$.

With the proposed representation of $\mathbf{W}$, the new objective will be TRACE $\left[(\mathbf{Y}-\mathbf{XBS})^{\top}(\mathbf{Y}-\mathbf{XBS})\right]$ and you need to learn both $\mathbf{B}$ and $\mathbf{S}$. While it is certainly possible to learn both (one way to do it is using a procedure called "alternating optimization"), for now, let's keep it simple and assume that the matrix B is known and only $\mathbf{S}$ needs to be estimated. So your problem is reduced to

$$\hat{\mathbf{S}} = \arg\min_{\mathbf{S}} \text{TRACE}\left[(\mathbf{Y}-\mathbf{XBS})^{\top}(\mathbf{Y}-\mathbf{XBS})\right]$$

Derive the expression for $\hat{\mathbf{S}}$. Briefly explain how the form of the solution is identical to the solution of standard multi-output regression but using a transformed version of the inputs $\mathbf{X}$.

Solution:

Consider the multi-output regression in which each output $y_n \in \mathbb{R}^m$ is a real-valued vector rather than a scalar. Assuming a linear model, we can model the outputs as $Y = XW$, where X is the usual input data matrix, Y is the response matrix of dimensions $N \times M, W$ is the weight matrix to be learned of dimensions $D \times M$. The least squares objective in this setting can be stated as

$$\mathcal{L}(W) = \sum_{n=1}^{N}\sum_{m=1}^{M}\left(y_n m - w_m^T x_n\right)^2$$

$$\therefore \mathcal{L}(W) = \sum_{m=1}^{M}\left(\sum_{n=1}^{N}\left(y_n m - w_m^T x_n\right)^2\right)$$

$$\therefore \mathcal{L}(W) = \sum_{m=1}^{M}\left(\sum_{n=1}^{N}\left(y_m - Xw_m\right)^T\left(y_m - Xw_m\right)\right)$$

$$\therefore \mathcal{L}(W) = TRACE\left((Y-XW)^T(Y-XW)\right)$$

Here, we will assume that the weight matrix W can be written as a product of two matrices, i.e., $W = BS$ where B is $D \times K$ and S is $K \times M$ (assume $K < \min\{D, M\}$). We further assume that B is known and we would like to estimate S. Thus, estimating S can be stated as an optimization problem as follows:

$$\hat{S} = \arg\min_{S} \text{TRACE}\left[(Y-XW)^T(Y-XW)\right].$$

The loss function can be stated as a function of S as follows:

$$\mathcal{L}(S) = \text{tr}\left[(Y-XBS)^T(Y-XBS)\right] = \text{tr}\left[Y^TY\right] - \text{tr}\left[(XBS)^TY\right] - \text{tr}\left[Y^TXBS\right] + \text{tr}\left[(XBS)^TXBS\right]$$

Now, we need to compute $\frac{\partial\mathcal{L}(S)}{\partial S}$ and equate it to 0 to find the optimal solution for S.

$$\frac{\partial\mathcal{L}(S)}{\partial S} = \frac{\partial\,\text{tr}\left(Y^TY\right)}{\partial S} - \frac{\partial\,\text{tr}\left[(XBS)^TY\right]}{\partial S} - \frac{\partial\,\text{tr}\left[\left(Y^TXB\right)S\right]}{\partial S} + \frac{\partial\,\text{tr}\left[(XBS)^TXBS\right]}{\partial S}$$

$$\therefore \frac{\partial\mathcal{L}(S)}{\partial S} = 0 - \frac{\partial\,\text{tr}\left[S^T\left(B^TX^TY\right)\right]}{\partial S} - \frac{\partial\,\text{tr}\left[\left(Y^TXB\right)S\right]}{\partial S} + \frac{\partial\,\text{tr}\left[S^T\left(B^TX^TXB\right)S\right]}{\partial S}$$

$$\therefore \frac{\partial\mathcal{L}(S)}{\partial S} = -B^TX^TY - \left(Y^TXB\right)^T + \left(B^TX^TXB\right)S + \left(B^TX^TXB\right)^TS$$

$$\therefore 0 = 2B^TX^TY + 2\left(B^TX^TXB\right)S$$

$$\therefore \left(B^TX^TXB\right)S = B^TX^TY$$

$$\therefore S = \left(B^TX^TXB\right)^{-1}B^TX^TY$$

$$\therefore S = \left((XB)^TXB\right)^{-1}(XB)^TY = \left(X^TX\right)^{-1}XY$$

where $\bar{X} = XB$ i.e., the input matrix transformed by the product with B. Thus, the form of the solution is identical to the solution of standard multi-output regression but uses a transformed version of the inputs X.

Problem 13. (Feature-Specific ℓ_2 Regularization)

The usual ℓ_2 regularized least squares regression objective can be written as $\mathcal{L}(\boldsymbol{w}) = \sum_{n=1}^{N}\left(y_n - \boldsymbol{w}^{\top}\boldsymbol{x}\right)^2 + \frac{\lambda}{2}\boldsymbol{w}^{\top}\boldsymbol{w}$. In this case, the extent of regularization is the same on all the features (and is controlled by λ). Propose an alternative that still uses ℓ_2 regularization on $\boldsymbol{w}$, but the extent of regularization is different for each entry w_d. Write down the objective function and derive the closed-form expression for the weight vector $\boldsymbol{w}$.

Solution:

The usual l_2 regularized least squares regression objective can be written as

$$\mathcal{L}(w) = \sum_{n=1}^{N} \left(y_n - w^T x_n\right)^2 + \frac{\lambda}{2} w^T w$$

Now, the first term can be rewritten in terms of the input data matrix and output label vector. Also, considering different hyperparameters for each feature, say λ_i for feature i, the objective function can be re-stated as

$$\mathcal{L}(w) = (y - Xw)^T (y - Xw) + \sum_{i=1}^{N} \frac{\lambda_i}{2} w_i^2$$

We need to compute $\frac{\partial \mathcal{L}(w)}{\partial w}$. We already know the derivative of the first term from the analysis of least squares regression. For the second term, w.r.t the feature i, we have

$$\frac{\partial \left(\sum_{i=1}^{N} \frac{\lambda_1}{2} w_i^2\right)}{\partial w_i} = 2 * (\lambda_i/2)\, w_i = \lambda_i w_i.$$

Thus, the gradient w.r.t w can be computed:

$$\frac{\partial \left(\sum_{i=1}^{N} \frac{\lambda_1}{2} w_i^2\right)}{\partial w} = \begin{bmatrix} \lambda_1 w_1 & \lambda_2 w_2 & \lambda_3 w_3 & \dots & \lambda_N w_N \end{bmatrix}^T$$

$$\therefore \frac{\partial \left(\sum_{i=1}^{N} \frac{\lambda_1}{2} w_i^2\right)}{\partial w} = \begin{bmatrix} \lambda_1 & 0 & 0 & \dots & 0 \\ 0 & \lambda_2 & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & \lambda_N \end{bmatrix} \begin{bmatrix} w_1 & w_2 & w_3 & \dots & w_N \end{bmatrix}^T = \text{diag}(\bar{\lambda}) w$$

where diag(v) is a diagonal matrix for vector v and $\bar{\lambda} = \begin{bmatrix} \lambda_1 & \lambda_2 & \lambda_3 & \dots & \lambda_N \end{bmatrix}$. Now, equating $\frac{\partial \mathcal{L}(w)}{\partial w} = 0$, we get

$$\begin{aligned} 0 &= X^T y - \left(X^T X + \text{diag}(\bar{\lambda})\right) w \\ \therefore \left(X^T X + \text{diag}(\bar{\lambda})\right) w &= X^T y \\ \therefore w &= \left(X^T X + \text{diag}(\bar{\lambda})\right)^{-1} X^T y \end{aligned}$$

Thus, we get the closed form solution as $\hat{w} = \left(X^T X + \text{diag}(\bar{\lambda})\right)^{-1} X^T y$.

Problem 14. (Linear Regression meets Nearest Neighbors)

(a) Show that for the un-regularized linear regression model, where the solution $\hat{\boldsymbol{w}} = \left(\mathbf{X}^\top \mathbf{X}\right)^{-1} \mathbf{X}^\top \boldsymbol{y}$, the prediction at a test input $\boldsymbol{x}_*$ can be written as a weighted sum of all the training responses, i.e.,

$$f\left(\boldsymbol{x}_*\right) = \sum_{n=1}^{N} w_n y_n.$$

Give the expression for the weights w_n 's in this case and briefly discuss (<50 words) in what way these weights are different from the weights in a weighted version of K nearest neighbors where each w_n typically is the inverse distance of $\boldsymbol{x}_*$ from the training input $\boldsymbol{x}_n$. Note: You do not need to give a very detailed expression for w_n (if it makes algebra messy) but must give a precise meaning as to what w_n depends on and how it differs from the weights in the weighted K nearest neighbors.

(b) An important notion for a classifier is that of consistency. A classification algorithm is said to be consistent if, whenever it has access to infinite amounts of training data, its error rate approaches the optimal error rate (a.k.a. Bayes optimal). Consider the noise-free setting (i.e., every training input is labeled correctly). Here, the Bayes optimal error rate is zero. Is the one-nearest-neighbor algorithm consistent in this setting? Briefly justify your answer.

Solution:

(a) The solution to the linear regression problem is given by $\hat{w} = \left(X^T X\right)^{-1} X^T y$. For a test example x_*, we can predict the output as follows:

$$f\left(x_*\right) = \hat{w}^T x_*$$
$$\therefore f\left(x_*\right) = \left(\left(X^T X\right)^{-1} X^T y\right)^T x_*$$
$$\therefore f\left(x_*\right) = y^T \left(X^T\right)^T \left[\left(X^T X\right)^{-1}\right]^T x_*$$
$$\therefore f\left(x_*\right) = y^T X \left[\left(X^T X\right)^T\right]^{-1} x_* = y^T X \left[X^T X\right]^{-1} x_*$$
$$\therefore f\left(x_*\right) = \left\langle X \left(X^T X\right)^{-1} x_*, y\right\rangle = \sum_{n=1}^{N} w_n y_n$$

where w_n is the n^{th} component of the term $X\left(X^T X\right)^{-1} x_*$. Let $A := X\left(X^T X\right)^{-1}$. Let us denote this weight vector corresponding to linear regression as w^{LR} and that of weighted kNN as w^{kNN}. Our goal is to find/interpret some sort of a relation between these two. As given, we know:

$$w_n^{kNN} = \frac{1}{d\left(x_n, x_*\right)}.$$

Now, we will try to interpret what w_n^{LR} means. Note that $w^k NN$ is a sort of similarity between the input data matrix X and x_*. Similarly, observe that $w^{LR} = X\left(X^T X\right)^{-1} x_*$ and thus we have

$$w_n^{LR} = x_n^T \left(X^T X\right)^{-1} x_*$$

Thus, w_n^{LR} is also a sort of similarity between x_n and x_* which is weighted by the matrix $\left(X^T X\right)^{-1}$ much like the Mahalanobois matrix.

(b) Claim: 1-nearest neighbor (1-NN) classification is a consistent classification method.
Justification: Suppose we consider the 1-nearest neighbor (1-NN) classification trained on infinite training examples. Note that k − NN has an error rate not worse than twice that of Bayes optimal classifier.

$$E_{k-NN} \leq 2E_{BO} \rightarrow 0 \Rightarrow E_{k-NN} \rightarrow 0$$

Note that the Bayes optimal error rate goes to 0 in a noise-free setting. Thus, the error rate for 1-NN also goes to zero. Another way to look at it is that since 1-NN considers distance from just one nearest sample point at test time, it is prone to overfitting and gives almost zero training error. Now, for a given test example, since we assume that the training set is infinite, it essentially implies that the test example will have some training example in very close proximity, and thus, the error rate would be very low, close to 0. Thus, 1-NN is a consistent classification method.

Problem 15. (Feature Selection)

We covered ridge (l_2- regularised) and l_1-regularised (lasso) regression in class. A hybrid version called elastic net exists which uses both l_1 and l_2 regularisation terms:

$$J_{\text{EL}} = \|\mathbf{y} - \mathbf{X}\mathbf{w}\|^2 + \lambda_1\|\mathbf{w}\|_1 + \lambda_2\|\mathbf{w}\|^2$$

Defining

$$J_2 = \|\tilde{\mathbf{y}} - \tilde{\mathbf{X}}\mathbf{w}\|^2 + c\lambda_1\|\mathbf{w}\|_1$$

where $c = (1+\lambda_2)^{-1/2}$ and

$$\tilde{\mathbf{X}} = c \begin{pmatrix} \mathbf{X} \\ \sqrt{\lambda_2}\mathbf{I}_d \end{pmatrix}, \quad \tilde{\mathbf{y}} = \begin{pmatrix} \mathbf{y} \\ \mathbf{0}_{d\times 1} \end{pmatrix}.$$

Show that

$$\arg\min_{\mathbf{w}} J_{\text{EL}}(\mathbf{w}) = c\left(\text{argmin}_{\mathbf{w}} J_2(\mathbf{w})\right)$$

This implies that an elastic net problem can be solved as a lasso problem using modified data.

Solution:

$$\begin{aligned}
J_{EL}(w) &= |y - Xw|^2 + \lambda_2|w|^2 + \lambda_1|w|_1 \\
cJ_{EL}(w) &= c|y - Xw|^2 + c\lambda_2|w|^2 + c\lambda_1|w|_1 \\
&= cy^Ty - 2cy^TXw + cw^TX^TXw + c\lambda_2|w|^2 + c\lambda_1|w|_1 \\
&= cy^Ty - 2cy^TXw + c\left(w^TX^TXw + \lambda_2|w|^2\right) + c\lambda_1|w|_1 \\
&= c\tilde{y}^T\tilde{y} - 2\tilde{y}^T\tilde{X}w + c\left(w^TX^TXw + \lambda_2|w|^2\right) + c\lambda_1|w|_1 \\
&= c\tilde{y}^T\tilde{y} - 2\tilde{y}^T\tilde{X}w + \frac{1}{c}\left(w^T\tilde{X}^T\tilde{X}w\right) + c\lambda_1|w|_1
\end{aligned}$$

Let $\tilde{w} = c^{-1}w$

$$\begin{aligned}
J_{EL}(\tilde{w}) &= c\tilde{y}^T\tilde{y} - 2c\tilde{y}^T\tilde{X}\tilde{w} + c\left(\tilde{w}^T\tilde{X}^T\tilde{X}\tilde{w}\right) + c^2\lambda_1|\tilde{w}|_1 \\
J_{EL}(\tilde{w}) &= c|\tilde{Y} - \tilde{X}w|^2 + c^2\lambda_1|\tilde{w}|_1
\end{aligned}$$

Problem 16. (On Statistical Distances)

In some situations in statistics and machine learning, the objective that we are going to optimize is some distribution (e.g., in the E-step of the EM algorithm). This motivates us to understand a bit more about the space of probability distributions. For simplicity, let $\mathcal{P}$ be the set of all probability distributions over the set $\{1,\ldots,n\}$, i.e.,

$$\mathcal{P} = \left\{(p_1,\ldots,p_n) \mid \sum p_i = 1, p_i \geq 0\right\}.$$

Usually, $\mathcal{P}$ is called the probability simplex, or simply the n-simplex. One can equip $\mathcal{P}$ with a metric, inducing a geometry on $\mathcal{P}$. Recall that a metric is a function $d : \mathcal{P}\times\mathcal{P} \to \mathbb{R}_+$ satisfying the following criteria:

- (Non-negativity) $d(p,q) \geq 0$ for all $p,q \in \mathcal{P}$ and equality holds iff $p = q$,
- (Symmetry) $d(p,q) = d(q,p)$,
- (Triangle inequality) $d(p,q) + d(q,r) \geq d(p,r)$.

A metric on the probability simplex is also called a **statistical distance**. Here we mention a few distances and some of their properties:

(a) **Total Variation Distance**. For $p,q \in \mathcal{P}$, we define their TV distance as

$$\text{D}_{\text{TV}}(p,q) := \frac{1}{2}\|p-q\|_1 = \frac{1}{2}\sum_{i=1}^n |p_i - q_i|.$$

Prove that TV distance is indeed a metric and equals to the largest possible difference between the probabilities that the two probability distributions p and q can assign to the same event, i.e.

$$\text{D}_{\text{TV}}(p,q) = \max_{E\subseteq\{1,\ldots,n\}} |p(E) - q(E)|.$$

(b) **Kullback-Leibler Divergence**. For $p,q \in \mathcal{P}$, we define their KL divergence as

$$\text{D}_{\text{KL}}(p\|q) = -\sum_{i=1}^n p_i \log\frac{q_i}{p_i}.$$

(i) Prove that KL divergence satisfies the first property of a metric: it is non-negative, and it is zero if and only if the distributions are equal.

(ii) Give an example that $\mathrm{D}_{\mathrm{KL}}(p\|q) \neq \mathrm{D}_{\mathrm{KL}}(q\|p)$.

(iii) Give a counter-example for the triangle inequality for KL divergence.

(iv) Prove the Pinsker's Inequality:

$$\mathrm{D}_{\mathrm{TV}}(p,q) \leq \sqrt{\frac{1}{2}\mathrm{D}_{\mathrm{KL}}(p\|q)}.$$

(v) Although KL divergence fails to be a metric on $\mathcal{P}$, it satisfies some convergence properties. As an example, prove the following theorem: Let $p^{(1)}, p^{(2)}, \ldots$ be a sequence of probability distributions in $\mathcal{P}$, such that

$$\lim_{n\to\infty} \mathrm{D}_{\mathrm{KL}}\left(p^{(n)}\|q\right) = 0,$$

i.e. the sequence is "converging" to q with respect to KL divergence. Prove that this sequence is actually converging to q in Euclidean sense, i.e.

$$\lim_{n\to\infty} \left\|p^{(n)} - q\right\|_2 = 0.$$

(vi) Let X and Y be two random variables with distributions p_X and p_Y and joint distribution $p_{X,Y}$. If X and Y were independent, the we had $p_{X,Y} = p_X p_Y$. Otherwise, if one tries to give a "measure of independence" of X and Y, one idea is to consider

$$\mathrm{D}_{\mathrm{KL}}\left(p_{X,Y}\|p_X p_Y\right)$$

This value is called the **mutual information** between X and Y, denoted by $I(X,Y)$. Prove that

$$I(X,Y) = H(X) - H(X \mid Y),$$

where $H(X)$ is the entropy[4] of X and $H(X \mid Y)$ is the conditional entropy of X given Y. From the Bayesian point of view, mutual information shows how much information knowledge about Y reveals about X.

Solution:

(a) We first prove that Total Variation is a distance function. Let $p, q, r \in \mathcal{P}$ be three probability distributions over the set $\{1, \ldots, n\}$. Non-negativity follows by definition,

$$\mathrm{D}_{\mathrm{TV}}(p,q) = \frac{1}{2}\|p-q\|_1 \geq 0.$$

Symmetry follows from $\|p-q\|_1 = \|q-p\|_1$. Also, Triangle inequality follows from the triangle inequality for ℓ_1 norm.

These three properties show that the Total Variation distance is indeed a distance function.

Now we prove the second argument. Let $E = \{i : p_i \geq q_i\}$ be the event that contains the elements which p gives higher probability than q. We claim that E attains the maximum value of $|p(F) - q(F)|$ among all events $F \subseteq \{1, \ldots, n\}$

By writing $F = (F \cap E) \cup (F \cap E^c)$ we observe that

$$p(F) - q(F) = p(F \cap E) - q(F \cap E) + \underbrace{p\left(F \cap E^c\right) - q\left(F \cap E^c\right)}_{\leq 0} \leq p(E) - q(E). \tag{3}$$

This is true since for all elements $i \in E^c$ we have $p_i < q_i$, which makes $p\left(F \cap E^c\right) - q\left(F \cap E^c\right) \leq 0$, and adding elements in E to $F \cap E$ will not decrease the difference in probability, meaning $p(F \cap E) - q(F \cap E) \leq p(E) - q(E)$.

With the same argument, but for E^c this time, we arrive at

$$q(F) - p(F) \leq q\left(E^c\right) - p\left(E^c\right). \tag{4}$$

[4] "Entropy of a random variable X is defined as $H(X) := \mathbb{E}_X[-\log X] = -\sum_x p_X(x)\log p_X(x)$, and is a measure of "uncertainty" of X. For example if X has the uniform distribution, it has the highest entropy. If the base of log is 2, entropy is measured with the unit "bits", suggesting the idea that one needs $H(X)$ bits to encode the outcome of X with zeros and ones. Convince yourself that this definition makes sense.

Since $p(E)-q(E)=q\left(E^c\right)-p\left(E^c\right)$, the upper bounds for Eqn. (3) and qn. (**??**) become the same, and we get

$$p(E)-q(E)=\max_F |p(F)-q(F)|.$$

Also, by definition of E, we can write

$$\|p-q\|_1=\sum_{i\in E}(p_i-q_i)+\sum_{j\notin E}(q_j-p_j)=p(E)-q(E)+q\left(E^c\right)-p\left(E^c\right)=2(p(E)-q(E)),$$

where the last equality is because $p(E)-q(E)=q\left(E^c\right)-p\left(E^c\right)$.

(b) The solutions for part (b) are given below:

(i) We prove that for $p,q\in\mathcal{P}$, we have $\mathrm{D}_{\mathrm{KL}}(p\|q)\geq 0$. Note that the positivity of the KL Divergence is regardless of the basis of the logarithm since for all $a>1$, we have $\log_a(x)=\ln(x)/\ln(a)$. Hence we prove that for all $p,q\in\mathcal{P}$ we have

$$-\sum_{i=1}^n p_i\ln\frac{q_i}{p_i}\geq 0.$$

A useful inequality about logarithms is $\ln(x)\leq x-1$ for all $x>0$, with equality iff $x=1$. Using this inequality, we have

$$\begin{aligned}-\sum_{i=1}^n p_i\ln\frac{q_i}{p_i}&\geq-\sum_{i=1}^n p_i\left(\frac{q_i}{p_i}-1\right)\\&=-\sum_{i=1}^n(q_i-p_i)=0.\end{aligned}$$

The equality only happens if $p_i=q_i$ for all i, or equivalently when $p=q$.

(ii) Take $p=(0.1,0.9)$ and $q=(0.5,0.5)$. Then we have

$$\mathrm{D}_{\mathrm{KL}}(p\|q)=0.1\times\log 0.2+0.9\times\log 1.8\approx 0.531$$

while

$$\mathrm{D}_{\mathrm{KL}}(q\|p)=0.5\times\log 5+0.5\times\log\frac{5}{9}\approx 0.737$$

(iii) To give a counterexample for Triangle inequality, we should provide three distributions $p,q,r\in\mathcal{P}$, such that

$$\mathrm{D}_{\mathrm{KL}}(p\|q)+\mathrm{D}_{\mathrm{KL}}(q\|r)<\mathrm{D}_{\mathrm{KL}}(p\|r).$$

By moving the first term to the right-hand side and expanding the definition of KL divergence, we need to have

$$\sum_i q_i\log\frac{q_i}{r_i}<\sum_i p_i\left(\log\frac{p_i}{r_i}-\log\frac{p_i}{q_i}\right)=\sum_i p_i\log\frac{q_i}{r_i}$$

This suggests that we take q and r, two arbitrary distributions, and find a p that makes this inequality possible. Take $q=(0.5,0.5)$ and $r=(0.1,0.9)$. Then $\log\frac{q_1}{r_1}\approx 2.322$ and $\log\frac{q_2}{r_2}\approx-0.847$. Now take $p=(1,0)$. In this way we have

$$\sum_i q_i\log\frac{q_i}{r_i}\approx 0.737$$

but

$$\sum_i p_i\log\frac{q_i}{r_i}=\log\frac{q_1}{r_1}\approx 2.322$$

(iv) We first prove this inequality for the case that p and q are two probability distributions over a set of two elements, i.e., $p=(p,1-p)$ and $q=(q,1-q)$. In this case we have $\|p-q\|_1=2|p-q|$. We shall prove

$$2(p-q)^2=\frac{1}{2}\|p-q\|_1^2\overset{?}{\leq}\mathrm{D}_{\mathrm{KL}}(p\|q)=p\log\frac{p}{q}+(1-p)\log\frac{1-p}{1-q}.$$

To prove this inequality, we fix p and look at

$$f(q)=p\log\frac{p}{q}+(1-p)\log\frac{1-p}{1-q}-2(p-q)^2$$

So it suffices to prove $f(q)$ is always nonnegative. First, observe that at $q = p$, we have $f(p) = 0$. Taking the derivative w.r.t. q we have

$$f'(q) = -\frac{p}{q} + \frac{1-p}{1-q} + 4(p-q) = (q-p)\left(\frac{1}{q(1-q)} - 4\right)$$

Since for $0 < q < 1$ we know $q(1-q) \leq \frac{1}{4}$, it follows immediately that the sign of the derivative is the same as $q - p$. This concludes that the point p is the minimum of f, with a value of 0. For the more general case, we try to reduce the problem to the case we already solved. Before doing this, let us state (without proof; the reader can carry out the proof) a useful and important lemma:

Lemma 3. *(Log-Sum inequality) Let $a_1, \ldots, a_n$ and $b_1, \ldots, b_n$ be nonnegative numbers. Then*

$$\sum_{i=1}^{n} a_i \log \frac{a_i}{b_i} \geq a \log \frac{a}{b}$$

where $a = \sum a_i$ and $b = \sum b_i$.

Now let p and q be two probability distributions over a set of n elements. Let $A = \{i : p_i \geq q_i\}$. Define $\tilde{p} = \sum_{i \in A} p_i$ and $\tilde{q} = \sum_{i \in A} q_i$ and take the distributions $\tilde{p} = (\tilde{p}, 1 - \tilde{p})$ and $\tilde{q} = (\tilde{q}, 1 - \tilde{q})$. We show that the Pinsker's inequality for p and q is reduced to the Pinsker inequality for $\tilde{p}$ and $\tilde{q}$, which we have already proved. To show this reduction, we first show that $\mathrm{D_{TV}}(p, q) = \mathrm{D_{TV}}(\tilde{p}, \tilde{q})$. This follows from

$$\begin{aligned}
\mathrm{D_{TV}}(p, q) &= \frac{1}{2} \sum_{i=1}^{n} |p_i - q_i| \\
&= \frac{1}{2} \sum_{i \in A} (p_i - q_i) + \frac{1}{2} \sum_{i \notin A} (q_i - p_i) \\
&= \tilde{p} - \tilde{q} = \mathrm{D_{TV}}(\tilde{p}, \tilde{q})
\end{aligned}$$

Next, we show that $\mathrm{D_{KL}}(p \| q) \geq \mathrm{D_{KL}}(\tilde{p} \| \tilde{q})$

$$\begin{aligned}
\mathrm{D_{KL}}(p \| q) &= \sum_{i=1}^{n} p_i \log \frac{p_i}{q_i} \\
&= \sum_{i \in A} p_i \log \frac{p_i}{q_i} + \sum_{i \notin A} p_i \log \frac{p_i}{q_i} \\
&\geq \tilde{p} \log \frac{\tilde{p}}{\tilde{q}} + (1 - \tilde{p}) \log \frac{1 - \tilde{p}}{1 - \tilde{q}} = \mathrm{D_{KL}}(\tilde{p} \| \tilde{q})
\end{aligned}$$

where the inequality follows from the Log-Sum inequality. Putting it all together we have

$$\mathrm{D_{KL}}(p \| q) \geq \mathrm{D_{KL}}(\tilde{p} \| \tilde{q}) \geq 2\mathrm{D_{TV}}(\tilde{p}, \tilde{q})^2 = 2\mathrm{D_{TV}}(p, q)^2$$

(v) By the Pinsker's inequality, we know that if $\mathrm{D_{KL}}\left(p^{(n)} \| q\right) \to 0$, then $\mathrm{D_{TV}}\left(p^{(n)}, q\right) \to 0$, meaning that $\left\|p^{(n)} - q\right\|_1 \to 0$. But since in finite dimensions, all norms are equivalent, meaning that there is a constant C such that $\|p - q\|_2 \leq C\|p - q\|_1$ for all p, q, then this implies that $\left\|p^{(n)} - q\right\|_2 \to 0$.

(vi) By definition we have

$$I(X, Y) = \mathrm{D_{KL}}\left(p_{X,Y} \| p_X p_Y\right) = \sum_{i,j} p_{X,Y}(i, j) \log \frac{p_{X,Y}(i, j)}{p_X(i) p_Y(j)}$$

Using the chain rule for probabilities, we have $p_{X,Y}(i, j) = p_{X|Y}(i \mid j) p_Y(j)$. Hence

$$\begin{aligned}
I(X, Y) &= \sum_{i,j} p_{X,Y}(i, j) \log \frac{p_{X|Y}(i \mid j) p_Y(j)}{p_X(i) p_Y(j)} \\
&= -\sum_{i,j} p_{X,Y}(i, j) \log p_X(i) + \sum_{i,j} p_{X,Y}(i, j) \log p_{X|Y}(i \mid j).
\end{aligned}$$

In the first sum, the summation on j changes $p_{X,Y}(i, j)$ to $p_X(i)$, and the sum becomes $H(X)$. The second sum is, by definition, minus the conditional entropy. So we have

$$I(X, Y) = H(X) - H(X \mid Y)$$

Problem 17. (Entropy for Classification Tree)

(a) Let X be a discrete random variable with $P(X = x_i) = p_i$ for $i \in 1, 2, \ldots, n$. The entropy $\mathcal{H}[X]$ of the random variable X is a measure of its uncertainty. It is defined as:

$$\mathcal{H}[X] = -\sum_{i=1}^{n} p_i \log p_i,$$

where log denotes the natural logarithm. Show that the entropy $\mathcal{H}[X]$ is maximized when $p_i = \frac{1}{n}$ for all i. You should do this by computing the gradient with respect to p_i and using Lagrange multipliers to enforce the constraint that $\sum_i p_i = 1$ (In the course, we use similar calculations in classification trees).

(b) The joint entropy of n discrete random variables $(X_1, X_2, \ldots, X_n)$ is defined as:

$$\mathcal{H}(X_1, X_2, \ldots, X_n) = -\sum_{x_1}\sum_{x_2}\ldots\sum_{x_n} P(x_1, x_2, \ldots, x_n) \log P(x_1, x_2, \ldots, x_n)$$

where the sums range over all possible instantiations of $(X_1, X_2, \ldots, X_n)$. Show that if the variables X_i are independent, then their joint entropy is the sum of their individual entropies: namely,

$$P(x_1, x_2, \ldots, x_n) = \prod_{i=1}^{n} P(x_i) \quad \text{implies} \quad \mathcal{H}(X_1, X_2, \ldots, X_n) = \sum_{i=1}^{n} \mathcal{H}(X_i).$$

Solution:

(a) We want to solve

$$\max_{p_1, p_2, \ldots, p_n} \mathcal{H}[X] = -\sum_{i=1}^{n} p_i \log p_i, \text{ s.t. } \sum_{i=1}^{n} p_i = 1$$

Firstly, we write down the Lagrange function as

$$\mathcal{L}(p_1, p_2 \ldots, p_n, \lambda) = -\sum_{i=1}^{n} p_i \log p_i - \lambda \cdot \left(\sum_{i=1}^{n} p_i - 1\right).$$

Secondly, we let

$$\begin{aligned} \frac{\partial \mathcal{L}(p_1, p_2, \ldots, p_n, \lambda)}{\partial p_i} &= -\left(\log p_i + p_i \cdot \frac{1}{p_i}\right) - \lambda \cdot 1 = -1 - \log p_i - \lambda = 0 \\ p_i &= e^{-1-\lambda}. \end{aligned}$$

Then, we rewrite the Lagrange function as

$$\mathcal{L}(\lambda) = -n \cdot e^{-1-\lambda} \cdot (1 - \lambda) - \lambda \cdot \left(n \cdot e^{-1-\lambda} - 1\right) = \lambda - n \cdot e^{-1-\lambda}.$$

When optimal occurs, we have

$$\begin{aligned} &\frac{\partial \mathcal{L}(\lambda)}{\partial \lambda} = 1 - n \cdot (-1) \cdot e^{-1-\lambda} = 0 \\ \Rightarrow \quad & e^{-1-\lambda} = \frac{1}{n} \\ \Rightarrow \quad & p_i = \frac{1}{n}. \end{aligned}$$

(b)

$$\begin{aligned}\mathcal{H}\left(X_1, X_2, \ldots, X_n\right) &= -\sum_{x_1}\sum_{x_2}\cdots\sum_{x_n}\left[P\left(x_1, x_2, \ldots, x_n\right)\log\left(\prod_{i=1}^{n} P\left(x_i\right)\right)\right] \\ &= -\sum_{x_1}\sum_{x_2}\cdots\sum_{x_n}\left[P\left(x_1, x_2, \ldots, x_n\right)\sum_{i=1}^{n}\log P\left(x_i\right)\right] \\ &= -\sum_{x_1}\sum_{x_2}\cdots\sum_{x_n}\left[\sum_{i=1}^{n}\left[P\left(x_1, x_2, \ldots, x_n\right)\log P\left(x_i\right)\right]\right]\end{aligned}$$

We can reorder the above summation formula and sum them through $\log P\left(x_i\right)$. Note that x_i was determined before the square brackets below.

$$\mathcal{H}\left(X_1, X_2, \ldots, X_n\right) = -\sum_{i=1}^{n}\sum_{x_i}\left[\log P\left(x_i\right)\cdot\sum_{j=1,j\neq i}^{n}\sum_{x_j}P\left(x_1, x_2, \ldots, x_n\right)\right]$$

From above, we can get $\mathcal{H}\left(X_i\right)$ by accumulating all other $x_j, j \in \{1, \ldots, n\}, j \neq i$, that is

$$\begin{aligned}\mathcal{H}\left(X_1, X_2, \ldots, X_n\right) &= -\sum_{i=1}^{n}\sum_{x_i}\left[\log P\left(x_i\right)\cdot P\left(x_i\right)\right] \\ &= \sum_{i=1}^{n}\left[-\sum_{x_i}P\left(x_i\right)\log P\left(x_i\right)\right] \\ &= \sum_{i=1}^{n}\mathcal{H}\left(X_i\right).\end{aligned}$$

Problem 18. (Misclassification Rate Vs. Information Gain)

Consider a binary classification data set consisting of 400 data points from class 0 and 400 data points from class 1. Suppose that a decision tree model $\mathcal{A}$ splits these into $(300, 100)$ at the first leaf node and $(100, 300)$ at the second leaf node. (Here, (n, m) denotes that n points are assigned to class 0 and m points are assigned to class 1.) Similarly, suppose that a second decision tree model $\mathcal{B}$ splits them into $(200, 400)$ and $(200, 0)$. (See the figure below.)

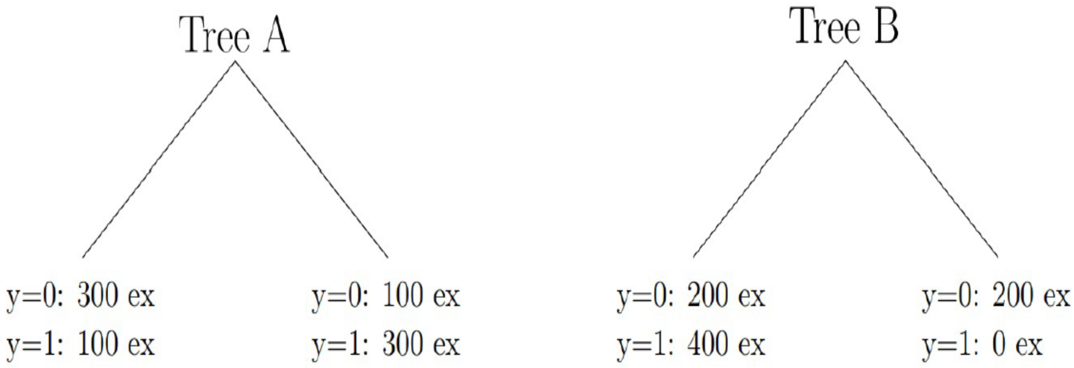


Figure 1: Two Decision Tree Models

(a) Compute the training data misclassification rate (i.e., what fraction of training examples will be misclassified) for the two trees: are they equal or not?

(b) Evaluate the information gain for the two trees and use these to compare the trees.

(c) Do you get different answers for (a) and (b)? Does this make sense?

Solution:

We will introduce the notations and state the definitions that will be needed for the problem.

- T_A : Tree A, T_B : Tree B
- $IG(T)$: Information gain of tree T
- $MR(T)$: Misclassification rate of tree T
- D : original training dataset, D_t : training subset at node t
- N : total number of training samples, N_t : number of training samples at node t
- $H(S)$: Entropy of $S \subseteq D$
- Let p denote the parent node, l denote the left node and r the right node
- $p_c(t) := \frac{N_t^c}{N_t}$: probability of a sample belonging to class c at node t

The formal definitions of misclassification rate and information gain are given by:

$$MR(T) := \sum_{t \in \text{ leaves } (T)} \frac{N_t}{N}\left(1 - \max_{c \in \{0,1\}} \{p_c(t)\}\right)$$

$$IG(T) := H(D_p) - \left(\frac{N_l}{N} H(D_l) + \frac{N_r}{N} H(D_r)\right) \text{ where } H(D_t) = -\sum_{c \in \text{ classes}} p_c(t) \log_2(p_c(t))$$

(a) Misclassification rate: For T_A, substituting $N = 800, N_l = N_r = 400$, and observing

$$p_0(l) = 300/400 = 3/4, p_1(l) = 100/400 = 1/4$$
$$p_0(r) = 100/400 = 1/4, p_1(r) = 300/400 = 3/4$$

Now, we substitute these values in the above formulae of MR(T) and get

$$\text{MR}(T_A) = (1/2)[1/4 + 1/4] = 1/4 = 0.25$$

Similarly, working out for T_B yields

$$MR(T_B) = (3/4)[1/3 + 0] = 1/4 = 0.25$$
$$MR(T_A) = 0.25, \quad MR(T_B) = 0.25$$

Thus, we observe that the misclassification rate for both trees is the same. Thus, the two trees are incomparable in terms of the misclassification rate metric.

(b) Information gain: First, we compute $H(D_p), H(D_l)$ and $H(D_r)$ using above formulae of IG(T) for tree T_A.

$$H(D_p) = -\frac{1}{2}\log_2\left(\frac{1}{2}\right) - \frac{1}{2}\log_2\left(\frac{1}{2}\right) = 1$$
$$H(D_l) = -\frac{3}{4}\log_2\left(\frac{3}{4}\right) - \frac{1}{4}\log_2\left(\frac{1}{4}\right) = 0.81125$$
$$H(D_r) = -\frac{1}{4}\log_2\left(\frac{1}{4}\right) - \frac{3}{4}\log_2\left(\frac{3}{4}\right) = 0.81125$$

Further, using the equation for information gain above, we compute the information gain for tree T_A

$$IG(T_A) = 1 - \left(\frac{400}{800}\right) 0.81125 - \left(\frac{400}{800}\right) 0.81125$$
$$\therefore IG(T_A) = 0.18875$$

Similarly, working out for tree T_B, we get

$$\therefore IG(T_B) = 0.3115$$

Thus, we note that $IG(T_B) > IG(T_A)$ indicating that data split in the latter results in a better decision tree in terms of the metric (information gain) based on entropy.

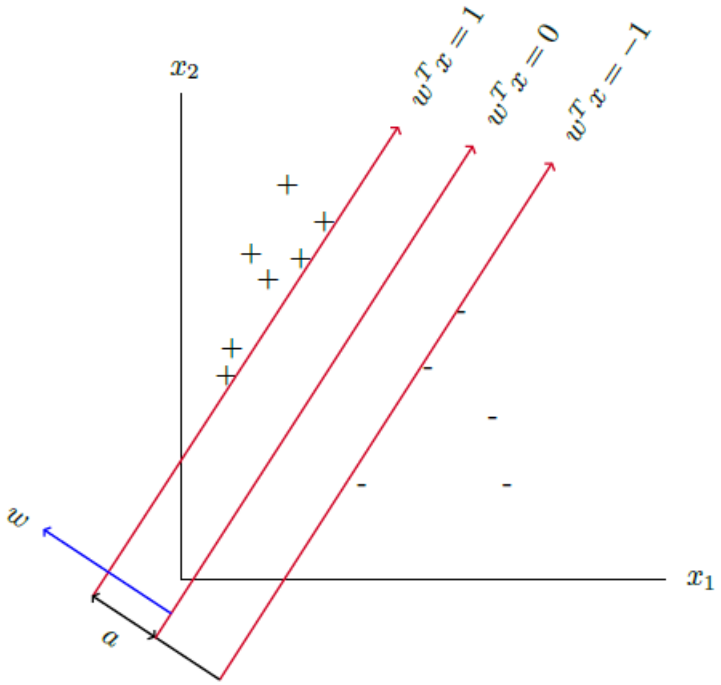


Figure 2: Classification

(c) Comparison: We get different answers in (a) and (b). This indicates that entropy-based information gain is a better metric than the misclassification rate since it is evident that the split is purer in the tree T_B relative to that of tree T_A, and the information gain reflects the same. This, however, is not the case with the misclassification rate, which assigns the same values to both trees.

Problem 19. (Classification)

Consider the data set plotted in Fig. 2. Show that $a = \frac{1}{\|w\|}$. How would L_2 regularization on w affect the margin around $w^T x = 0$?

Solution:

First, we use the fact that the minimal Euclidean distance from any point on a plane $\left(w^T x = b\right)$ to the origin is $\frac{|b|}{\|w\|}$.

We know that our other vectors are $w^T x + b = 1$ and $w^T x + b = -1$. This is re-written as $w^T x = 1 - b$ and $w^T x = -1 - b$. The distance between these two vectors is then $\frac{2}{\|w\|}$. Therefore, a is $\frac{1}{\|w\|}$.

We see that the margin around $w^T x = 0$ is maximized when $\|w\|$ is minimized.

Problem 20. (Model selection and Cross-validation)

Suppose we are given a noise-free set of points $X = \{x_i\}_{i=1}^n \subset (-1, 1), Y = \{\sin(x_i)\}$, which we want to fit with a polynomial, but we do not know which degree to choose. Suppose our candidate polynomial families are $\mathcal{P}_k = \{\mathbb{P}_{2i+1}\}_{i=0}^k$, where $\mathbb{P}_{2i+1}$ denotes the family of polynomials with real-valued coefficients of maximum degree $2i+1$. We want to find the optimal hyperparameter value $\hat{k} \in \{1, \ldots, k\}$.

Given a family of polynomials $\mathbb{P}_{2\ell+1}$ and a training set, suppose we have an oracle (i.e., an exact algorithm) that is able to find the polynomial $\hat{p} \in \mathbb{P}_{2\ell+1}$ with optimal coefficients with respect to the square loss objective

$$\mathcal{L}(X, Y, p)=\sum_{i=1}^{n}\left(y_{i}-p\left(x_{i}\right)\right)^{2}, \quad p \in \mathbb{P}_{2\ell+1}$$

(a) Show that when the optimization is performed on each family in $\mathcal{P}_k$, the lowest score is achieved when $\hat{p} \in \mathbb{P}_{2k+1} \backslash \mathbb{P}_{2k-1}$ (i.e., $\hat{p}$ will be of degree $2k+1$).

(b) What potential issue with using cross-validation does this demonstrate?

(c) Suppose we widen the boundaries of X to $(-2\pi, 2\pi)$. Write a short script to simulate samples $\left(X_i, \tilde{Y}_i\right)$ with different values of σ_i^2 and use 10-fold cross-validation to find corresponding optimal values $\hat{k}_i$. How do $\mathcal{L}\left(X_i, \hat{Y}_i, p\right)$ and $\hat{k}_i$ behave as k and σ^2 increase?

Solution:

(a) Note: We should consider polynomials of the following type, not the odd-ordered polynomials stated in the question previously

$$\begin{aligned}
&\mathcal{P}_1 = w_1 x\\
&\mathcal{P}_3 = w_1 x - w_2 x^3\\
&\mathcal{P}_5 = w_1 x - w_2 x^3 + w_3 x^5\\
&\mathcal{P}_7 = w_1 x - w_2 x^3 + w_3 x^5 - w_3 x^7\\
&\ldots
\end{aligned}$$

We also remember the Taylor-series approximation of $\sin(x)$

$$\sin(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} \cdots$$

The loss can be calculated as:

$$\begin{aligned}
\mathcal{L}(X, Y, p, k) &= \sum_{i=1}^{n}\left(y_i - p\left(x_i\right)\right)^2, \quad p \in \mathbb{P}_{2\ell+1}. \\
&= \sum_{i=1}^{n}\left(\sin\left(x_i\right) - p_{2k+1}\left(x_i\right)\right)^2 \qquad \text{all terms in Taylor series expansion will be eliminated up to 2k + 1} \\
&= \sum_{i=1}^{n} \mathcal{O}\left(x_i^{2k+3}\right)^2 = \sum_{i=1}^{n} \mathcal{O}\left(x_i^{4k+6}\right) > \sum_{i=1}^{n} \mathcal{O}\left(x_i^{4(k+1)+6}\right) = \sum_{i=1}^{n} \mathcal{O}\left(x_i^{4k+10}\right)
\end{aligned}$$

(b) When fitting a model, we can still choose overly complex models even when using cross-validation. In this case, it occurred because of a special relationship between our data and the family of functions we were using to approximate it.

(c) As σ^2 increases, we will no longer have a special relation between our learned function and $\sin(x)$. This means that we will no longer be guaranteed to get the more complex model when using cross-validation (also see Fig. 3).

Problem 21. (Kernel function)

Kernel functions implicitly define some mapping function $\phi(\cdot)$ that transforms an input instance $\mathbf{x} \in \mathbb{R}^d$ to high dimensional space Q by giving the form of the dot product in $Q : K\left(\mathbf{x}_i, \mathbf{x}_j\right) \equiv \left\langle\phi\left(\mathbf{x}_i\right), \phi\left(\mathbf{x}_j\right)\right\rangle$.

(a) Prove that the kernel is symmetric, i.e. $K\left(\mathbf{x}_i, \mathbf{x}_j\right) = K\left(\mathbf{x}_j, \mathbf{x}_i\right)$.

(b) Assume we use radial basis kernel function $K\left(\mathbf{x}_i, \mathbf{x}_j\right) = \exp\left(-\frac{1}{2}\left\|\mathbf{x}_i - \mathbf{x}_j\right\|^2\right)$. Thus there is some implicit unknown mapping function $\phi(\mathbf{x})$. Prove that for any two input instances $\mathbf{x}_i$ and $\mathbf{x}_j$, the squared Euclidean distance of their corresponding points in the feature space Q is less than 2, i.e., prove that $\left\|\phi\left(\mathbf{x}_i\right) - \phi\left(\mathbf{x}_j\right)\right\|^2 \leq 2$.

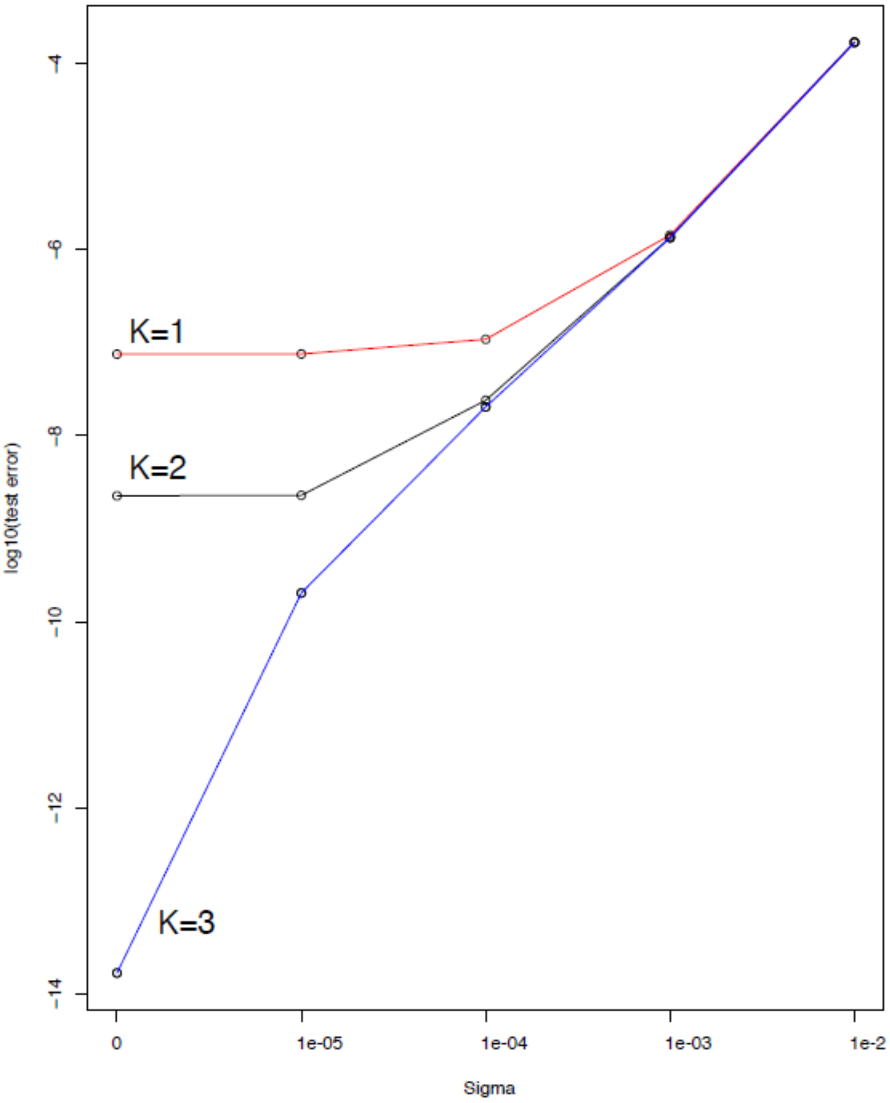


Figure 3: Model selection and Cross-validation

Solution:

(a) We have $K(\mathbf{x}_i, \mathbf{x}_j) = \langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_j) \rangle = \langle \phi(\mathbf{x}_j), \phi(\mathbf{x}_i) \rangle = K(\mathbf{x}_j, \mathbf{x}_i)$.

(b) We have

$$
\begin{aligned}
&\|\phi(\mathbf{x}_i) - \phi(\mathbf{x}_j)\|^2 \\
=&\langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_i) \rangle + \langle \phi(\mathbf{x}_j), \phi(\mathbf{x}_j) \rangle - 2 \cdot \langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_j) \rangle \\
=&K(\mathbf{x}_i, \mathbf{x}_i) + K(\mathbf{x}_j, \mathbf{x}_j) - 2 \cdot K(\mathbf{x}_i, \mathbf{x}_j) \\
=&1 + 1 - 2\exp\left(-\frac{1}{2}\|\mathbf{x}_i - \mathbf{x}_j\|^2\right) < \ 2
\end{aligned}
$$

Problem 22. (Support vector machine)

With the help of a kernel function, SVM attempts to construct a hyper-plane in the feature space Q that maximizes the margin between two classes. The classification decision of any $\mathbf{x}$ is made on the basis of the sign of

$$\langle \hat{\mathbf{w}}, \phi(\mathbf{x}) \rangle + \hat{w}_0 = \sum_{i \in SV} y_i \alpha_i K\left(\mathbf{x}_i, \mathbf{x}\right) + \hat{w}_0 = f\left(\mathbf{x}; \alpha, \hat{w}_0\right),$$

where $\hat{\mathbf{w}}$ and $\hat{w}_0$ are parameters for the classification hyper-plane in the feature space Q, SV is the set of support vectors, and α_i is the coefficient for the i-th support vector. Again we use the radial basis kernel function. Assume that the training instances are linearly separable in the feature space Q, and assume that the SVM finds a margin that perfectly separates the points.

If we choose a test point $\mathbf{x}_{far}$ which is far away from any training instance $\mathbf{x}_i$ (distance here is measured in the original space $\mathbb{R}^d$), prove that $f\left(\mathbf{x}_{\text{far}}\,; \alpha, \hat{w}_0\right) \approx \hat{w}_0$.

Solution:

$$\begin{aligned}
&\left\|\mathbf{x}_{far} - \mathbf{x}_i\right\| \gg 0 \quad \forall i \in SV \\
\Rightarrow &K\left(\mathbf{x}_{far}, \mathbf{x}_i\right) \approx 0 \quad \forall i \in SV \\
\Rightarrow &\sum_{i \in SV} y_i \alpha_i K\left(\mathbf{x}_i, \mathbf{x}\right) \approx 0 \\
\Rightarrow &f\left(\mathbf{x}; \alpha, \hat{w}_0\right) \approx \hat{w}_0
\end{aligned}$$

Problem 23. (Perceptron/ SVM)

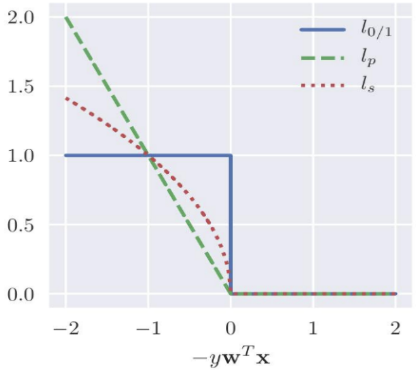
(a) How does the perceptron algorithm relate to stochastic gradient descent?

(b) How does the perceptron objective relate to the support vector machine objective?

(c) Write down the training objective for the SVM and derive the gradient updates using stochastic gradient descent. Assume a minibatch size of B.

(d) The perceptron, in its original formulation, uses a 0/1 loss function (shown below, solid). A surrogate loss function $l_p(\mathbf{w}; \mathbf{x}, y) = \max\left(0, -y\mathbf{w}^T\mathbf{x}\right)$ is instead used in optimisation (dashed). We see that this surrogate loss is a poor match for the 0/1 loss near zero. Suppose we try (shown in a dotted line):

$$l_s(\mathbf{w}; \mathbf{x}, y) = \begin{cases} 0, & \text{for sign}\left(\mathbf{w}^T\mathbf{x}\right) = y \\ \sqrt{-y\mathbf{w}^T\mathbf{x}}, & \text{for sign}\left(\mathbf{w}^T\mathbf{x}\right) \neq y \end{cases}$$



(i) Show that $f(x) = \sqrt{x}$ is not convex.

(ii) Show that $f(x) = x^p$ is convex for all $p \in \mathbb{N}_{>0}$ and $x \in [0, \infty)$. (Hint: use properties of derivatives of convex functions.)

Solution:

(a) Perceptron is SGD on the perception loss function

$$\nabla l_p\left(w_i, x, y\right)=\left\{\begin{array}{l}0 \text{ if } y=\operatorname{sign}\left(w^T x\right) \\ -y x \text{ if } y \neq \operatorname{sign}\left(w^T x\right)\end{array}\right.$$

(b) Perceptron

$$\min \sum_i \max \left\{0,-y_i \alpha^T k_i\right\}$$

SVM

$$\min \sum_i \max \left\{0,1-y_i \alpha^T k_i\right\}+\lambda\|w\|_2^2$$

Difference is essentially an L2 penalty

(c)

$$\begin{gathered}\nabla G(w)=\frac{1}{n} \sum_i^B \nabla g_i(x) \\ \nabla g_i(w)=\nabla \max \left(0,1-y_i w^T x_i\right)+\nabla \lambda\|w\|_2^2\end{gathered}$$

So looking at these separately:

$$\begin{gathered}\nabla \lambda\|w\|^2=2 \lambda w_k \\ \nabla \max \left(0,1-y_i w^T x_i\right)=\left\{\begin{array}{l}0 \text{ if } y_i w^T x_i \geq 1 \\ -y_i x_i \text{ otherwise}\end{array}\right.\end{gathered}$$

So from this, we get:

$$\begin{aligned}& w=w+\eta(-2 \lambda w) \text{ if } y_i w^T x_i \geq 1 \\ & w=w+\eta\left(y_i x_i-2 \lambda w\right) \text{ otherwise}\end{aligned}$$

(d) From the given graph we have:

(i) Can show that $-f(x)$ is convex

$$\begin{aligned}\sqrt{t x_1+(1-t) x_2} & >t \sqrt{x_1}+(1-t) \sqrt{x_2} \\ t x_1+(1-t) x_2 & >t^2 x_1+(1-t)^2 x_2+t 2(1-t) \sqrt{x_1 x 2} \\ x_1+x_2 & >2 \sqrt{x_1 x_2} \\ \left(\sqrt{x_1}-\sqrt{x_2}\right)^2 & >0\end{aligned}$$

(ii) Note that x is restricted to be positive, so one can easily check a few examples of p to gain some intuition.

$$\begin{aligned}& f^{\prime \prime}=p(p-1) x^{p-2} \\ & p(p-1)>0\end{aligned}$$

Since p is positive and x positive, the second derivated will always be positive

Problem 24. (Kernels)

(a) For $\mathbf{x}, \mathbf{x}^{\prime} \in \mathbb{R}^d$, and $K\left(\mathbf{x}, \mathbf{x}^{\prime}\right)=\left(\mathbf{x}^T \mathbf{x}^{\prime}+1\right)^2$, find a feature map $\phi(\mathbf{x})$, such that $k\left(\mathbf{x}, \mathbf{x}^{\prime}\right)=\phi(\mathbf{x})^{\top} \phi\left(\mathbf{x}^{\prime}\right)$.

(b) For the dataset $X=\left\{\mathbf{x}_i\right\}_{i=1,2}=\{(-3,4),(1,0)\}$ and the feature map $\phi(\mathbf{x})=\left[x^{(1)}, x^{(2)},\|\mathbf{x}\|\right]$, calculate the **Gram matrix** (for a vector $\mathbf{x} \in \mathbb{R}^2$ we denote by $x^{(1)}, x^{(2)}$ its components).

Solution:

(a)

$$
\begin{aligned}
k\left(\begin{pmatrix} x_1 \\ x_2 \\ \cdot \\ \cdot \\ \cdot \\ x_n \end{pmatrix}\begin{pmatrix} x'_1 \\ x'_2 \\ \cdot \\ \cdot \\ \cdot \\ x'_n \end{pmatrix}\right) &= \left(1 + x_1x'_1 + x_2x'_2 + \ldots + x_nx'_n\right)^2 \\
&= \Big(1 + \left(x_1x'_1\right)^2 + \ldots + \left(x_nx'_n\right)^2 + 2x_1x'_1 + \ldots + 2x_nx'_n + 2x_1x'_1x_2x'_2 + \ldots + 2x_1x'_1x_nx'_n + \ldots\Big) \\
&= \phi(x)^T\phi(x\prime)
\end{aligned}
$$

$$\text{Thus, } \phi(x) = \Big(1, \sqrt{2}x_1, \ldots, \sqrt{2}x_n\Big), \sqrt{2}x_2x_1, \sqrt{2}x_{n-1}x_1, \ldots, \sqrt{2}x_{n-1}x_{n-2}, \sqrt{2}x_nx_1, , \ldots \sqrt{2}x_nx_{n-1}, x_1^2, \ldots, x_n^2\Big).$$

(b)

$$
\begin{aligned}
k(x,x) &= \phi(x)^T\phi\left(x'\right) \\
k(x,x) &= \begin{pmatrix} x^{(1)}x^{(1)} + x^{(2)}x^{(2)} + \|x\|^2 & x^{(1)}x'^{(1)} + x^{(2)}x'^{(2)} + \|x\|\,\|x'\| \\ x^{(1)}x'^{(1)} + x^{(2)}x'^{(2)} + \|x\|\,\|x'\| & x'^{(1)}x'^{(1)} + x'^{(2)}x'^{(2)} + \|x\prime\|^2 \end{pmatrix} \\
k(x,x) &= \begin{pmatrix} 50 & 2 \\ 2 & 2 \end{pmatrix}
\end{aligned}
$$

Problem 25. (The Equivalence)

Consider a constrained version of least squares linear regression where we constrain the ℓ_2 norm of $\boldsymbol{w}$ to be less than or equal to some $c > 0 : \hat{\boldsymbol{w}} = \arg\min_{\boldsymbol{w}} \sum_{n=1}^{N}\left(y_n - \boldsymbol{w}^\top \boldsymbol{x}_n\right)^2, \quad \text{s.t.} \quad \|\boldsymbol{w}\| \leq c.$

Show, formally, that it is possible to have an ℓ_2 regularized least squares linear regression model that will give the exact same solution as the solution to the above-constrained optimization problem.

Solution:

Consider the following constrained version of least squares linear regression:

$$\hat{\mathbf{w}} = \arg\min_{w} \sum_{n=1}^{N}\left(y_n - \mathbf{w}^T\mathbf{x}_n\right)^2 \quad \text{subject to } \|\mathbf{w}\| \leq c$$

Note that the constraint $\|\mathbf{w}\| \leq c$ is equivalent to $\|\mathbf{w}\|^2 \leq c^2$ since $\|\mathbf{w}\| \geq 0$. The Lagrangian function for the modified constrained problem can be stated as follows:

$$\mathcal{L}(\mathbf{w}, \alpha) = \sum_{n=1}^{N}\left(y_n - \mathbf{w}^T\mathbf{x}_n\right)^2 + \alpha\left(\|\mathbf{w}\|^2 - c^2\right)$$

Ignoring the constant and re-writing,

$$\mathcal{L}(\mathbf{w}, \alpha) = \sum_{n=1}^{N}\left(y_n - \mathbf{w}^T\mathbf{x}_n\right)^2 + \alpha\mathbf{w}^T\mathbf{w} = (\mathbf{y} - \mathbf{X}\mathbf{w})^T(\mathbf{y} - \mathbf{X}\mathbf{w}) + \alpha\mathbf{w}^T\mathbf{w}$$

Solving using the dual variable technique (We can do it since both the objective and constraint are convex),

$$
\begin{gathered}
\frac{\partial\mathcal{L}}{\partial\mathbf{w}} = -2\mathbf{X}^T\mathbf{y} + 2\left(\mathbf{X}^T\mathbf{X} + \alpha\mathbf{I}\right)\mathbf{w} = 0 \\
\mathbf{w} = \left(\mathbf{X}^T\mathbf{X} + \alpha\mathbf{I}\right)^{-1}\mathbf{X}^T\mathbf{y} \\
\therefore \mathcal{L}_D(\alpha) = \mathbf{y}^T\mathbf{y} - \mathbf{w}^T\mathbf{X}^Ty - \mathbf{y}^T\mathbf{X}\mathbf{w} + \mathbf{w}^T\left(\mathbf{X}^T\mathbf{X} + \alpha\mathbf{I}\right)\mathbf{w} \\
\therefore \mathcal{L}_D(\alpha) = \mathbf{y}^T\mathbf{y} - \mathbf{w}^T\mathbf{X}^Ty - \mathbf{y}^T\mathbf{X}\mathbf{w} + \mathbf{w}^T\mathbf{X}^T\mathbf{y} = \mathbf{y}^T\mathbf{y} - \mathbf{y}^T\mathbf{X}\mathbf{w}
\end{gathered}
$$

Ignoring the term $\mathbf{y}^T\mathbf{y}$, we get

$$\mathcal{L}_D(\alpha) = -\mathbf{y}^T\mathbf{X}\mathbf{w} = -\left(\mathbf{X}^T\mathbf{y}\right)^T\left(\mathbf{X}^T\mathbf{X}+\alpha\mathbf{I}\right)^{-1}\mathbf{X}^T\mathbf{y} = -\mathbf{Z}^T\mathbf{M}_\alpha^{-1}\mathbf{Z}$$

$$\frac{\partial\mathcal{L}}{\partial\alpha} = -\frac{\partial\left[\mathbf{Z}^T\mathbf{M}_\alpha^{-1}\mathbf{Z}\right]}{\partial\mathbf{M}_\alpha}\frac{\partial\mathbf{M}_\alpha}{\partial\alpha} = \left(\mathbf{M}_\alpha^{-1}\mathbf{Z}\mathbf{Z}^T\mathbf{M}_\alpha^{-1}\right)^{-1}\mathbf{I}$$

$$\therefore\frac{\partial\mathcal{L}}{\partial\alpha} = \left(\mathbf{M}_\alpha^{-1}\mathbf{X}^T\mathbf{y}\mathbf{y}^T\mathbf{X}\mathbf{M}_\alpha^{-1}\right)^{-1}$$

Note that $\mathbf{M}_\alpha^T = \mathbf{M}_\alpha$ and thus we have

$$\therefore\frac{\partial\mathcal{L}}{\partial\alpha} = \left(\left(\mathbf{M}_\alpha^{-1}\mathbf{X}^T\mathbf{y}\right)\left(\mathbf{M}_\alpha^{-1}\mathbf{X}^T\mathbf{y}\right)^T\right)^{-1}$$

Problem 26. (Arbitrary Choice?)

Show that changing the condition $y_n\left(\boldsymbol{w}^\top\boldsymbol{x}_n+b\right)\geq 1$ in SVM to a different condition $y_n\left(\boldsymbol{w}^\top\boldsymbol{x}_n+b\right)\geq m$ does not change the effective separating hyperplane that the SVM learns. Assume the hard-margin SVM for simplicity.

Solution:

The original hard-margin SVM optimization problem can be stated as

$$\arg\min_{w,b}\frac{\|w\|^2}{2}$$
$$\text{subject to}\quad y_n\left(w^Tx_n+b\right)\geq 1,\quad\forall n=1,2,\ldots,N$$

The modified version of SVM involves changing the inequalities as $y_n\left(w^Tx_n+b\right)\geq m,\quad\forall n=1,2,..,N$. Let $\alpha=[\alpha_1,\alpha_2,..,\alpha_N]$ be the Lagrange variables. The Lagrangian can be stated as

$$\mathcal{L}(w,b,\alpha)=\frac{\|w\|^2}{2}+\sum_{n=1}^N\alpha_n\left(m-y_n\left(w^Tx_n+b\right)\right)$$

Using the dual formulation to solve the constrained optimization problem, we have

$$\frac{\partial\mathcal{L}(w,\alpha)}{\partial w}=0\Rightarrow w=\sum_{n=1}^N\alpha_ny_nx_n,\frac{\partial\mathcal{L}(w,\alpha)}{\partial b}=0\Rightarrow\sum_{n=1}^N\alpha_ny_n=0$$

Now, Substituting $w=\sum_{n=1}^N\alpha_ny_nx_n$ in Lagrangian, we get the dual problem as

$$\mathcal{L}_D(\alpha)=w^Tw+\sum_{n=1}^N(\alpha_nm-y_nb)-\sum_{n=1}^Ny_nw^Tx_n=\sum_{n=1}^N\alpha_nm-\frac{1}{2}\sum_{n,l=1}^N\alpha_l\alpha_ny_ly_n\left(x_l^Tx_n\right)$$

Now, the objective can be stated in a compact form as

$$\max_{\alpha\geq 0}m\left(\alpha^T\mathbf{1}\right)-\frac{1}{2}\alpha^TG\alpha$$

where G is an $N\times N$ matrix with $G_{ln}=y_ly_nx_l^Tx_n$, and $\mathbf{1}$ is a vector of ones. Note that m simply turns out to be a multiplicative constant and thus does not affect the solution of the optimization problem. Thus, the solution to modified SVM effectively remains the same as that of the original one.

Problem 27. (Learning SVM via Co-ordinate Ascent)

Consider the soft-margin linear SVM problem

$$\arg\max_{0\leq\boldsymbol{\alpha}\leq C}f(\boldsymbol{\alpha}),$$

where $f(\boldsymbol{\alpha})=\boldsymbol{\alpha}^\top\mathbf{1}-\frac{1}{2}\boldsymbol{\alpha}^\top\mathbf{G}\boldsymbol{\alpha},\mathbf{G}$ is an $N\times N$ matrix such that $G_{nm}=y_ny_m\boldsymbol{x}_n^\top\boldsymbol{x}_m$ and $\boldsymbol{\alpha}=[\alpha_1,\alpha_2,\ldots,\alpha_N]$ are the Lagrange multipliers. Given the optimal $\boldsymbol{\alpha}$, the SVM weight vector is $\boldsymbol{w}=\sum_{n=1}^N\alpha_ny_n\boldsymbol{x}_n$.

Your goal is to derive a **co-ordinate ascent** procedure for the vector $\boldsymbol{\alpha}$, such that each iteration updates a uniformly randomly chosen entry α_n of the vector $\boldsymbol{\alpha}$. However, instead of updating $\boldsymbol{\alpha}$ via standard co-ordinate descent as $\alpha_n = \alpha_n + \eta g_n$ where g_n denotes the n-th entry of the gradient vector $\nabla_{\boldsymbol{\alpha}} f(\boldsymbol{\alpha})$, we will update it as $\alpha_n = \alpha_n + \delta_*$ where $\delta_* = \arg\max_\delta f(\boldsymbol{\alpha} + \delta \mathbf{e}_n)$ and $\mathbf{e}_n$ denotes a vector of all zeros except a 1 at entry n.

Essentially, this will give the new α_n that guarantees the maximum increase in f, with all other α_n's fixed at their current value. Derive the expression for δ_* and give a sketch of the overall co-ordinate ascent algorithm.
Note that your expression for δ_* should be such that the constraint $0 \leq \alpha_n \leq C$ is maintained.

Solution:

The soft-margin linear SVM dual objective to compute weights can be stated as

$$\arg \max_{0 \leq \alpha \leq C} f(\alpha),$$

where $f(\alpha) = \alpha^T.\mathbf{1} - \frac{1}{2}\alpha^T \mathbf{G} \alpha$. The gradient-ascent step that we consider for α_n is:

$$\alpha_n = \alpha_n + \delta_*; \text{ where } \delta_* := \arg\max_\delta f(\alpha + \delta \mathbf{e}_n)$$

First, we will have to compute δ_* in order to apply the ascent step.

$$f(\alpha + \delta \mathbf{e}_n) = (\alpha + \delta \mathbf{e}_n)^T.\mathbf{1} - \frac{1}{2}(\alpha + \delta \mathbf{e}_n)^T \mathbf{G} (\alpha + \delta \mathbf{e}_n)$$

$$\therefore f(\alpha + \delta \mathbf{e}_n) = \alpha^T.\mathbf{1} + \delta \mathbf{e}_n^T.\mathbf{1} - \frac{1}{2}\alpha^T \mathbf{G} \alpha - \frac{1}{2}\alpha^T \mathbf{G} \delta \mathbf{e}_n - \frac{1}{2}\delta \mathbf{e}_n^T \mathbf{G} \alpha - \frac{1}{2}\delta \mathbf{e}_n^T \mathbf{G} \delta \mathbf{e}_n$$

$$f(\alpha + \delta \mathbf{e}_n) = f(\alpha) + \delta\left(\mathbf{e}_n^T.\mathbf{1} - \alpha^T \mathbf{G} \mathbf{e}_n\right) - \frac{1}{2}\delta^2 \mathbf{e}_n^T \mathbf{G} \mathbf{e}_n$$

Taking partial derivative w.r.t. δ, and equating it to 0 , we get

$$\frac{\partial f(\alpha + \delta \mathbf{e}_n)}{\partial \delta} = \left(\mathbf{e}_n^T \cdot \mathbf{1} - \alpha^T \mathbf{G} \mathbf{e}_n\right) - \delta \mathbf{e}_n^T \mathbf{G} \mathbf{e}_n$$

$$\Rightarrow \delta_* = \frac{\mathbf{e}_n^T \cdot \mathbf{1} - \alpha^T \mathbf{G} \mathbf{e}_n}{\mathbf{e}_n^T \mathbf{G} \mathbf{e}_n} \tag{5}$$

Note that we want to ensure that $0 \leq \alpha_n^{(t)} \leq C$, thus if the updated value for α_n becomes $\geq C$, we will just set it to C and if it goes less 0, we will just set it to 0. Thus, the update for α_n will become as follows:

$$\alpha_n^{(t)} = \begin{cases} 0, & \text{if } \alpha_n^{(t-1)} + \delta_* < 0 \\ C, & \text{if } \alpha_n^{(t-1)} + \delta_* > C \\ \alpha_n^{(t-1)} + \delta_* & \text{otherwise} \end{cases} \tag{6}$$

Coordinate ascent algorithm sketch

1. Inititalize $\alpha = \alpha^{(0)} = \left[\alpha_1^{(0)}, \ldots, \alpha_N^{(0)}\right]$
2. Randomly choose $n \in \{1, 2, \ldots, N\}$. Compute δ_* from equation 5.
3. Update $\alpha_n^{(t)}$ in terms of $\alpha_n^{(t-1)}$ and δ_* as in equation 6.
4. If not converged, go to step 2.

Problem 28. (Separating Convex Hulls)

Given a set of data points $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N$, we define the convex hull to be the set of all points $\boldsymbol{x}$ given by $\boldsymbol{x} = \sum_n \alpha_n \boldsymbol{x}_n$ where $\alpha_n \geq 0$ and $\sum_n \alpha_n = 1$ (Intuitively, the convex hull of a set of points in the solid region that they enclose). Consider a second set of points $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_M$ together with their corresponding convex hull. Show that the set of $\boldsymbol{x}$ and the set of $\boldsymbol{y}$ are linearly separable if and only if the convex hulls do not intersect.

Solution:

Let C_x and C_y be the convex hulls corresponding to points $X = \{\mathbf{x}_i\}_{i=1}^N$ and $Y = \{\mathbf{y}_i\}_{i=1}^N$ respectively.

$$C_X := \left\{ x = \sum_{n=1}^{N} \alpha_n x_n : \alpha_i \geq 0, \sum_i \alpha_i = 1 \right\}$$
$$C_Y := \left\{ y = \sum_{n=1}^{N} \beta_n y_n : \beta_i \geq 0, \sum_i \beta_i = 1 \right\}$$

Definition 1. *The sets of points X and Y are said to be linearly seperable if $\exists$ a line (hyperplane in high dimensional space) $L := \left\{ \mathbf{x} \in \mathbb{R}^D : m^T\mathbf{x} + b = 0 \right\}$ such that $m^T\mathbf{x}_i + b \geq 0$, and $m^T\mathbf{y}_i + b < 0 \; \forall \; i = 1, 2, \ldots, N$.*

We have to show that X and Y are linearly separable iff $C_X \cap C_Y = \phi$
($\Rightarrow$) Assume that X and Y are linearly separable. Suppose for contradiction that $C_X \cap C_Y \neq \phi$. This implies that $\exists z \in C_X \cap C_Y$. Thus, by definition,

$$z = \sum_{n=1}^{N} \alpha_n x_n = \sum_{n=1}^{N} \beta_n y_n$$

where $\alpha_i \geq 0, \sum_i \alpha_i = 1, \beta_i \geq 0, \sum_i \beta_i = 1$. Since, X and Y are linearly separable, there exists a line L satisfying $m^T\mathbf{x}_i + b \geq 0$, and $m^T\mathbf{y}_i + b < 0 \; \forall \; i = 1, 2, \ldots, N$.

Now, since $\alpha_i \geq 0, \; \forall \; i$,

$$\alpha_i m^T\mathbf{x}_i + \alpha_i b \geq 0, \forall i = 1, 2, \ldots, N$$
$$\therefore \sum_{n=1}^{N} \alpha_n m^T\mathbf{x}_n + \sum_{n=1}^{N} \alpha_n b = \sum_{n=1}^{N} \alpha_n m^T\mathbf{x}_n + b.1 = m^T z + b \geq 0 \tag{7}$$

Similarly, since $\beta_i m^T\mathbf{y}_i + \beta_i b < 0 \; \forall \; i = 1, 2, \ldots, N$,

$$\therefore \sum_{n=1}^{N} \beta_n m^T\mathbf{x}_n + \sum_{n=1}^{N} \beta_n b = \sum_{n=1}^{N} \beta_n m^T\mathbf{x}_n + b.1 = m^T z + b < 0 \tag{8}$$

The equations 7 and 8 present a contradiction. Thus, we have $C_X \cap C_Y = \phi$.
($\Leftarrow$) To show the converse, assume that $C_X \cap C_Y = \phi$. To show that X and Y are linearly separable i.e. we have to find L satisfying definition 1 Define

$$d := \min_{x \in C_X, y \in C_Y} \|x - y\|^2$$

Note, since C_X and C_Y are closed and bounded, the quantity d exists, and suppose it achieves the minima at $x_o \in C_X, y_o \in C_Y$. Also $d > 0$ since $C_X \cap C_Y = \phi$. Let L be the perpendicular bisector of the line joining x_o and y_o. It is trivial to observe that L is a linear separator of X and Y. Hence, we are done.

Problem 29. (A Circular Definition)

Consider a logistic regression model $p\left(y_n \mid \boldsymbol{w}_n, \boldsymbol{w}\right) = \frac{1}{1+\exp(-y_n \boldsymbol{w}^\top \boldsymbol{x}_n)}$, with a zero-mean Gaussian prior $p(\boldsymbol{w}) = \mathcal{N}\left(\mathbf{0}, \lambda^{-1}\mathbf{I}\right)$. Note that this loss function for logistic regression assumes $y_n \in \{-1, +1\}$ instead of $\{0, 1\}$. Show that the MAP estimate for $\boldsymbol{w}$ can be written as $\boldsymbol{w} = \sum_{n=1}^N \alpha_n y_n \boldsymbol{x}_n$ where each α_n itself is a function of $\boldsymbol{w}$. Based on the expression of α_n, you would see that it has a precise meaning. Briefly, state what α_n means, and also briefly explain why the result $\boldsymbol{w} = \sum_{n=1}^N \alpha_n y_n \boldsymbol{x}_n$ makes sense for this model.

Solution:

Consider a logistic regression model with Gaussian prior.

$$p\left(y_n \mid \mathbf{x_n}, \mathbf{w}\right) = \frac{1}{1 + \exp\left(-y_n \mathbf{w}^T \mathbf{x_n}\right)}, \quad p(\mathbf{w}) = \mathcal{N}\left(\mathbf{0}, \lambda^{-1}\mathbf{I}\right)$$

For MAP estimation, we need to solve the following maximization problem:

$$\hat{\mathbf{w}}_{MAP} = \arg\min_{w} \mathcal{L}(\mathbf{w}) = \arg\min_{w} \left(- \sum_{n=1}^{N} \log\left(p\left(y_n \mid \mathbf{x_n}, \mathbf{w}\right)\right) - \log(p(\mathbf{w})) \right)$$

$$\log\left(p\left(y_n \mid \mathbf{x_n}, \mathbf{w}\right)\right) = -\log\left(1 + \exp\left(-y_n \mathbf{w}^T \mathbf{x_n}\right), \ \log(p(\mathbf{w})) = -(D/2)\log(2\pi) + \log(\lambda) - (\lambda/2)\mathbf{w}^T\mathbf{w}\right.$$

$$\therefore \mathcal{L}(\mathbf{w}) = \sum_{n=1}^{N} \log\left(1 + \exp\left(-y_n \mathbf{w}^T \mathbf{x_n}\right)\right) + \frac{\lambda}{2}\mathbf{w}^T\mathbf{w}$$

Computing the partial derivative w.r.t $\mathbf{w}$, and equating to 0, we get

$$\frac{\partial \mathcal{L}}{\partial \mathbf{w}} = \sum_{n=1}^{N} \frac{-y_n \mathbf{x}_n}{1 + \exp\left(-y_n \mathbf{w}^T \mathbf{x_n}\right)} + \lambda \mathbf{w} \Rightarrow \mathbf{w} = \sum_{n=1}^{N} y_n \alpha_n \mathbf{x}_n,$$

$$\text{where} \ \ \alpha_n(\mathbf{w}) = \frac{1}{\lambda\left(1 + \exp\left(-y_n \mathbf{w}^T \mathbf{x_n}\right)\right)} = \frac{p\left(y_n \mid \mathbf{x}_n, \mathbf{w}\right)}{\lambda}.$$

Thus, the MAP estimate tries to weigh examples based on the prediction probability of each given example. It will ignore the examples on which the predicted probability is low ($p\left(y_n \mid \mathbf{x_n}, \mathbf{w}\right) \to 0$) and consider only those examples that have this predicted probability high much like the support vectors in SVM. The hyperparameter λ will decide the amount of regularization as usual.

Problem 30. (Softmax and Variants)

Consider N training examples $\{\boldsymbol{x}_n, y_n\}_{n=1}^{N}$ where $\boldsymbol{x}_n \in \mathbb{R}^D$, and $y_n \in \{1, \ldots, K\}$. Suppose we wish to use this data to learn the multiclass logistic (or "softmax") regression model which defines

$$p\left(y_n = k \mid \boldsymbol{x}_n, \mathbf{W}\right) = \mu_{nk} = \frac{\exp\left(\boldsymbol{w}_k^\top \boldsymbol{w}_n\right)}{\sum_{\ell=1}^{K} \exp\left(\boldsymbol{w}_\ell^\top \boldsymbol{w}_n\right)},$$

where μ_{nk} is the predicted probability of $y_n = k$. Derive the MLE solution for $\mathbf{W} = [\boldsymbol{w}_1, \boldsymbol{w}_2, \ldots, \boldsymbol{w}_K]$. You would notice that, just like logistic regression, there is no closed-form solution for $\mathbf{W}$. So you will need to write down the log-likelihood, take its derivative w.r.t. each column $\boldsymbol{w}_k$ of $\mathbf{W}$ to compute the gradient, and derive the gradient descent (GD) update rule for each $\boldsymbol{w}_k$. Show the basic steps of your derivation and write down the final expression for the GD update of each $\boldsymbol{w}_k$. Assume a fixed learning rate $\eta = 1$ for simplicity.

Next, consider the stochastic gradient descent (SGD) update for the same model where each iteration takes a randomly chosen example $(\boldsymbol{x}_n, y_n)$ to update $\mathbf{W}$. Write down the expressions for these SGD updates and the overall sketch of the corresponding SGD algorithm.

Finally consider a special case of the above SGD algorithm for this model, where the predicted "soft" class probabilities μ_{nk} are replaced by hard class assignments, i.e., $\mu_{nk} = 1$ for $k = \arg\max_\ell \{\mu_{n\ell}\}_{\ell=1}^{K}$, and $\mu_{nk'} = 0, \forall k' \neq k$. Write down the expressions for the SGD update in this case and the overall sketch of the SGD algorithm. How are these updates different from the previous case where you used soft probabilities μ_{nk}?

Solution:

Consider N training examples $\{x_n, y_n\}_{n=1}^{N}$ where $x_n \in \mathbb{R}^D, y_n \in \mathbb{R}^K$. Consider the softmax regression model for multi-class classification:

$$p\left(y_n = k \mid x_n, W\right) = \mu_{nk} = \frac{\exp\left(w_k^T x_n\right)}{\sum_{l=1}^{K} \exp\left(w_l^T x_n\right)}$$

The MLE objective can be stated as $\hat{W}_{MLE} = \arg\min_W -NLL(W)$. Let us work for each column w_k of W:

$$\frac{-\partial NLL(W)}{\partial w_k} = -\sum_{n=1}^{N} \frac{\partial \log\left(p\left(y_n = k \mid x_n, W\right)\right)}{\partial w_k}$$

$$\log\left(p\left(y_n = k \mid x_n, W\right)\right) = w_k^T x_n - \log\left(\sum_{l=1}^{K} \exp\left(w_l^T x_n\right)\right)$$

$$\therefore \frac{\partial \log\left(p\left(y_n = k \mid x_n, W\right)\right)}{\partial w_k} = x_n - \mu_{nk} x_n$$

$$\therefore \frac{-\partial NLL(W)}{\partial w_k} = \sum_{n=1}^{N} x_n\left(\mu_{nk} - 1\right) = X^T \text{ Diagonal}\left(\left[\mu_{1k} - 1, \mu_{2k} - 1, \ldots, \mu_{Nk} - 1\right]\right)$$

Thus, no closed-form solution for w_k can be computed from this equation. We can write the gradient descent update for w_k with $\eta = 1$ as follows:

$$w_k^{(t+1)} = w_k^{(t)} - \sum_{n=1}^{N} x_n\left(\mu_{nk}^{(t)} - 1\right)$$

Similarly, the stochastic gradient descent update will turn out to be:

$$w_k^{(t+1)} = w_k^{(t)} - x_n\left(\mu_{nk}^{(t)} - 1\right)$$

Consider the case of 'hard' assignments, i.e., let $k = \arg\max_k \mu_{nk}$ and $\mu_{nk'} = 0,\ \forall\ k' \neq k$. The SGD update will simply be reduced to

$$w_k^{(t+1)} = w_k^{(t)} - x_n\left(\mu_{nk}^{(t)} - 1\right) = \begin{cases} w_k^{(t)} & \text{for } p\left(y_n = k \mid x_n, W\right) = 1 \\ w_k^{(t)} + x_n & \text{for } p\left(y_n = k \mid x_n, W\right) = 0 \end{cases}$$

Thus, we simply do NOT update the weight vector w_k if we encounter an example (x_n, y_n) such that the current prediction is correct, i.e. whose label is k. Otherwise, we make the weight vector w_k move in the direction of x_n. The SGD algorithm sketch for this case is presented in Algorithm 1

Algorithm 1 Multi-class Stochastic Gradient Descent

```
Choose initial $w_k = w_k^{(0)}, \forall k \in 1, 2, \ldots, K$.
  for $k \in 1, 2, \ldots, K$ do
    Keeping $w_l$, for $l \neq k$ fixed, update $w_k$
    for $s$ iterations until convergence do
      if $p(y_n|x_n, W^{(t)}) = 1$ then
        $w_k^{(t+1)} = w_k^{(t)}$
      else
        $w_k^{(t+1)} = w_k^{(t)} + x_n$
      end if
    end for
  end for
```

Problem 31. (Estimating a Gaussian when Data is Missing)

Suppose we have collected N observations $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$ using a sensor. Let us assume each $\boldsymbol{x} \in \mathbb{R}^D$ as generated from a Gaussian distribution $\mathcal{N}(\mu, \Sigma)$. We would like to estimate the mean and covariance of this Gaussian. However, suppose the sensor was faulty and each $\boldsymbol{x}_n$ could only have part of it as observed (think of a blacked-out image). Denote $\boldsymbol{x}_n = \left[\boldsymbol{x}_n^{obs}, \boldsymbol{x}_n^{\text{miss}}\right]$ where $\boldsymbol{x}_n^{obs}$ and $\boldsymbol{x}_n^{\text{miss}}$ denote the observed and missing parts, respectively, of $\boldsymbol{x}_n$. We only get to see $\boldsymbol{x}_n^{obs}$. Note that different observations could have different parts as missing (e.g., different images may have different sets of pixels as missing), so the indices of the observed/missing entries of the vector $\boldsymbol{x}_n$ may be different for different n.

Your goal is to develop an EM algorithm that gives maximum likelihood estimates of μ and Σ given this partially observed data. In particular, in the EM setting, you will treat each $\boldsymbol{x}_n^{\text{miss}}$ as a latent variable and estimate its conditional distribution

$p\left(\boldsymbol{x}_n^{\text{miss }} \mid \boldsymbol{x}_n^{obs}, \mu, \Sigma\right)$, given the current estimates μ and Σ of the parameters. In the M step, you will re-estimate μ and Σ and will alternate between E and M steps until convergence.

Solve the following:

1. The expression for $p\left(\boldsymbol{x}_n^{\text{miss }} \mid \boldsymbol{x}_n^{obs}, \mu, \Sigma\right)$
2. The expected CLL for this model
3. The update equations for μ and Σ.

Also clearly write down all the steps of the EM algorithm in this case, with appropriate equations.

Hint: For this problem, you may find it useful to use the result that if $\boldsymbol{x}=\left[\boldsymbol{x}_a, \boldsymbol{x}_b\right]$ is Gaussian then $p\left(\boldsymbol{x}_a \mid \boldsymbol{x}_b\right)$ is also Gaussian.

Solution:

Let $X=\left\{\mathbf{x}_1, \mathbf{x}_2, .., \mathbf{x}_N\right\}, \Theta=\{\mu, \Sigma\}$ and

$$p\left(x_n \mid \Theta\right)=\mathcal{N}\left(x_n \mid \Theta\right)$$

Part 1: Estimating the conditional given current parameter estimates

Let's say we have Θ as our current parameter estimates. Let us try to estimate the missing part of data for a point x_n. We have

$$p\left(\left[x_n^{obs}, x_n^{miss}\right] \mid \Theta\right)=\mathcal{N}\left(\left[x_n^{obs}, x_n^{miss}\right] \mid \Theta\right)$$

Based on the missing and observed part of this data point, let us denote the parameters as follows:

$$\mu=\left(\begin{array}{c}\mu_{\text{miss}} \\ \mu_{obs}\end{array}\right)=\left(\begin{array}{c}\mu_1 \\ \mu_2\end{array}\right), \Sigma=\left(\begin{array}{cc}\Sigma_{11} & \Sigma_{12} \\ \Sigma_{21} & \Sigma_{22}\end{array}\right)$$

Note that this miss-observed split may be different for different examples. This also makes intuitive sense, for example, if for the first example the first two features are missing, then its estimate should be based on the means of these features of other examples and also covariances with other features. For an example with a different set of missing features, the estimates will be different. Since a conditional of a Gaussian is a Gaussian, we have

$$\begin{gathered}p\left(x_n^{\text{miss }} \mid x_n^{obs}, \Theta\right)=\mathcal{N}\left(x_n^{\text{miss }} \mid x_n^{obs},\left\{\mu_{miss \mid obs}, \Sigma_{miss \mid obs}\right\}\right) \text { where } \\ \mu_{miss \mid obs}=\mu_1+\Sigma_{12} \Sigma_{22}^{-1}\left(\mathbf{x}_n^{obs}-\mu_2\right) \\ \Sigma_{\text{miss } \mid obs}=\Sigma_{11}-\Sigma_{12} \Sigma_{22}^{-1} \Sigma_{21}\end{gathered}$$

Part 2: Expected CLL

First, I will compute the CLL as follows and then take the expectation:

$$\begin{gathered}CLL=\sum_{n=1}^N \log p\left(\mathbf{x}_n \mid \Theta=\{\mu, \Sigma\}\right) \\ CLL=\sum_{n=1}^N \log p\left(\left[\mathbf{x}_n^{obs}, \mathbf{x}_n^{miss}\right] \mid \Theta=\{\mu, \Sigma\}\right) \\ \mathbb{E}[CLL]=\sum_{n=1}^N \mathbb{E}\left[\log p\left(\left[\mathbf{x}_n^{obs}, \mathbf{x}_n^{\text{miss }}\right] \mid \mu, \Sigma\right)\right] \\ \mathbb{E}[CLL]=\text { constant }+\frac{N}{2}\left(\log |\Sigma|^{-1}\right)-\frac{1}{2} \operatorname{Trace}\left(\Sigma^{-1} \mathbb{E}[S]\right) \\ S=\sum_{n=1}^N\left(\mathbf{x}_n-\mu\right)\left(\mathbf{x}_n-\mu\right)^T\end{gathered}$$

We will need to compute the expectation of S over the posterior. Note that we will have the following expectation values:

$$\mathbb{E}\left[\mathbf{x}_n\right]=\left(\begin{array}{c}\mathbb{E}\left[\mathbf{x}_n^{obs}\right] \\ \mathbb{E}\left[\mathbf{x}_n^{miss}\right]\end{array}\right)=\left(\begin{array}{c}\mathbf{x}_n^{obs} \\ \mathbb{E}\left[\mathbf{x}_n^{miss}\right]\end{array}\right) \tag{9}$$

$$\mathbb{E}\left[\mathbf{x}_n\mathbf{x}_n^T\right]=\left(\begin{array}{cc}\mathbf{x}_n^{obs}\mathbf{x}_n^{(obs)T} & \mathbf{x}_n^{obs}\mathbb{E}\left[\mathbf{x}_n^{miss}\right]^T \\ \mathbb{E}\left[\mathbf{x}_n^{miss}\right]\left(\mathbf{x}_n^{obs}\right)^T & \mathbb{E}\left[\mathbf{x}_n^{\text{miss}}\ \left(\mathbf{x}_n^{miss}\right)^T\right]\end{array}\right) \tag{10}$$

$$\mathbb{E}[S]=\sum_{n=1}^{N}\left\{\mathbb{E}\left[\mathbf{x}_n\mathbf{x}_n^T\right]+\mu\mu^T-2\mu\mathbb{E}\left[\mathbf{x}_n\right]^T\right\} \tag{11}$$

Note that from part 1, we have got the conditional distribution for $\mathbf{x}_n^{\text{miss}}$, thus we will have the mean $\mathbb{E}\left[\mathbf{x}_n^{\text{miss}}\ \right]=\mu_{\text{miss obs}}$. The expected CLL computation is over the conditional distribution. Thus, we can plug the value of $\mathbb{E}\left[\mathbf{x}_n^{\text{miss}}\ \right]$ in the above equations to obtain the expected CLL. The only computation that will remain is

$$\mathbb{E}\left[\mathbf{x}_n^{miss}\left(\mathbf{x}_n^{miss}\right)^T\right]=\mathbb{E}\left[\mathbf{x}_n^{miss}\right]\mathbb{E}\left[\mathbf{x}_n^{miss}\right]^T+\operatorname{cov}\left(\mathbf{x}_n^{miss}\right)=\mathbb{E}\left[\mathbf{x}_n^{miss}\right]\mathbb{E}\left[\mathbf{x}_n^{miss}\right]^T+\Sigma_{11}$$

Part 3: Estimation of parameters: M step

Let us denote $\mathbb{E}[CLL]=\mathcal{L}(\mu,\Sigma)$. In order to obtain the updates for these paramteres, we will set the partial derivatives to 0 treating terms involving $\mathbb{E}\left[\mathbf{x}_n^{miss}\right]$ as constant.

$$\frac{\partial\mathcal{L}}{\partial\mu}=-\frac{1}{2}\left(\Sigma^{-1}\right)^T\frac{\partial\mathbb{E}[S]}{\partial\mu}$$

$$\frac{\partial\mathbb{E}[S]}{\partial\mu}=\sum_{n=1}^{N}\left\{\frac{\partial\mu\mu^T}{\partial\mu}-2\frac{\partial\left(\mu\mathbb{E}\left[\mathbf{x}_n\right]^T\right)}{\partial\mu}\right\}$$

$$\frac{\partial\mathbb{E}[S]}{\partial\mu}=N\frac{\partial\mu\mu^T}{\partial\mu}-2\sum_{n=1}^{N}\frac{\partial\left(\left[\mu\mathbf{x}_n^{obs(T)}\mu\mathbb{E}\left[\mathbf{x}_n^{\text{miss}}\ \right]^T\right]\right)}{\partial\mu}$$

$$\frac{\partial\mathcal{L}}{\partial\Sigma}=\frac{N|\Sigma|}{2}\frac{\partial|\Sigma|^{-1}}{\partial\Sigma}-\frac{1}{2}\frac{\partial\operatorname{Trace}\left(\Sigma^{-1}\mathbb{E}[S]\right)}{\partial\Sigma}$$

EM Algorithm steps:

1. Initialize $\Theta=\Theta^{(0)}=\left\{\mu^{(0)},\Sigma^{(0)}\right\}$, set $t=1$
2. E step: Now, assuming the values of $\Theta^{(t-1)}$, compute the expected CLL using expression for $\mathbb{E}[S]$ of equation 11 which in turn needs the equations 9 and 10.
3. **M** step: Now, update the values for $\mu^{(t)}$ and $\Sigma^{(t)}$ using the equations in part 3. Update $t=t+1$
4. Repeat steps 2-4 until convergence.

Problem 32. (Semi-supervised Classification)

Consider learning a generative classification model for K-class classification with Gaussian class-conditionals $\mathcal{N}\left(\boldsymbol{x}\mid\mu_k,\Sigma_k\right)$, $k=1,\ldots,K$ with class marginals $p(y=k)=\pi_k$. However, unlike traditional generative classification, in this setting, we are given N labeled examples $\left\{\left(\boldsymbol{x}_n,y_n\right)\right\}_{n=1}^{N}$ and an additional M unlabeled examples $\left\{\boldsymbol{x}_{N+1},\ldots,\boldsymbol{x}_{N+M}\right\}$. Design an EM algorithm to estimate all the unknowns of this model and clearly write down the expressions required in each step of the EM algorithm.

Solution:

Let us fix the following notation: $X_1=\left\{\left(\mathbf{x}_n,y_n\right)\right\}_{n=1}^{N}, X_2=\left\{x_m\right\}_{m=N+1}^{N+M}$ and let $Z_2=\left\{z_m\right\}_{m=N+1}^{N+M}$ be the latent variables. The set of parameters is $\Theta=\left\{\mu_k,\Sigma_k,\pi_k\right\}_{k=1}^{K}$. We assume that all the examples, whether with known labels or not are i.i.d sampled. We have the conditional distribution as follows:

$$p\left(z_m = k \mid x_m, X_1, \Theta\right) = p\left(z_m = k \mid x_m, \Theta\right) = \frac{p\left(x_m \mid z_m = k, \Theta\right) p\left(z_m = k \mid \Theta\right)}{p\left(x_m \mid \Theta\right)} = \frac{p\left(x_m \mid z_m = k, \Theta\right) p\left(z_m \mid \Theta\right)}{\sum_{l=1}^{K} p\left(x_m, z_m = l \mid \Theta\right)}$$

Let $\gamma_{mk} = p\left(z_m = k \mid x_m, X_1, \Theta\right) = \dfrac{\pi_k \mathcal{N}\left(x_m \mid \mu_k, \Sigma_k\right)}{\sum_{l=1}^{K} \pi_l \mathcal{N}\left(x_m \mid \mu_l, \Sigma_l\right)}$

$$\gamma_{mk} = \frac{\pi_k \mathcal{N}\left(x_m \mid \mu_k, \Sigma_k\right)}{\mathcal{N}\left(\sum_l \pi_l \mu_l, \right)} \tag{12}$$

Expected CLL

Let us fix the notation to be (as usual): $z_{mk} = 1$ iff $z_m = k$ and $y_{nk} = 1$ iff $y_n = k$. Now, the CLL will be computed as follows, note the assumption of IID.

$$CLL = \log p\left(X_1, (X_2, Z_2) \mid \Theta\right) = \log p\left(X_2, Z_2 \mid X_1, \Theta\right) + \log p\left(X_1 \mid \Theta\right) = \log p\left(X_2, Z_2 \mid \Theta\right) + \log p\left(X_1 \mid \Theta\right)$$

Now, we have

$$\log p\left(X_1 \mid \Theta\right) = \sum_{n=1}^{N} \sum_{k=1}^{K} y_{nk}\left(\log \pi_k + \log \mathcal{N}\left(x_n \mid y_n = k, \mu_k, \Sigma_k\right)\right)$$

$$\log p\left(X_2, Z_2 \mid \Theta\right) = \sum_{m=N+1}^{N+M} \sum_{k=1}^{K} z_{mk}\left(\log \pi_k + \log \mathcal{N}\left(x_m \mid z_m = k, \mu_k, \Sigma_k\right)\right)$$

Now, notice that for computing the expected CLL, we will need the expectation over the posterior: $\mathbb{E}\left[z_{nk}\right] = 1 * p\left(z_{nk} = 1\right) + 0 * p\left(z_{nk} = 0\right) = p\left(z_{nk} = 1\right) = \gamma_{mk}$.

$$\mathbb{E}[CLL] = \sum_{n=1}^{N} \sum_{k=1}^{K} y_{nk}\left(\log \pi_k + \log \mathcal{N}\left(x_n \mid y_n = k, \mu_k, \Sigma_k\right)\right) + \sum_{m=N+1}^{N+M} \sum_{k=1}^{K} \gamma_{mk}\left(\log \pi_k + \log \mathcal{N}\left(x_m \mid z_m = k, \mu_k, \Sigma_k\right)\right) \tag{13}$$

Updating the paramters

We have the following objective.

$$\hat{\Theta} = \arg\max_{\Theta} \mathbb{E}[CLL]$$

It basically boils down to solving the problem of Gaussian generative classification with known labels:

$$\mathbb{E}[CLL] = \sum_{n=1}^{N+M} \sum_{k=1}^{K} t_{nk}\left(\log \pi_k + \log \mathcal{N}\left(x_n \mid \mu_k, \Sigma_k\right)\right),$$

where $t_{nk} = y_{nk}, \forall n \in \{1, 2, \ldots, N\}$ and $t_{nk} = \gamma_{nk}, \forall n \in \{N+1, N+2, \ldots, N+M\}$. Thus the parameters will become:

$$\hat{\pi}_k = \frac{\sum_{n=1}^{N+M} t_{nk}}{N+M} = \frac{\sum_{n=1}^{N} y_{nk} + \sum_{m=N+1}^{N+M} \gamma_{nk}}{N+M} = \frac{N_k + M_k}{N+M} \tag{14}$$

$$\hat{\mu}_k = \frac{1}{N_k + M_k} \sum_{n=1}^{N+M} t_{nk} \mathbf{x}_n \tag{15}$$

$$\hat{\Sigma}_k = \frac{1}{N_k + M_k} \sum_{n=1}^{N+M} t_{nk}\left(\mathbf{x}_n - \hat{\mu}_k\right)\left(\mathbf{x}_n - \hat{\mu}_k\right)^T \tag{16}$$

The EM algorithm

1. Initialize $\Theta = \Theta^{(0)} = \left\{\pi_k^{(0)}, \mu_k^{(0)}, \Sigma_k^{(0)}\right\}_k$, set $t = 1$

2. E step: Now, assuming the values of $\Theta^{(t-1)}$, compute the expected CLL from equation 13 using expression for $\mathbb{E}[z_{mk}] = \gamma_{mk}$ (See equation 12).

3. M step: Now, update the values for $\pi_k^{(t)}, \mu_k^{(t)}$ and $\Sigma_k^{(t)}$ using the equations 14, 15, and 16 respectively. Update $t = t + 1$.

4. Repeat steps 2-4 until convergence.

Problem 33. (Latent Variable Models for Supervised Learning)

Consider learning a regression model given training data $\{(\boldsymbol{x}_n, y_n)\}_{n=1}^N$, with $\boldsymbol{x}_n \in \mathbb{R}^D$ and $y_n \in \mathbb{R}$. Let us give a small twist to the standard probabilistic linear model for regression that we have seen in class. In particular, we will be introducing a latent variable $\boldsymbol{z}_n$ with each training example $(\boldsymbol{x}_n, y_n)$. The generative story would now be as follows

$$\begin{aligned} z_n &\sim \text{multinoulli}\left(\pi_1, \ldots, \pi_K\right) \\ y_n &\sim \mathcal{N}\left(\boldsymbol{w}_{z_n}^\top \boldsymbol{x}_n, \beta^{-1}\right) \end{aligned}$$

Note that the model for the responses y_n is still discriminative since the inputs are not being modeled. The latent variables are $\mathbf{Z} = (z_1, \ldots, z_N)$ and the global parameters are $\Theta = \{(\boldsymbol{w}_1, \ldots, \boldsymbol{w}_K), (\pi_1, \ldots, \pi_K)\}$.

1. Give a brief explanation (max. 5 sentences) of what the above model is doing and why you might want to use it instead of the standard probabilistic linear model which models each response as $y_n \sim \mathcal{N}\left(\boldsymbol{w}^\top \boldsymbol{x}_n, \beta^{-1}\right)$.

2. Derive an ALT-OPT algorithm to estimate $\mathbf{Z}$ and (MLE of) Θ, and clearly write down each step's update equations. For $\mathbf{Z}$, you must give the update equation for each individual latent variable $z_n, n = 1, \ldots, N$. Likewise, for Θ, you must give the update equation for each $\boldsymbol{w}_k, k = 1, \ldots, K$, and $\pi_k, k = 1, \ldots, K$. Also, what will be the update of each z_n if $\pi_k = 1/K, \forall k$. Give a brief intuitive explanation (max 1-2 sentences) as to what this update does.

3. Derive an expectation-maximization (EM) algorithm to estimate $\mathbf{Z}$ and (MLE of) Θ, and clearly write down each step's update equations. Also show that, as $\beta \to \infty$, the EM algorithm reduces to ALT-OPT.

Solution:

Let the observed data be $\{(\mathbf{x}_n, y_n)\}_{n=1}^N$ and latent variables $Z = \{z_1, z_2, .., z_K\}$. The global set of parameters be given by $\Theta = \{(\mathbf{w}_1, \ldots, \mathbf{w}_K), (\pi_1, \ldots, \pi_K)\}$, we have

$$\begin{aligned} z_n &\sim \text{multinoulli}\left(\pi_1, .., \pi_K\right) \\ y_n &\sim \mathcal{N}\left(\mathbf{w}_{z_n}^T \mathbf{x}_n, \beta^{-1}\right) \end{aligned}$$

Part 1: Mixture of regression models

The model is basically fitting multiple regression models $\mathbf{w}_{z_n}$ on the input data. It is similar to doing multi-output regression. In simple probabilistic linear regression, we assume that all of the data points are sampled, i.i.d. from a linear space. However, as shown in Figure 4, for such kind of data, a mixture of multiple regression models can help.

Part 2: ALT-OPT Algorithm

Estimating the latent variables fixing the parameters

Suppose we have the optimal value of $\Theta = \hat{\Theta}$, then we can estimate z_n as follows:

$$\begin{aligned} \hat{z}_n &= \arg\max_k p\left(z_n = k \mid y_n, \mathbf{x}_n, \hat{\Theta}\right) = \arg\max_k \frac{p\left(y_n, z_n = k \mid \mathbf{x}_n, \hat{\Theta}\right)}{p\left(y_n \mid \mathbf{x}_n, \hat{\Theta}\right)} \\ &= \arg\max_k \frac{p\left(y_n \mid z_n = k, \mathbf{x}_n, \hat{\Theta}\right) p\left(z_n = k \mid \hat{\Theta}\right)}{\sum_{l=1}^K p\left(y_n \mid z_n = l, \mathbf{x}_n, \hat{\Theta}\right) p\left(z_n = l \mid \hat{\Theta}\right)} = \arg\max_k \frac{\pi_k \mathcal{N}\left(\mathbf{w}_k^T \mathbf{x}_n, \beta^{-1}\right)}{\sum_{l=1}^K \pi_l \mathcal{N}\left(\mathbf{w}_l^T \mathbf{x}_n, \beta^{-1}\right)} \end{aligned}$$

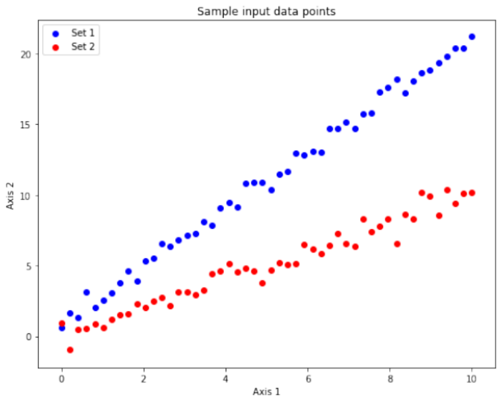


Figure 4: If the data looks somewhat like this, using multiple mixtures of linear regressions would be a better idea.

Let us denote the term

$$\frac{\pi_k \mathcal{N}\left(\mathbf{w}_k^T \mathbf{x}_n, \beta^{-1}\right)}{\sum_{l=1}^{K} \pi_l \mathcal{N}\left(\mathbf{w}_l^T \mathbf{x}_n, \beta^{-1}\right)} = \gamma_{nk}.$$

Notice that the denominator does not depend on k and can be dropped, but I am keeping it since the expression takes the form of probability. It will not matter for the maximization. Also, let's use the notation that $z_{nk} = 1$ iff $z_n = k$.

Estimating the parameters fixing the latent variables

Once we pick $\hat{z}_n$'s that maximize γ_{nk}'s, we can now keep them fixed and estimate parameters.

$$\hat{\Theta} = \arg\max_{\Theta} \mathcal{L}\left(\Theta, \{\hat{z}_n\}\right) = \arg\max_{\Theta} \sum_{n=1}^{N} \log p\left(y_n, \hat{z}_n \mid \mathbf{x}_n, \Theta\right)$$

$$\hat{\Theta} = \arg\max_{\Theta} \sum_{n=1}^{N} \sum_{k=1}^{K} \hat{z}_{nk} \left[\log \pi_k + \log \mathcal{N}\left(\mathbf{w}_k^T \mathbf{x}_n, \beta^{-1}\right)\right]$$

Note that we will also have to incorporate the constraint $\sum_k \pi_k = 1$. We can then solve this by constrained optimization. For π_k, since the term involving it is not coupled with any of the other parameters, we can solve for it independently. The problem for finding π_k is exactly the same as that in generative classification with known labels thus we have

$$\hat{\pi}_k = \frac{1}{N} \sum_{n=1}^{N} \hat{z}_{nk} = \frac{N_k}{N}$$

Now, to estimate the weights $\mathbf{w}_k$'s, we have

$$\{\hat{\mathbf{w}}_k\}_{k=1}^{K} = \arg\max_{\mathbf{w}_1,\ldots,\mathbf{w}_K} \sum_{n=1}^{N} \sum_{k=1}^{K} \hat{z}_{nk} \left(\mathbf{w}_k^T \mathbf{x}_n - y_n\right)^2$$

$$\{\hat{\mathbf{w}}_k\}_{k=1}^{K} = \arg\max_{\mathbf{w}_1,.,\mathbf{w}_K} \sum_{k=1}^{K} \left(\mathbf{y} - \mathbf{X}\mathbf{w}_k\right)^T Z_k \left(\mathbf{y} - \mathbf{X}\mathbf{w}_k\right) \text{ where } Z_k = \mathrm{Diag}\left(z_{1k}, .., z_{Nk}\right)$$

Note that these terms can be maximized independently since they do not depend upon each other. Thus, we get

$$\hat{\mathbf{w}}_k = \arg\max_{\mathbf{w}_k} \left(\mathbf{y} - \mathbf{X}\mathbf{w}_k\right)^T Z_k \left(\mathbf{y} - \mathbf{X}\mathbf{w}_k\right)$$

This is the same as solving a linear regression problem with examples being weighted by $\{z_{nk}\}$'s. Thus, the solution will be as follows:

$$\hat{\mathbf{w}}_k = \left[\mathbf{X}^T Z_k \mathbf{X}\right] \mathbf{X}^T Z_k \mathbf{y}$$

A Special case

For the case when $\pi_k = (1/K), \forall k = 1, 2, ..K$ implies that

$$\hat{z}_n = \arg\max_k \delta_{nk} = \arg\max_k \frac{\mathcal{N}\left(\mathbf{w}_k^T \mathbf{x}_n, \beta^{-1}\right)}{\sum_{l=1}^K \mathcal{N}\left(\mathbf{w}_l^T \mathbf{x}_n, \beta^{-1}\right)}$$

This kind of update ignores the number of points that can be explained by a single model $\mathbf{w}_k$. Thus, it will only consider the probability of x_n explained by these different models giving them equal weightage.

Part 3: EM Algorithm

A. Computing the posterior

We have already computed the conditional distribution for the latent variables in Part 2 as:

$$p\left(z_n = k \mid y_n, \mathbf{x}_n, \Theta\right) = \gamma_{nk} = \frac{\pi_k \mathcal{N}\left(\mathbf{w}_k^T \mathbf{x}_n, \beta^{-1}\right)}{\sum_{l=1}^K \pi_l \mathcal{N}\left(\mathbf{w}_l^T \mathbf{x}_n, \beta^{-1}\right)}$$

From the discrete distribution, we have the expectation as simply:

$$\mathbb{E}\left[z_n\right] = \sum_{k=1}^K k p\left(z_n = k \mid y_n, \mathbf{x}_n, \Theta\right) = \sum_{k=1}^K k \gamma_{nk}$$

$$\mathbb{E}\left[z_{nk}\right] = 1 * p\left(z_{nk} = 1\right) + 0 * p\left(z_{nk} = 0\right) = p\left(z_n = k\right) = \gamma_{nk} \tag{17}$$

B. Computing the expected CLL

Now, let us compute the expected CLL.

$$CLL = \log p(Y, Z \mid X, \Theta) = \sum_{n=1}^N \log p\left(y_n, z_n \mid \mathbf{x}_n, \Theta\right) = \sum_{n=1}^N \sum_{k=1}^K z_{nk}\left[\log \pi_k + \log \mathcal{N}\left(\mathbf{w}_k^T \mathbf{x}_n, \beta^{-1}\right)\right]$$

$$\mathbb{E}[CLL] = \sum_{n=1}^N \sum_{k=1}^K \mathbb{E}\left[z_{nk}\right]\left[\log \pi_k + \log \mathcal{N}\left(\mathbf{w}_k^T \mathbf{x}_n, \beta^{-1}\right)\right] \tag{18}$$

We can get $\mathbb{E}\left[z_{nk}\right]$ from equation 17, and thus the expected CLL. The rest of the procedure for maximizing the expected CLL will remain the same as in the previous part, except that instead of z_{nk}, we will be using γ_{nk}. Thus, we have

$$\hat{\pi}_k = \frac{1}{N} \sum_{n=1}^N \gamma_{nk}, \quad \hat{\mathbf{w}}_k = \left[\mathbf{X}^T \Gamma_k \mathbf{X}\right] \mathbf{X}^T \Gamma_k \mathbf{y}, \tag{19}$$

where $\Gamma_k = \text{Diag}\left(\gamma_{1k}, \gamma_{2k}, .., \gamma_{nk}\right)$.

The Algorithm

1. Initialize $\Theta = \Theta^{(0)} = \left\{\pi_k^{(0)}, \mathbf{w}_k^{(0)}\right\}$, set $t = 1$
2. E step: Now, assuming the values of $\Theta^{(t-1)}$, compute the expected CLL from equation 18 using expression for $\mathbb{E}\left[z_{nk}\right] = \gamma_{nk}$.
3. M step: Now, update the values for $\pi_k^{(t)}, \mathbf{w}_k^{(t)}$ using the equation 19. Update $t = t + 1$.
4. Repeat steps 2-4 until convergence.

The Particular case

We have to show that when $\beta \to \infty$, the EM algorithm boils down to the ALT-OPT algorithm. Basically, we want to show the following:

$$\gamma_{nk} = 1 \text{ iff } k_* = \arg\max_k \frac{\pi_k \mathcal{N}\left(\mathbf{w}_k^T \mathbf{x}_n, \beta^{-1}\right)}{\sum_{l=1}^K \pi_l \mathcal{N}\left(\mathbf{w}_l^T \mathbf{x}_n, \beta^{-1}\right)}$$

We have

$$\gamma_{nk}(\beta) = \frac{\pi_k \exp\left[-\beta^2 \left\|y_n - \mathbf{w}_k^T \mathbf{x}_n\right\|^2\right]}{\sum_{l=1}^K \pi_l \exp\left[-\beta^2 \left\|y_n - \mathbf{w}_l^T \mathbf{x}_n\right\|^2\right]}$$
$$\Rightarrow \gamma_{nk}(\beta) = \frac{1}{1 + \sum_{l \neq k} \frac{\pi_l}{\pi_k} \exp\left[-\beta^2 \left(\left\|y_n - \mathbf{w}_l^T \mathbf{x}_n\right\|^2 - \left\|y_n - \mathbf{w}_k^T \mathbf{x}_n\right\|^2\right)\right]}$$

Now, consider the case when $k \neq k_*$ as defined above, the term $\left\|y_n - \mathbf{w}_l^T \mathbf{x}_n\right\|^2 - \left\|y_n - \mathbf{w}_k^T \mathbf{x}_n\right\|^2 < 0, \forall l = k_*$, thus the exponent becomes ∞ making $\gamma_{nk} \to 0$. For $k = k_*$, clearly, all other terms will go to 0, and thus $\gamma_{nk_*} = 1$. Thus, the claim is proved, and hence we will have

$$\mathbb{E}\left[z_{nk}\right] = 1 \text{ only when } \hat{z}_{nk} = 1$$

This implies that the expectation step becomes the same as that of ALT-OPT, and so does maximization since the expected likelihood will be the same as the parameter update step in ALT-OPT.

Problem 34. (Probabilistic Formulation of Matrix Factorization with Side Information)

Consider an $N \times M$ rating matrix $\mathbf{X}$, where the rows represent the N users, and the columns represent the M items. We are also given some side information: for each user n, a feature vector $\boldsymbol{a}_n \in \mathbb{R}^{D_U}$, and for each item m, a feature vector $\boldsymbol{b}_m \in \mathbb{R}^{D_I}$. Let's model each entry of $\mathbf{X}$ as $p\left(X_{nm} \mid \boldsymbol{u}_n, \boldsymbol{v}_m, \theta_n, \phi_m\right) = \mathcal{N}\left(X_{nm} \mid \theta_n + \phi_m + \boldsymbol{u}_n^\top \boldsymbol{v}_m, \lambda_x^{-1}\right)$.

In this model, $\boldsymbol{u}_n \in \mathbb{R}^K$ and $\boldsymbol{v}_m \in \mathbb{R}^K$ represent the user n and item m latent factors, respectively. In addition, for each user n, we have a user-specific bias $\theta_n \in \mathbb{R}$ (bias regardless of the item being rated), and for each item m, we have an item-specific bias $\phi_m \in \mathbb{R}$ ("popularity" of the item, regardless is who has rated this item).

Assume Gaussian priors on the latent factors $\boldsymbol{u}_n \in \mathbb{R}^K$ and $\boldsymbol{v}_m \in \mathbb{R}^K : p\left(\boldsymbol{u}_n\right) = \mathcal{N}\left(\boldsymbol{u}_n \mid \mathbf{W}_u \boldsymbol{a}_n, \lambda_u^{-1} \mathbf{I}_K\right)$ and $p\left(\boldsymbol{v}_m\right) = \mathcal{N}\left(\boldsymbol{v}_m \mid \mathbf{W}_v \boldsymbol{b}_m, \lambda_v^{-1} \mathbf{I}_K\right)$. Note that the mean of the priors of the latent factors $\boldsymbol{u}_n$ and $\boldsymbol{v}_m$ depends on the given user and item features ($\boldsymbol{a}_n$ and $\boldsymbol{b}_m$, respectively) via a linear model with regression parameters matrices $\mathbf{W}_u \in \mathbb{R}^{K \times D_U}$ and $\mathbf{W}_v \in \mathbb{R}^{K \times D_I}$, respectively. You don't need to assume any priors on the bias parameters θ_n, ϕ_m, and the regression parameters $\mathbf{W}_u, \mathbf{W}_v$.

Assume $\Omega = \{(n, m)\}$ to denote the set of indices of the observed entries of $\mathbf{X}, \Omega_{r_n}$ to be the set of items rated by user n, and Ω_{c_m} to be the set of users who rated item m. Your goal is to estimate $\left\{\boldsymbol{u}_n, \theta_n\right\}_{n=1}^N, \left\{\boldsymbol{v}_n, \phi_m\right\}_{m=1}^M, \mathbf{W}_u$, and $\mathbf{W}_v$. Assume all other parameters to be known.

Write down the loss function for this model (this will be the negative of the MAP objective for the model) and use the ALT-OPT algorithm to derive the update equations for all the unknowns. The expressions must be in closed form (it is possible for this model).

Solution:

Consider an $N \times M$ rating matrix X, where the rows represent the N users, and the columns represent the M items. We are also given some side information: for each user n, a feature vector $\mathbf{a}_n \in \mathbb{R}^{D_U}$, and for each item m, a feature vector $\mathbf{b}_m \in \mathbb{R}^{D_I}$. Let $\mathbf{u}_n, \mathbf{v}_m \in \mathbb{R}^K$ represent the latent factors for user n and item m respectively. Let θ_n be user bias and ϕ_m be the popularity of the item. We have

$$p\left(X_{nm} \mid \mathbf{u}_n, \theta_n, \mathbf{v}_m, \phi_m\right) = \mathcal{N}\left(X_{nm} \mid \theta_n + \phi_m + \mathbf{u}_n^T \mathbf{v}_m, \lambda^{-1}\right)$$
$$p\left(\mathbf{u}_n\right) = \mathcal{N}\left(\mathbf{u}_n \mid \mathbf{W}_u \mathbf{a}_n, \lambda_u^{-1} \mathbf{I}_K\right)$$
$$p\left(\mathbf{v}_m\right) = \mathcal{N}\left(\mathbf{v}_m \mid \mathbf{W}_v \mathbf{b}_m, \lambda_v^{-1} \mathbf{I}_K\right)$$

Assume $\Omega = \{(n, m)\}$ to denote the set of indices of the observed entries of X, Ω_{r_n} to be the set of items rated by user n, and Ω_{c_m} to be the set of users who rated item m. Let $\Theta := \{\{(\mathbf{u}_n, \theta_n)\}, \{\mathbf{v}_m, \phi_m\}, \mathbf{W}_u, \mathbf{W}_v\}$
The MAP objective can be written as follows:

$$\hat{\Theta}_{MAP} = \arg\max_{\Theta}\{\log(p(X \mid \Theta)) + \log(p(\Theta))\}$$

$$\log(p(X \mid \Theta)) = \sum_{(n,m)\in\Omega} \log\left(p\left(X_{nm} \mid \Theta\right)\right) = -\frac{\lambda}{2} \sum_{(n,m)\in\Omega} \left(X_{nm} - \left(\theta_n + \phi_m + \mathbf{u}_n^T \mathbf{v}_m\right)\right)^2$$

$$\log(p(\Theta)) = \sum_{n=1}^{N} \log\left(p\left(\mathbf{u}_n\right)\right) + \sum_{m=1}^{M} \log\left(p\left(\mathbf{v}_m\right)\right) = -\frac{\lambda_u}{2} \sum_{n=1}^{N} \|\mathbf{u}_n - \mathbf{W}_u \mathbf{a}_n\|^2 - \frac{\lambda_v}{2} \sum_{m=1}^{M} \|\mathbf{v}_m - \mathbf{W}_v \mathbf{b}_m\|^2$$

Thus, the consolidated loss can be written as

$$\mathcal{L}(\Theta) = \tfrac{\lambda}{2} \textstyle\sum_{(n,m)\in\Omega} \left(X_{nm} - \left(\theta_n + \phi_m + \mathbf{u}_n^T \mathbf{v}_m\right)\right)^2 + \tfrac{\lambda_u}{2} \sum_{n=1}^{N} \|\mathbf{u}_n - \mathbf{W}_u \mathbf{a}_n\|^2 + \tfrac{\lambda_v}{2} \sum_{m=1}^{M} \|\mathbf{v}_m - \mathbf{W}_v \mathbf{b}_m\|^2.$$

Optimizing using ALT-OPT

- **Estimating latent variables and parameters for users:** Let us keep $\left\{\left\{\hat{\mathbf{v}}_m, \hat{\phi}_m\right\}, \hat{\mathbf{W}}_v\right\}$ fixed and estimate the remaining variables. We will only consider relevant terms in the loss function.

$$\mathcal{L}(\Theta) = \frac{\lambda}{2} \sum_{(n,m)\in\Omega} \left(X_{nm} - \left(\theta_n + \phi_m + \mathbf{u}_n^T \mathbf{v}_m\right)\right)^2 + \frac{\lambda_u}{2} \sum_{n=1}^{N} \|\mathbf{u}_n - \mathbf{W}_u \mathbf{a}_n\|^2$$

1. Estimating $\mathbf{u}_n$ keeping $\theta_n, \mathbf{W}_u$ fixed: Note that this will turn out to be simple a least-squares problem with the prior on $\mathbf{u}_n$ being non-zero mean. Thus, we can write the solution in closed form as follows:

$$\mathbf{u}_n = \left(\sum_{m\in\Omega_{r_n}} \mathbf{v}_m^T \mathbf{v}_m + \lambda_u \mathbf{I}_K\right)^{-1} \left(\lambda_u \mathbf{W}_u \mathbf{a}_n + \lambda \sum_{m\in\Omega_{r_n}} \left(X_{nm} - \theta_n - \phi_m\right) \mathbf{v}_m\right)$$

2. Estimating θ_n keeping $\mathbf{u}_n, \mathbf{W}_u$ fixed: Now, we can simply set the derivative to 0 and obtain θ_n as follows:

$$\theta_n = \frac{\sum_{m\in\Omega_{r_n}} \left(X_{nm} - \phi_m - \mathbf{u}_n^T \mathbf{v}_m\right)}{\sum_{m\in\Omega_{r_n}} 1}$$

3. Estimating $\mathbf{W}_u$ keeping $\theta_n, \mathbf{u}_n, \forall n$ fixed: This time we get the loss function which resembles the loss of a multi-output linear regression problem. Thus, we can write the solution as follows -

$$\mathbf{W}_u = \left(\sum_{n=1}^{N} \mathbf{u}_n \mathbf{a}_n^T\right) \left(\sum_{n=1}^{N} \mathbf{a}_n \mathbf{a}_n^T\right)^{-1}$$

- **Estimating latent variables and parameters for items:** Similar procedure (as for estimating user variables and parameters) can be followed for items.

1. Estimating $\mathbf{v}_m$ keeping $\phi_m, \mathbf{W}_v$ fixed:

$$\mathbf{v}_m = \left(\sum_{n\in\Omega_{c_m}} \mathbf{u}_n^T \mathbf{u}_n + \lambda_v \mathbf{I}_K\right)^{-1} \left(\lambda_v \mathbf{W}_v \mathbf{b}_m + \lambda \sum_{n\in\Omega_{c_m}} \left(X_{nm} - \theta_n - \phi_m\right) \mathbf{u}_n\right)$$

2. Estimating ϕ_m keeping $\mathbf{v}_m, \mathbf{W}_v$ fixed:

$$\phi_m = \frac{\sum_{n\in\Omega_{c_m}} \left(X_{nm} - \theta_n - \mathbf{u}_n^T \mathbf{v}_m\right)}{\sum_{n\in\Omega_{c_m}} 1}$$

3. Estimating $\mathbf{W}_v$ keeping $\phi_m, \mathbf{v}_m, \forall m$ fixed:

$$\mathbf{W}_v = \left(\sum_{m=1}^{M} \mathbf{v}_m \mathbf{b}_m^T\right)\left(\sum_{m=1}^{M} \mathbf{b}_m \mathbf{b}_m^T\right)^{-1}$$

Problem 35. (EM for Naïve Bayes)

Assume that you want to train a naïve Bayes model on data with missing class labels. Specifically, there are k binary variables $X_1, \ldots X_k$ corresponding to the features, and a variable Y taking on values in $\{1, 2, \ldots, m\}$ denoting the class. Let us denote the set of model parameters as $P\left(X_i = 1 \mid Y = y\right) = \theta_{i|y}$ and $P(Y = y) = \theta_y$. You are given n data points $\mathcal{D} = \{\left(\mathbf{x}_1, y_1\right), \ldots, \left(\mathbf{x}_n, y_n\right)\}$ where $\mathbf{x}_i \in \{0,1\}^k$ and $y_i \in \{1, 2, \ldots, m, \times\}$. The value x means that the label of the data point is missing.

- Write down the log-likelihood $\ell(\theta)$ of the data as a function of the parameters θ.
- Recall that the E-step of the EM algorithm computes the posterior over the unknown variables when we fix the parameters θ. Compute these probabilities $\gamma_j\left(\mathbf{x}_i\right) = P\left(Y = j \mid \mathbf{x}_i; \theta\right)$ for j s.t. $y_i = x$.
- Once we have the quantities $\gamma_j(\cdot)$, we can compute the M-step update, which is computed as the maximizer θ^* of $\sum_{i=1}^{n} \sum_{j=1}^{m} \gamma_j\left(\mathbf{x}_i\right) \log P\left(\mathbf{x}_i, y_i = j; \theta\right)$. Show how to compute θ^*. Note that there are constraints on θ^* to make sure that the distributions are valid (non-negative and sum up to 1).

Solution:

The log-likelihood is equal to

$$\begin{aligned}
\ell(\theta) &= \log P(\mathcal{D}) \\
&= \sum_{\substack{i=1 \\ y_i = x}}^{n} \log P\left(\mathbf{x}_i; \theta\right) + \sum_{\substack{i=1 \\ y_i \neq x}}^{n} \log P\left(\mathbf{x}_i, y_i; \theta\right) \\
&= \sum_{\substack{i=1 \\ y_i = x}}^{n} \log \sum_{j=1}^{m} P\left(\mathbf{x}_i, Y = j; \theta\right) + \sum_{\substack{i=1 \\ y_i \neq x}}^{n} \log P\left(\mathbf{x}_i, y_i; \theta\right) \\
&= \sum_{\substack{i=1 \\ y_i = x}}^{n} \log \sum_{j=1}^{m} P\left(\mathbf{x}_i \mid Y = j; \theta\right) P(Y = j; \theta) + \sum_{\substack{i=1 \\ y_i \neq x}}^{n} \log P\left(\mathbf{x}_i, y_i; \theta\right) \\
&= \sum_{\substack{i=1 \\ y_i = x}}^{n} \log \sum_{j=1}^{m} \theta_j \prod_{l=1}^{k} \theta_{l|j}^{x_{i,l}} \left(1 - \theta_{l|j}\right)^{1 - x_{i,l}} + \sum_{\substack{i=1 \\ y_i \neq x}}^{n} \log \theta_{y_i} \prod_{l=1}^{k} \theta_{l|y_i}^{x_{i,l}} \left(1 - \theta_{l|y_i}\right)^{1 - x_{i,l}}.
\end{aligned}$$

To compute the requested posterior probabilities, note that by Bayes' rule

$$\begin{aligned}
\gamma_j\left(\mathbf{x}_i\right) &= P\left(y_i = j \mid \mathbf{x}_i; \theta\right) \\
&= \frac{P\left(\mathbf{x}_i \mid y_i = j; \theta\right) P\left(y_i = j; \theta\right)}{P\left(\mathbf{x}_i; \theta\right)} \\
&= \frac{1}{Z} P\left(\mathbf{x}_i \mid y_i = j; \theta\right) P\left(y_i = j; \theta\right) \\
&= \frac{1}{Z} \theta_j \prod_{l=1}^{k} \theta_{l|j}^{x_{i,l}} \left(1 - \theta_{l|j}\right)^{1 - x_{i,l}}.
\end{aligned}$$

We then have to compute the normalizer Z so that $\sum_{j=1}^{m} \gamma_j\left(\mathbf{x}_i\right) = 1$. Note that for those data points $\mathbf{x}_i$ for which we are given the labels y_i we set the $\gamma_j\left(\mathbf{x}_i\right)$ to be a deterministic distribution, i.e. $\gamma_j\left(x_i\right) = [j = y_i]$.

To compute the M-step update we have to optimize the following quantity

$$\sum_{i=1}^{n}\sum_{j=1}^{m}\gamma_j\left(\mathbf{x}_i\right)\log P\left(\mathbf{x}_i, y_i=j;\theta\right)=\sum_{i=1}^{n}\sum_{j=1}^{m}\gamma_j\left(\mathbf{x}_i\right)\left[\log\theta_j+\sum_{l=1}^{k}\log\theta_{l|j}^{x_{i,l}}\left(1-\theta_{l|j}\right)^{1-x_{i,l}}\right]$$

with respect to the parameters θ. We form the Lagrangian by adding a multiplier λ to make sure that $\sum_{j=1}^{m}\theta_j=1$:

$$\mathcal{L}(\theta,\lambda)=\sum_{i=1}^{n}\sum_{j=1}^{m}\gamma_j\left(\mathbf{x}_i\right)\left[\log\theta_j+\sum_{l=1}^{k}\log\theta_{l|j}^{x_{i,l}}\left(1-\theta_{l|j}\right)^{1-x_{i,l}}\right]+\lambda\left(\sum_{j=1}^{m}\theta_j-1\right).$$

By setting the derivatives to zero we obtain:

$$\frac{\partial}{\partial_{\theta_{l|j}}}\mathcal{L}(\theta,\lambda)=\sum_{\substack{i=1\\x_{i,l}=1}}^{n}\gamma_j\left(\mathbf{x}_i\right)/\theta_{l|j}+\sum_{\substack{i=1\\x_{i,l}=0}}^{n}\gamma_j\left(\mathbf{x}_i\right)/\left(\theta_{l|j}-1\right)=0\Longrightarrow\theta_{l|j}=\frac{\sum_{i=1}^{n}\left[x_{i,l}=1\right]\gamma_j\left(\mathbf{x}_i\right)}{\sum_{i=1}^{n}\gamma_j\left(\mathbf{x}_i\right)}$$

$$\frac{\partial}{\partial_{\theta_j}}\mathcal{L}(\theta,\lambda)=\sum_{i=1}^{n}\gamma_j\left(\mathbf{x}_i\right)/\theta_j+\lambda=0\Longrightarrow\theta_j=-\frac{\sum_{i=1}^{n}\gamma_j\left(\mathbf{x}_i\right)}{\lambda}$$

From the constraint $\sum_{j=1}^{m}\theta_j=1$ we find the correct multiplier to be $\lambda=-\sum_{i=1}^{n}\sum_{j=1}^{m}\gamma_j\left(\mathbf{x}_i\right)=-n$.

Problem 36. (EM for a 1D Laplacian Mixture Model)

In this problem, you will derive the EM algorithm for a one-dimensional Laplacian mixture model. You are given n observations $x_1,\ldots,x_n\in\mathbb{R}$ and we want to fit a mixture of m Laplacians, which has the following density

$$f(x)=\sum_{j=1}^{m}\pi_j f_L\left(x;\mu_j,\beta_j\right),$$

where $f_L\left(x;\mu_j,\beta_j\right)=\frac{1}{2\beta_j}e^{-\frac{1}{\beta_j}\left|x-\mu_j\right|}$, and the mixture weights π_j are a convex combination, i.e. $\pi_j\geq 0$ and $\sum_{j=1}^{m}\pi_j=1$. For simplicity, assume that the scale parameters $\beta_j>0$ are known beforehand and thus fixed.

- Introduce latent variables so that we can apply the EM procedure.
- Analogously to the previous question, write down the steps of the EM procedure for this model. If some updates cannot be written analytically, give an approach on how to compute them.

  (Hint: Recall a property of functions that makes them easy to optimize.)

Solution:

For each data point x_i, we introduce a latent variable $Y_i\in\{1,2,\ldots,m\}$ denoting the component that point belongs to. For the E-step, we compute the posterior over the classes similarly to the previous problem, i.e.

$$\gamma_j\left(x_i\right)=P\left(y_i=j\mid x_i\right)\propto P\left(x_i\mid y_i=j\right)P\left(y_i=j\right)=\pi_j f_L\left(x_i;\mu_j,\beta_j\right).$$

Again, we have to normalize so that the final posterior is equal to

$$\gamma_j\left(x_i\right)=\frac{\pi_j f_L\left(x_i;\mu_j,\beta_j\right)}{\sum_{l=1}^{m}\pi_l f_L\left(x_i;\mu_l,\beta_l\right)}$$

In the M-step, we optimize

$$\begin{aligned}\sum_{i=1}^{n}\sum_{j=1}^{m}\gamma_j\left(x_i\right)\log P\left(x_i,y_i=j\right)&=\sum_{i=1}^{n}\sum_{j=1}^{m}\gamma_j\left(x_i\right)\log\pi_j f_L\left(x_i;\mu_j,\beta_j\right)\\&=\sum_{i=1}^{n}\sum_{j=1}^{m}\gamma_j\left(x_i\right)\left(\log\pi_j-\frac{1}{\beta_j}\left|x_i-\mu_j\right|\right)+\text{const.}\end{aligned}\tag{20}$$

We add a Lagrange multiplier λ to make sure that $\sum_{j=1}^{m} \pi_j = 1$ and obtain the Lagrangian

$$\mathcal{L}(\pi, \mu, \lambda) = \sum_{i=1}^{n} \sum_{j=1}^{m} \gamma_j (x_i) \left(\log \pi_j - \frac{1}{\beta_j} |x_i - \mu_j| \right) + \lambda \left(\sum_{j=1}^{m} \pi_j - 1 \right)$$

Exactly as in the previous problem, by setting the gradient with respect to π_j to zero, we obtain

$$\frac{\partial}{\partial_{\pi_j}} \mathcal{L}(\pi, \mu, \lambda) = \sum_{i=1}^{n} \gamma_j (x_i) / \pi_j + \lambda = 0 \Longrightarrow \pi_j = \frac{\sum_{i=1}^{n} \gamma_j (x_i)}{-\lambda}.$$

The multiplier is again equal to $\lambda = -n$, and we arrive at the same equation as in the last example. If we want to maximize Eqn. (20) with respect to the variables μ_j, we have to solve m separate optimization problems, one for each μ_j. These m problems have the following form

$$\underset{\mu_j}{\text{maximize}} - \sum_{i=1}^{n} \frac{\gamma_j (x_i)}{\beta_j} |x_i - \mu_j|$$

These are one-dimensional convex optimization problems (the negative of the objective is easily seen to be convex). While one can try solving this via an iterative process like subgradient descent, a direct approach is also possible if we observe that the function is piecewise linear. The breakpoints are $x_1, x_2, \ldots, x_n$. Hence, the optimum must be attained at one of these n points, and we can simply set μ_j to the point x_i with the largest objective value.

Problem 37. (A different perspective on EM)

In this question, you will show that EM can be seen as iteratively maximizing a lower bound on the log-likelihood. We will treat any general model $P(X, Z)$ with observed variables X and latent variables Z. For the sake of simplicity, we will assume that Z is discrete and takes on values in $\{1, 2, \ldots, m\}$. If we observe $X = \mathbf{x}$, the goal is to maximize the log-likelihood

$$\ell(\theta) = \log P(\mathbf{x}; \theta) = \log \sum_{z=1}^{m} P(\mathbf{x}, z; \theta)$$

with respect to the parameter vector θ. In what follows, we will denote any distribution over the latent variables by $Q(Z)$.

- Show that if $Q(z) > 0$ when $P(\mathbf{x}, z) > 0$, then it holds that (Hint: Consider using Jensen's inequality)

  $$\ell(\theta) \geq \mathbb{E}_Q[\log P(X, Z)] - \sum_{z=1}^{m} Q(z) \log Q(z).$$

  Hence, we have a bound on the log-likelihood parametrized by a distribution $Q(Z)$ over the latent variables.

- Show that for a fixed θ, the lower bound is maximized for $Q^*(Z) = P(Z \mid X; \theta)$. Moreover, it shows that the bound is exact (holds with equality) for this specific distribution $Q^*(Z)$.

  (Hint: Do not forget to add Lagrange multipliers to make sure that Q^* is a valid distribution.)

- Show that if we optimize with respect to Q and θ in an alternating manner, this corresponds to the EM procedure. Discuss what this implies for the convergence properties of EM.

Solution:

For the first part, note that

$$\begin{aligned} \ell(\theta) = \log P(\mathbf{x}; \theta) = \log \sum_{z=1}^{m} P(\mathbf{x}, z; \theta) = \log \sum_{z=1}^{m} \frac{P(\mathbf{x}, z; \theta)}{Q(z)} Q(z) = \log \mathbb{E}_{Z \sim Q} \left[\frac{P(\mathbf{x}, z; \theta)}{Q(z)} \right] \\ \geq \mathbb{E}_{Z \sim Q} \left[\log \frac{P(\mathbf{x}, z; \theta)}{Q(z)} \right] = \mathbb{E}_{Z \sim Q}[\log P(\mathbf{x}, z; \theta)] - \sum_{z=1}^{m} Q(z) \log Q(z), \end{aligned}$$

where for the inequality, we have used Jensen's inequality. Now, assume that we want to maximize the above with respect to Q, and let us add a multiplier λ to make sure that Q sums up to 1. Then, we have the following Lagrangian

$$\mathcal{L}(Q,\lambda) = \sum_{z=1}^{m} Q(z)\log P(\mathbf{x},z;\theta) - \sum_{z=1}^{m} Q(z)\log Q(z) + \lambda\left(\sum_{z=1}^{m} Q(z) - 1\right).$$

By setting the derivative of the Lagrangian with respect to $Q(z)$ to zero, we have

$$\frac{\partial}{\partial_{Q(z)}}\mathcal{L}(Q,\lambda) = \log P(\mathbf{x},z;\theta) - 1 - \log Q(z) + \lambda = 0 \Longrightarrow Q(z) = e^{\lambda-1}P(\mathbf{x},z;\theta)$$

Hence, we have that $Q(z) \propto P(\mathbf{x},z;\theta)$ and this is exactly the posterior $P(Z \mid \mathbf{x};\theta)$, which we had to show. It is also easy to see that the bound is tight, as

$$\mathbb{E}_{Z\sim Q}\left[\log\frac{P(\mathbf{x},z;\theta)}{Q(z)}\right] = \sum_{z=1}^{m} Q(z)\log\frac{P(\mathbf{x},z;\theta)}{Q(z)} = \sum_{z=1}^{m} P(z \mid \mathbf{x};\theta)\log\frac{P(z \mid \mathbf{x};\theta)P(\mathbf{x};\theta)}{P(z \mid \mathbf{x};\theta)} = \log P(\mathbf{x};\theta)$$

Then we can easily see the EM algorithm as optimizing the lower bound with respect to $Q(\cdot)$ and θ in an alternating manner. Specifically, if we optimize with respect to Q, we have shown that the optimal Q is posterior, and this is exactly the E-step. Optimizing with respect to θ for fixed Q is clearly equivalent to the M-step. The EM algorithm has to converge as the lower bound is monotonically increased at every step.

Problem 38. (EM for Censored Linear Regression)

Suppose you are trying to learn a model that can predict how long a program will take to run for different settings. In some situations, when the program is taking too long, you abort the program and note down the time at which you abort it. These values are lower bounds for the actual running time of the program. We call this type of data **right-censored**. Concretely, all you know is that the running time $y_i \geq c_i$, where c_i is the censoring time. Written in another way, one can say $y_i = \min\{z_i, c_i\}$ where z_i is the true running time. Our goal is to derive an EM algorithm for fitting a linear regression model to right-censored data.

(a) Let $z_i = \mu_i + \sigma\varepsilon_i$, where $\varepsilon_i \sim \mathcal{N}(0,1)$. Suppose that we do not observe z_i, but we observe the fact that it is higher than some threshold. Namely, we observe the event $E = \mathbb{I}(z_i \geq c_i)$. Show that

$$\mathbb{E}[z_i \mid z_i \geq c_i] = \mu_i + \sigma R\left(\frac{c_i - \mu_i}{\sigma}\right)$$

and

$$\mathbb{E}\left[z_i^2 \mid z_i \geq c_i\right] = \mu_i^2 + \sigma^2 + \sigma(c_i + \mu_i)R\left(\frac{c_i - \mu_i}{\sigma}\right),$$

where we have defined

$$R(x) := \frac{\phi(x)}{1 - \Phi(x)}.$$

Here, $\phi(x)$ is the pdf of the standard Gaussian, and $\Phi(x)$ is its cdf.

(b) Derive the EM algorithm for fitting a linear regression model to right-censored data. Describe completely the E-step and M-step.

Solution:

(a) First note that $p(\varepsilon_i \mid E) = \frac{p(\varepsilon_i, E)}{p(E)}$. Also for brevity, define $a_i := \frac{c_i - \mu_i}{\sigma}$. Then we have $E = \mathbb{I}(z_i \geq c_i) = \mathbb{I}(\varepsilon_i \geq a_i)$. So we can write

$$\begin{aligned}\mathbb{E}[z_i \mid z_i \geq c_i] &= \int_{\mathbb{R}} z_i p(\varepsilon_i \mid E)\,d\varepsilon_i = \int_{\mathbb{R}} z_i \frac{p(\varepsilon_i, E)}{p(E)}d\varepsilon_i \\ &= \frac{1}{p(E)}\int_{a_i}^{\infty}(\mu_i + \sigma\varepsilon_i)\,p(\varepsilon_i)\,d\varepsilon_i = \mu_i + \frac{\sigma}{p(E)}\int_{a_i}^{\infty}\varepsilon_i p(\varepsilon_i)\,d\varepsilon_i\end{aligned}$$

The equality follows from the fact that $p(E) = \int_{a_i}^{\infty} p(\varepsilon_i)\, d\varepsilon_i = 1 - \Phi(a_i)$. Now observe that for the Standard Normal distribution density $\phi(x)$, we have

$$\frac{d}{dx}\phi(x) = -x\phi(x),$$

implying that

$$\int_{a_i}^{\infty} \varepsilon_i p(\varepsilon_i)\, d\varepsilon_i = \phi(a_i) - \phi(+\infty) = \phi(a_i).$$

Putting it all together we get

$$\mathbb{E}[z_i \mid z_i \geq c_i] = \mu_i + \sigma \frac{\phi(a_i)}{1 - \Phi(a_i)} = \mu_i + \sigma R\left(\frac{c_i - \mu_i}{\sigma}\right).$$

To compute $\mathbb{E}\left[z_i^2 \mid z_i \geq c_i\right]$, we first note that

$$\frac{d^2}{dx^2}\phi(x) = -\phi(x) + x^2\phi(x),$$

implying that

$$\int_a^b x^2\phi(x)dx = \Phi(b) - \Phi(a) + a\phi(a) - b\phi(b). \tag{21}$$

Now we have

$$\begin{aligned}
\mathbb{E}\left[z_i^2 \mid z_i \geq c_i\right] &= \frac{1}{p(E)} \int_{a_i}^{\infty} \left(\mu_i^2 + 2\mu_i\sigma\varepsilon_i + \sigma^2\varepsilon_i^2\right) p(\varepsilon_i)\, d\varepsilon_i \\
&= \mu_i^2 + \frac{2\mu_i\sigma}{p(E)} \int_{a_i}^{\infty} \varepsilon_i p(\varepsilon_i)\, d\varepsilon_i + \frac{\sigma^2}{p(E)} \int_{a_i}^{\infty} \varepsilon_i^2 p(\varepsilon_i)\, d\varepsilon_i \\
&= \mu_i^2 + \frac{2\mu_i\sigma}{p(E)}\phi(a_i) + \frac{\sigma^2}{p(E)}\left(1 - \Phi(a_i) + a_i\phi(a_i)\right) \\
&= \mu_i^2 + \frac{2\mu_i\sigma}{1 - \Phi(a_i)}\phi(a_i) + \frac{\sigma^2}{1 - \Phi(a_i)}\left(1 - \Phi(a_i) + a_i\phi(a_i)\right) \\
&= \mu_i^2 + \sigma^2 + \left(2\mu_i\sigma + a_i\sigma^2\right) R(a_i) \\
&= \mu_i^2 + \sigma^2 + \sigma(\mu_i + c_i) R(a_i)
\end{aligned}$$

(b) The model we have for linear regression is $z_i \sim \mathcal{N}\left(\mathbf{w}^\top \mathbf{x}_i, \sigma\right)$ where z_i is missing. Our observed variable is $y_i = \min\{z_i, c_i\}$. For ease of notation, let

$$d_i = \begin{cases} 1 & \text{if } z_i \leq c_i \\ 0 & \text{if } z_i > c_i \end{cases}$$

to be the censoring indicator, i.e., it is 1 if the observation is not censored, and is 0 otherwise. We denote by z the set of all z_i's, by $\mathbf{X}$ the set of all $\mathbf{x}_i$'s, by $\mathbf{y}$ the set of all y_i's, by $\mathbf{c}$ the set of all c_i's, and by $\mathbf{d}$ the set of all d_i's.

The complete-data log-likelihood would be

$$\log p(z_i \mid \mathbf{w}) = -\frac{1}{2\sigma^2}\left(z_i - \mathbf{w}^\top \mathbf{x}_i\right)^2 + \text{ const.}$$

For the first step, we need to find the posterior of the missing data given the observed data and parameters. We have

$$p(z_i \mid \underbrace{\mathbf{x}_i, y_i, c_i, d_i}_{\text{observed}}, \mathbf{w}) = \begin{cases} \delta(z_i - y_i) & \text{if } d_i = 1 \\ \frac{\mathcal{N}\left(z_i \mid \mathbf{w}^\top \mathbf{x}_i, \sigma\right)}{1 - \Phi\left(\frac{c_i - \mathbf{w}^\top \mathbf{x}_i}{\sigma}\right)} & \text{if } d_i = 0 \end{cases},$$

in which $\delta(\cdot)$ is the dirac delta function, and $1-\Phi\left(\frac{c_i-\mu_i}{\sigma}\right)$ is the probability that $z_i > c_i$.
Now we should compute the expected value of the complete-data log-likelihood w.r.t the posterior $p\left(z_i \mid \mathbf{x}_i, y_i, c_i, d_i, \mathbf{w}'\right)$. This can be computed as

$$\int_{\mathbb{R}} \log p\left(z_i \mid \mathbf{w}\right) \cdot p\left(z_i \mid \mathbf{x}_i, y_i, c_i, d_i, \mathbf{w}'\right) d z_i .$$

Note that if $d_i=1$, the integral is evaluated as $-\frac{1}{2\sigma^2}\left(y_i-\mathbf{w}^{\top}\mathbf{x}_i\right)^2$, and if $d_i=0$, we can use part (a) to compute the expectation. For ease of notation, we call $\mu_i := \mathbf{w}^{\top}\mathbf{x}_i$ and $\mu_i' := \mathbf{w}'^{\top}\mathbf{x}_i$ and $a_i = \frac{c_i-\mu_i'}{\sigma}$. We then have

$$\begin{aligned}
\mathbb{E}\left[\log p\left(z_i \mid \mathbf{w}\right) \mid z_i > c_i\right] &= -\frac{1}{2\sigma^2}\left\{\mu_i^2+\mathbb{E}\left[z_i^2 \mid z_i>c_i\right]-2\mu_i \mathbb{E}\left[z_i \mid z_i>c_i\right]\right\} \\
&= -\frac{1}{2\sigma^2}\left\{\mu_i^2+\mu_i'+R\left(a_i\right)-2\mu_i \underbrace{\left(\left(\mu_i'\right)^2+\sigma^2+\sigma\left(\mu_i'+c_i\right)R\left(a_i\right)\right)}_{:=b_i}\right\} .
\end{aligned}$$

Adding the evaluated expectation for all data and removing the terms that are not dependent on w, we get

$$\begin{aligned}
Q\left(\mathbf{w}, \mathbf{w}'\right) &= -\frac{1}{2\sigma^2} \sum_{i=1}^{n}\left(y_i-\mu_i\right)^2 \cdot d_i+\left(\mu_i^2-2 b_i \mu_i\right) \cdot\left(1-d_i\right) \\
&\doteq -\frac{1}{2\sigma^2} \sum_{i=1}^{n}\left(\mu_i^2-2 y_i \mu_i\right) \cdot d_i+\left(\mu_i^2-2 b_i \mu_i\right) \cdot\left(1-d_i\right) \\
&= -\frac{1}{2\sigma^2} \sum_{i=1}^{n} \mu_i^2-2\mu_i \underbrace{\left(y_i d_i+b_i\left(1-d_i\right)\right)}_{:=e_i} \\
&= -\frac{1}{2\sigma^2}\left(\mathbf{w}^{\top}\mathbf{X}^{\top}\mathbf{X}\mathbf{w}-2\mathbf{w}^{\top}\mathbf{X}^{\top}\mathbf{e}\right)
\end{aligned}$$

The maximizer for $Q\left(\mathbf{w}, \mathbf{w}'\right)$ would be

$$\mathbf{w}^{\star} = -\frac{1}{2}\left(\mathbf{X}^{\top}\mathbf{e}\right)\left(\mathbf{X}^{\top}\mathbf{X}\right)^{-1},$$

which sums up the M-step.

Problem 39. (Yet another perspective on EM)

The EM algorithm is a general technique for finding maximum likelihood solutions for probabilistic models having latent variables. Take a probabilistic model in which we denote all of the observed variables as $\mathbf{X}$ and all of the hidden variables as $\mathbf{Z}$ (here we assume $\mathbf{Z}$ is discrete, for the sake of simplicity). Let us assume that the joint distribution is $p(\mathbf{X}, \mathbf{Z} \mid \boldsymbol{\theta})$, where $\boldsymbol{\theta}$ is the set of all parameters describing this distribution (e.g. for a Gaussian distribution, $\boldsymbol{\theta} = (\mu, \Sigma)$). The goal is to maximize the likelihood function

$$p(\mathbf{X} \mid \boldsymbol{\theta}) = \sum_{\mathbf{Z}} p(\mathbf{X}, \mathbf{Z} \mid \boldsymbol{\theta}).$$

(a) For an arbitrary distribution $q(\mathbf{Z})$ over the latent variables, show that the following decomposition holds:

$$\ln p(\mathbf{X} \mid \boldsymbol{\theta}) = \mathcal{L}(q, \boldsymbol{\theta}) + \mathrm{D}_{\mathrm{KL}}\left(q \| p_{\text{post}}\right) \tag{22}$$

where $p_{\text{post}} = p(\mathbf{Z} \mid \mathbf{X}, \boldsymbol{\theta})$ is the posterior distribution. Also find the formulation of $\mathcal{L}(q, \boldsymbol{\theta})$.

(b) Verify that $\mathcal{L}(q, \boldsymbol{\theta}) \leq \ln p(\mathbf{X} \mid \boldsymbol{\theta})$, and that equality holds if and only if $q(\mathbf{Z}) = p(\mathbf{Z} \mid \mathbf{X}, \boldsymbol{\theta})$.

(c) Suppose that the current value of the parameters is θ_{curr}. Verify that in the E-step, the lower bound $\mathcal{L}\left(q, \boldsymbol{\theta}_{\text{curr}}\right)$ is maximized with respect to the distribution $q(\mathbf{Z})$, while keeping $\boldsymbol{\theta}_{\text{curr}}$ fixed. Since the lefthand-side of Eqn. (22) does not depend on $q(\mathbf{Z})$, maximizing $\mathcal{L}\left(q, \boldsymbol{\theta}_{\text{curr}}\right)$ will result in minimizing the KL divergence between q and p_{post}, which happens at $q^{\star} = p_{\text{post}}$.

(d) Verify that in the M-step, the lower bound $\mathcal{L}(q, \boldsymbol{\theta})$ is maximized with respect to $\boldsymbol{\theta}$ while keeping $q(\mathbf{Z})$ fixed, resulting in a new value of parameters θ_{new}. This step will increase the left-hand side of Eqn. (22) (if it is not already in a local maximum).

(e) Substitute $q(\mathbf{Z}) = p\left(\mathbf{Z} \mid \mathbf{X}, \boldsymbol{\theta}_{\text{curr}}\right)$ in Eqn. (22), and observe that

$$\mathcal{L}(q, \boldsymbol{\theta}) = \mathbb{E}_q \left[\text{complete-data log likelihood }\right] - H(q).$$

In other words, in the M-step, we are maximizing the expectation of the complete-data log-likelihood $(p(X, Z|\theta))$ since the entropy term is independent of $\boldsymbol{\theta}$. Compare this result with the EM for Gaussian mixture models.

(f) Show that the lower bound $\mathcal{L}(q, \boldsymbol{\theta})$, where $q(\mathbf{Z}) = q^{\star}(\mathbf{Z}) = p\left(\mathbf{Z} \mid \mathbf{X}, \boldsymbol{\theta}_{\text{curr}}\right)$, has the same gradient w.r.t. $\boldsymbol{\theta}$ as the log likelihood function $p(\mathbf{X} \mid \boldsymbol{\theta})$ at the point $\boldsymbol{\theta} = \boldsymbol{\theta}_{\text{curr}}$. This shows that the lower bound becomes tangent to the log-likelihood function at the end of the E-step.

Solution:

(a) By computing the KL Divergence of q to p_{post} we get

$$\mathrm{D}_{\mathrm{KL}}\left(q \| p_{\text{post}}\right) = \sum_{\mathbf{Z}} q(\mathbf{Z}) \log \frac{q(\mathbf{Z})}{p(\mathbf{Z} \mid \mathbf{X}, \boldsymbol{\theta})}.$$

Knowing that $p(\mathbf{Z} \mid \mathbf{X}, \boldsymbol{\theta}) = \frac{p(\mathbf{X}, \mathbf{Z} | \boldsymbol{\theta})}{p(\mathbf{X} | \boldsymbol{\theta})}$, we get

$$\mathrm{D}_{\mathrm{KL}}\left(q \| p_{\text{post}}\right) = \sum_{\mathbf{Z}} q(\mathbf{Z}) \log \frac{q(\mathbf{Z}) p(\mathbf{X} \mid \boldsymbol{\theta})}{p(\mathbf{Z}, \mathbf{X} \mid \boldsymbol{\theta})} = \sum_{\mathbf{Z}} q(\mathbf{Z}) \log \frac{q(\mathbf{Z})}{p(\mathbf{Z}, \mathbf{X} \mid \boldsymbol{\theta})} + \log p(\mathbf{X} \mid \boldsymbol{\theta}).$$

This implies that

$$\log p(\mathbf{X} \mid \boldsymbol{\theta}) = \mathrm{D}_{\mathrm{KL}}\left(q \| p_{\text{post}}\right) + \mathcal{L}(q, \boldsymbol{\theta}),$$

where

$$\mathcal{L}(q, \boldsymbol{\theta}) = \sum_{\mathbf{Z}} q(\mathbf{Z}) \log \frac{p(\mathbf{Z}, \mathbf{X} \mid \boldsymbol{\theta})}{q(\mathbf{Z})}. \tag{23}$$

(b) Since KL divergence is always nonnegative and is zero only if the distributions are the same, we have

$$\mathcal{L}(q, \boldsymbol{\theta}) \leq \log p(\mathbf{X} \mid \boldsymbol{\theta}).$$

with equality iff $q = p_{\text{post}}$.

(c) This is for you to verify. As seen in the examples in the slides of the course, in some situations, the E-step is just computing the posterior, which is equivalent to minimizing the KL divergence of q to the posterior.

(d) Another thing to verify by yourself. Look at GMMs as an example and try to relate the variables defined in here and the parameters and variables there.

(e) Putting $q = p_{\text{post}}$ in Eqn. (23) we get

$$\begin{aligned}
\mathcal{L}\left(p_{\text{post}}, \boldsymbol{\theta}\right) &= \sum_{\mathbf{Z}} p\left(\mathbf{Z} \mid \mathbf{X}, \boldsymbol{\theta}_{\text{curr}}\right) \log \frac{p(\mathbf{Z}, \mathbf{X} \mid \boldsymbol{\theta})}{p(\mathbf{Z} \mid \mathbf{X}, \boldsymbol{\theta})} \\
&= \sum_{\mathbf{Z}} p(\mathbf{Z} \mid \mathbf{X}, \boldsymbol{\theta}) \log p(\mathbf{Z}, \mathbf{X} \mid \boldsymbol{\theta}) - \sum_{\mathbf{Z}} p(\mathbf{Z} \mid \mathbf{X}, \boldsymbol{\theta}) \log p(\mathbf{Z} \mid \mathbf{X}, \boldsymbol{\theta}) \\
&= \mathbb{E}_q[\text{complete-data log likelihood}] + H(q)
\end{aligned}$$

(f) We have

$$\begin{aligned}
\nabla_{\boldsymbol{\theta}} \mathcal{L}(q, \boldsymbol{\theta})|_{\boldsymbol{\theta}=\boldsymbol{\theta}_{\text{curr}}} &= \nabla_{\boldsymbol{\theta}} \sum_{\mathbf{Z}} q(\mathbf{Z}) \log \frac{p(\mathbf{Z}, \mathbf{X} \mid \boldsymbol{\theta})}{q(\mathbf{Z})} \\
&= \sum_{\mathbf{Z}} q(\mathbf{Z}) \frac{\nabla_{\boldsymbol{\theta}} p(\mathbf{Z}, \mathbf{X} \mid \boldsymbol{\theta})|_{\boldsymbol{\theta}=\boldsymbol{\theta}_{\text{curr}}}}{p\left(\mathbf{Z}, \mathbf{X} \mid \boldsymbol{\theta}_{\text{curr}}\right)} \\
&= \sum_{\mathbf{Z}} \frac{\nabla_{\boldsymbol{\theta}} p(\mathbf{Z}, \mathbf{X} \mid \boldsymbol{\theta})|_{\boldsymbol{\theta}=\boldsymbol{\theta}_{\text{curr}}}}{p\left(\mathbf{X} \mid \boldsymbol{\theta}_{\text{curr}}\right)} \\
&= \frac{\nabla_{\boldsymbol{\theta}} p(\mathbf{X} \mid \boldsymbol{\theta})|_{\boldsymbol{\theta}=\boldsymbol{\theta}_{\text{curr}}}}{p\left(\mathbf{X} \mid \boldsymbol{\theta}_{\text{curr}}\right)} = \nabla_{\boldsymbol{\theta}} \log p(\mathbf{X} \mid \boldsymbol{\theta})|_{\boldsymbol{\theta}=\boldsymbol{\theta}_{\text{curr}}}
\end{aligned}$$

Problem 40. (Generative meets Discriminative)

Consider a generative classification model for binary classification. Assume the class-marginal distribution to be defined as $p(y=1)=\pi$ and assume each class-conditional distribution to be defined as a product of D Bernoulli distributions, i.e., $p(\boldsymbol{x} \mid y=1)=\prod_{d=1}^{D} p\left(x_d \mid y=1\right)$ where $p\left(x_d \mid y=1\right)=\operatorname{Bernoulli}\left(x_d \mid \mu_{d,1}\right)$, and $p(\boldsymbol{x} \mid y=0)=\prod_{d=1}^{D} p\left(x_d \mid y=0\right)$ where $p\left(x_d \mid y=0\right)=\operatorname{Bernoulli}\left(x_d \mid \mu_{d,0}\right)$. Note that this makes use of the naïve Bayes assumption.

Show that this model is equivalent (in its mathematical form) to a probabilistic discriminative classifier. In particular, derive the expression for $p(y=1 \mid \boldsymbol{x})$, and state what type of decision boundary this model will learn - linear, quadratic, or something else (looking at the expression of $p(y=1 \mid \boldsymbol{x})$ should reveal that)? Clearly write down the expressions for the parameters of the equivalent probabilistic discriminative model in terms of the generative model parameters $(\pi, \mu_{d,0}, \mu_{d,1})$. Note that you do not have to estimate the parameters $\pi, \mu_{d,0}, \mu_{d,1}$ (but you may try that for practice if you want).

Solution:

Consider a generative classification model for binary classification with the Naive Bayes assumption of feature independence. Let the class marginal and class conditional distributions be as follows:

$$p(y=1)=\pi, \quad p(\mathbf{x} \mid y=j)=\prod_{d=1}^{D} \operatorname{Bernoulli}\left(x_d \mid \mu_{d,j}\right)=\prod_{d=1}^{D} \mu_{d,j}^{x_d}\left(1-\mu_{d,j}\right)^{1-x_d}, \quad j=0,1$$

For convenience, let us consider simplification of only $p(y=1 \mid \mathbf{x})$.

$$\begin{aligned}
p(y=1 \mid \mathbf{x}) &= \frac{p(\mathbf{x} \mid y=1) p(y=1)}{p(\mathbf{x})}=\frac{p(\mathbf{x} \mid y=1) p(y=1)}{p(\mathbf{x} \mid y=1) p(y=1)+p(\mathbf{x} \mid y=0) p(y=0)} \\
\therefore p(y=1 \mid \mathbf{x}) &= \frac{\pi \prod_{d=1}^{D} B\left(x_d, \mu_{d,1}\right)}{\pi \prod_{d=1}^{D} B\left(x_d, \mu_{d,1}\right)+(1-\pi) \prod_{d=1}^{D} B\left(x_d, \mu_{d,0}\right)}; \quad \text{where } B \text{ denotes Bernoulli} \\
\therefore p(y=1 \mid \mathbf{x}) &= \frac{1}{1+\frac{1-\pi}{\pi} \frac{\prod_{d=1}^{D} B\left(x_d, \mu_{d,1}\right)}{\prod_{d=1}^{D} B\left(x_d, \mu_{d,0}\right)}}=\frac{1}{1+\frac{1-\pi}{\pi} \prod_{d=1}^{D}\left(\frac{\mu_{d0}}{\mu_{d1}}\right)^{x_d}\left(\frac{1-\mu_{d0}}{1-\mu_{d1}}\right)^{1-x_d}}
\end{aligned}$$

Now, let $s_d=\frac{\mu_{d0}}{\mu_{d1}}, r_d=\frac{1-\mu_{d0}}{1-\mu_{d1}}$ and $C=\frac{1-\pi}{\pi}$. On further simplification, we get

$$p(y=1 \mid \mathbf{x})=\frac{1}{1+C \prod_{d=1}^{D} s_d^{x_d} r_d^{1-x_d}}=\frac{1}{1+\exp \left(\log (C)+\sum_{d=1}^{D} x_d \log \left(s_d\right)+\left(1-x_d\right) \log \left(r_d\right)\right)}$$

$$\therefore p(y=1 \mid \mathbf{x})=\frac{1}{1+\exp \left(\mathbf{w}^T \mathbf{x}+b\right)},$$

where $\mathbf{w}=\left[\log \left(s_1\right)-\log \left(r_1\right), \ldots, \log \left(s_D\right)-\log \left(r_D\right)\right]^T$ and $b=\log \left(C\left(\prod_{d=1}^{D} r_d\right)\right)$.

Thus, this corresponds to the discriminative logistic regression model for binary classification with labels $\in\{0,1\}$. The decision boundary is linear.

Problem 41. (K-means convergence)

In the K-means clustering algorithm, you are given a set of n points $x_i \in \mathbb{R}^d$, $i \in \{1, \ldots, n\}$ and you want to find the centers of k clusters $\mu = (\mu_1, \ldots, \mu_k)$ by minimizing the average distance from the points to the closest cluster center. Formally, you want to minimize the following loss function

$$L(\mu) = \sum_{i=1}^{n} \min_{j \in \{1,\ldots,k\}} \|x_i - \mu_j\|_2^2 .$$

To approximate the solution, we introduce new assignment variables $z_i \in \arg\min_{j\in\{1,\ldots,k\}} \|x_i - \mu_j\|_2^2$ for each data point x_i. The K-means algorithm iterates between updating the variables z_i (assignment step) and updating the centers $\mu_j = \frac{1}{|\{i:z_i=j\}|} \sum_{i:z_i=j} x_i$ (refitting step). The algorithm stops when no change occurs during the assignment step. Show that K-means are guaranteed to converge (to a local optimum).

Hint: You need to prove that the loss function is guaranteed to decrease monotonically in each iteration until convergence. Prove this separately for the assignment step and the refitting step.

Solution:

To prove the convergence of the K-means algorithm, we show that the loss function is guaranteed to decrease monotonically in each iteration until convergence for the assignment step and for the refitting step. Since the loss function is non-negative, the algorithm will eventually converge when the loss function reaches its (local) minimum.
Let $z = (z_1, \ldots, z_n)$ denote the cluster assignments for the n points.

(i) **Assignment step**

We can write down the original loss function $L(\mu)$ as follows:

$$L(\mu, z) = \sum_{i=1}^{n} \|x_i - \mu_{z_i}\|_2^2$$

Let us consider a data point x_i, and let z_i be the assignment from the previous iteration and z_i^* be the new assignment obtained as:

$$z_i^* \in \arg \min_{j \in \{1,\ldots,k\}} \|x_i - \mu_j\|_2^2$$

Let z^* denote the new cluster assignments for all the n points. The change in loss function after this assignment step is then given by:

$$L\left(\mu, z^*\right) - L(\mu, z) = \sum_{i=1}^{n} \left(\left\|x_i - \mu_{z_i^*}\right\|_2^2 - \|x_i - \mu_{z_i}\|_2^2 \right) \leq 0$$

The inequality holds by the rule z_i^* is determined, i.e. to assign x_i to the nearest cluster.

(ii) **Refitting step**

We can write down the original loss function $L(\mu)$ as follows:

$$L(\mu, z) = \sum_{j=1}^{k} \left(\sum_{i:z_i=j} \|x_i - \mu_j\|_2^2 \right)$$

Let us consider the j^{th} cluster, and let μ_j be the cluster center from the previous iteration and μ_j^* be the new cluster center obtained as:

$$\mu_j^* = \frac{1}{|\{i : z_i = j\}|} \sum_{i:z_i=j} x_i$$

Let μ^* denote the new cluster centers for all the k clusters. The change in loss function after this refitting step is then given by:

$$L\left(\mu^*, z\right) - L(\mu, z) = \sum_{j=1}^{k} \left(\left(\sum_{i:z_i=j} \left\|x_i - \mu_j^*\right\|_2^2 \right) - \left(\sum_{i:z_i=j} \|x_i - \mu_j\|_2^2 \right) \right) \leq 0$$

The inequality holds because the update rule of μ_j^* essentially minimizes this quantity.

Problem 42. (K-medians clustering)

In this exercise, you are asked to derive a new clustering algorithm that would use a different loss function given by

$$L(\mu) = \sum_{i=1}^{n} \min_{j \in \{1,\dots,k\}} \|x_i - \mu_j\|_1 .$$

(i) Find the update steps for both z_i and for μ_j in this case.

(ii) What can you say about the convergence of your algorithm?

(iii) In which situation would you prefer to use K-medians clustering instead of K-means clustering?

Solution:

(i) As in the K-means algorithm, let's again introduce hidden variables $z_i = \arg\min_{j \in 1,\dots,k} \|x_i - \mu_j\|_1$ for each data point x_i. Then the initial problem

$$\mu = \arg\min_{\mu} \sum_{i=1}^{n} \min_{j \in 1,\dots,k} \|x_i - \mu_j\|_1$$

can be rewritten in a different form (because we know where exactly the minimum is achieved):

$$\mu = \arg\min_{\mu} \sum_{i=1}^{n} \|x_i - \mu_{z_i}\|_1$$

In order to find the solution with respect to μ_j with fixed z_i, let's leave only the data points that correspond to the j^{th} component:

$$\mu_j = \arg\min_{\mu_j} \sum_{i:z_i=j} \|x_i - \mu_j\|_1 \qquad \mu_j = \arg\min_{\mu_j} \sum_{i:z_i=j} \sum_{q=1}^{d} |x_{i,q} - \mu_{j,q}|$$

This can again be separated component-wise:

$$\mu_{j,q} = \arg\min_{\mu_{j,q}} \sum_{i:z_i=j} |x_{i,q} - \mu_{j,q}|$$

Again, as in the K-means algorithm, we find the derivative of the function and set it to zero. In order to get rid of the L_1 norm, we also separate the functional into the sum over those $x_{i,q}$ that are smaller than $\mu_{j,q}$ and those that are larger:

$\sum_{i:z_i=j, x_{i,q} \le \mu_{j,q}} |x_{i,q} - \mu_{j,q}| + \sum_{i:z_i=j, x_{i,q} > \mu_{j,q}} |x_{i,q} - \mu_{j,q}| = \sum_{i:z_i=j, x_{i,q} \le \mu_{j,q}} (\mu_{j,q} - x_{i,q}) + \sum_{i:z_i=j, x_{i,q} > \mu_{j,q}} (x_{i,q} - \mu_{j,q})$
The derivative of every bracket in the sum is either $+1$ or -1, and the number of $+1$'s is exactly $|\{i : z_i = j, x_{i,q} \le \mu_{j,q}\}|$. Therefore, we need to set

$$|\{i : z_i = j, x_{i,q} \le \mu_{j,q}\}| - |\{i : z_i = j, x_{i,q} > \mu_{j,q}\}| = 0$$

This means that $\mu_{j,q}$ is nothing but the median of all the numbers $x_{i,q}, i : z_i = j$.

The resulting algorithm then iterates between two steps:

- $z_i = \arg\min_{j \in 1,\dots,k} \|x_i - \mu_j\|_1$
- $\mu_{j,q} = \text{median}\left(x_{i,q}, i : z_i = j\right), \ \forall j = 1, \dots, k; \ \forall q = 1, \dots, d.$

(ii) You can prove the same convergence properties for K-medians as for K-means.

(iii) In comparison with K-means, K-medians clustering is particularly robust to outliers. Thus, if we expect our input data to have many outliers, it is preferable to use K-medians clustering.

Problem 43. (PCA)

Suppose we have a dataset with 4 points:

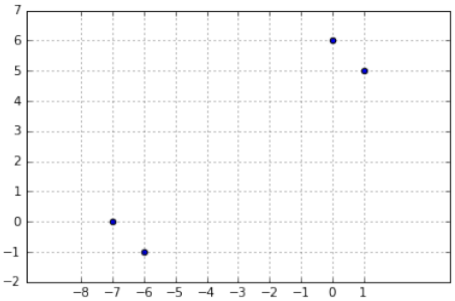
$$\mathcal{D} = \{(1,5), (0,6), (-7,0), (-6,-1)\}$$

(a) Plot the dataset and try to guess two principal components ($k = 2$).

(b) Compute the empirical covariance matrix, its eigenvalues and eigenvectors. Do the eigenvectors correspond to your guess of principal components? Please do not forget the assumptions of PCA. (The dataset should be centered and we want unit eigenvectors.)

Solution:

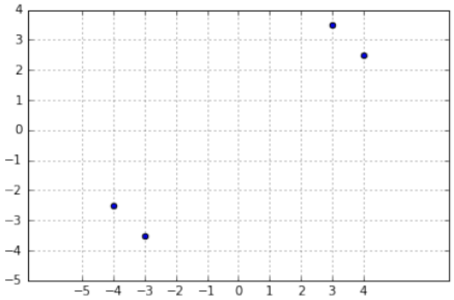
(a) Plot of the original dataset:



(b) We first need to center the data by subtracting from it its mean $(-3, 2.5)^T$, obtaining

$$\boldsymbol{x_1} = (4, 2.5)^T, \quad \boldsymbol{x_2} = (3, 3.5)^T, \quad \boldsymbol{x_3} = (-4, -2.5)^T, \quad \boldsymbol{x_4} = (-3, -3.5)^T.$$

The plot of the centered dataset:



For the empirical covariance matrix, we obtain

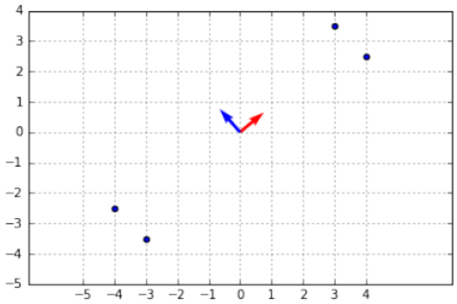
$$\Sigma = \frac{1}{n}\sum_{i=1}^{n} \boldsymbol{x_i}\boldsymbol{x_i}^T = \frac{1}{4} \cdot \begin{pmatrix} 50 & 41 \\ 41 & 37 \end{pmatrix} = \begin{pmatrix} 12.5 & 10.25 \\ 10.25 & 9.25 \end{pmatrix}$$

The unit-length eigenvectors of Σ are $v_1 = (0.76045416, 0.64939162)^T$ and $v_2 = (-0.64939162, 0.76045416)^T$ with eigenvalues $w_1 = 21.25301161$ and $w_2 = 0.49698839$, respectively.

Plot of the centered dataset with principal components 1 (red) and 2 (blue):

Problem 44. (Clustering - Within and Across)

Suppose we wish to cluster some data by learning a function f such that $f_n = f(\boldsymbol{x}_n)$ is the cluster assignment for point $\boldsymbol{x}_n$. Show that finding f by minimizing $\mathcal{L}_W$, which is defined as the sum of squared distances between all pairs of points that are within the same cluster, i.e.,



$$\arg\min_{f} \mathcal{L}_W = \arg\min_{f} \sum_{n,m} \mathbb{I}\left[f_n = f_m\right] \left\|\boldsymbol{x}_n - \boldsymbol{x}_m\right\|^2$$

implicitly also maximizes the sum of squared distances between all pairs of points that are in different clusters. (Note: You can also show that the above is equivalent to the K-means objective!)

Solution:

Let $\hat{f}$ be the function learned by minimizing $\mathcal{L}_W$, which is defined as the sum of squared distances between all pairs of points that are within the same cluster, i.e.,

$$\hat{f} = \arg\min_{f} \sum_{n,m} \mathbb{I}\left(f_n = f_m\right) \left\|\mathbf{x}_n - \mathbf{x}_m\right\|^2$$

Note that we can write the indicator function as follows:

$$\mathbb{I}\left(f_n = f_m\right) = 1 - \mathbb{I}\left(f_n \neq f_m\right)$$

Thus, replacing the loss function, we obtain

$$\hat{f} = \arg\min_{f} \left[\sum_{n,m} \left\|\mathbf{x}_n - \mathbf{x}_m\right\|^2 - \sum_{n,m} \mathbb{I}\left(f_n \neq f_m\right) \left\|\mathbf{x}_n - \mathbf{x}_m\right\|^2\right]$$

We can rewrite it as follows since the first term does not involve the argument for minimization. Essentially, it remains to be a constant w.r.t the minimization.

$$\therefore \hat{f} = \sum_{n,m} \left\|\mathbf{x}_n - \mathbf{x}_m\right\|^2 - \arg\min_{f} \sum_{n,m} \mathbb{I}\left(f_n \neq f_m\right) \left\|\mathbf{x}_n - \mathbf{x}_m\right\|^2$$

$$\therefore \hat{f} = - \arg\min_{f} \sum_{n,m} \mathbb{I}\left(f_n \neq f_m\right) \left\|\mathbf{x}_n - \mathbf{x}_m\right\|^2$$

$$\therefore \hat{f} = \arg\max_{f} \sum_{n,m} \mathbb{I}\left(f_n \neq f_m\right) \left\|\mathbf{x}_n - \mathbf{x}_m\right\|^2$$

Hence, it turns out to be equivalent to maximizing the inter-cluster distances - the distance between points of different clusters.

Problem 45. (Eigenchangers!)

Suppose we wish to do PCA for an $N \times D$ matrix $\mathbf{X}$ and assume $D > N$. The traditional way to do PCA is to compute the eigenvectors of the covariance matrix $\mathbf{S} = \frac{1}{N}\mathbf{X}^\top\mathbf{X}$ (assuming centered data). Show that, if someone instead gives you an eigenvector $v \in \mathbb{R}^N$ of the matrix $\frac{1}{N}\mathbf{X}\mathbf{X}^\top$, you can use it to get an eigenvector $\boldsymbol{u} \in \mathbb{R}^D$ of $\mathbf{S}$. What is the advantage of this way of obtaining the eigenvectors of $\mathbf{S}$?

Solution:

Let X be the $N \times D$ data matrix. The covariance matrix is given by (assuming centered data) $S = \frac{1}{N} X^T X$. Let $T = \frac{1}{N} X X^T$. Suppose λ is an eigenvalue and $\mathbf{v}$ is an eigenvector of T, then my claim is that λ is also an eigenvalue of S with corresponding eigenvector being $X^T \mathbf{v}$. The proof follows:

$$
\begin{aligned}
T\mathbf{v} &= \lambda \mathbf{v} \\
\therefore \frac{1}{N} X X^T \mathbf{v} &= \lambda \mathbf{v} \\
\therefore \frac{1}{N} X^T X X^T \mathbf{v} &= \lambda X^T \mathbf{v} \\
\therefore S\left(X^T \mathbf{v}\right) &= \lambda \left(X^T \mathbf{v}\right)
\end{aligned}
$$

Let $\mathbf{u} := X^T \mathbf{v}$. We have $S\mathbf{u} = \lambda \mathbf{u}$. Therefore, $\mathbf{u}$ turns out to be the eigenvector for S corresponding to the eigenvalue λ. Thus, if we know the eigenvectors of matrix T, we can find the eigenvectors of matrix S by simple matrix multiplication, which is $O(ND)$. The advantage of using this approach to obtain the eigenvectors is that we will need to diagonalize the $N \times N$ matrix T instead of the $D \times D$ matrix S for getting eigenvectors. Note that we have been given $N < D$. Thus, obtaining eigenvectors in this manner is computationally cheaper when $D > N$. Also, note that we can kernelize the matrix T and then conduct eigendecomposition of the kernel matrix enabling us to do non-linear PCA.

Problem 46. (Soft k-means, Revisited)

(a) Consider the following optimization problem:

$$\max_{\mathbf{c} \in \mathbb{R}^k} \sum_{i=1}^{k} v_i \log\left(c_i\right) \qquad \text{s.t.} \qquad c_i > 0,\ \sum_{i=1}^{k} c_i = 1,$$

where $\mathbf{v} \in \mathbb{R}_+^k$ is a vector of non-negative weights. Check that the M-step of soft k-means includes solving such an optimization problem.

(b) Let $\mathbf{c}^\star = \frac{1}{\sum_i v_i} \mathbf{v}$. Verify that $\mathbf{c}^\star$ is a probability vector.

(c) Show that the optimization problem is equivalent to the following problem:

$$\min_{\mathbf{c} \in \mathbb{R}^k} \mathrm{D}_{\mathrm{KL}}\left(\mathbf{c}^\star \| \mathbf{c}\right) \qquad \text{s.t.} \qquad c_i > 0,\ \sum_{i=1}^{k} c_i = 1.$$

(d) Using the properties of KL divergence, prove that $\mathbf{c}^\star$ is indeed the solution to the optimization problem.

Solution:

(a) Check class notes.

(b) The components of $\mathbf{c}^\star$ are non-negative (since $\mathbf{v}$ is non-negative), and add up to 1.

(c) Since the optimization is over $\mathbf{c}$, it makes no difference if we divide it by a positive number or add/subtract terms that are not dependent on $\mathbf{c}$. We first divide the objective by $\sum_i v_i$ and then subtract from the sum $\sum_{i=1}^k c_i^\star \log c_i^\star$. We get

$$\sum_{i=1}^{k} c_i^\star \log\left(c_i\right) - \sum_{i=1}^{k} c_i^\star \log\left(c_i^\star\right) = \sum_{i=1}^{k} c_i^\star \log \frac{c_i}{c_i^\star} = -\mathrm{D}_{\mathrm{KL}}\left(\mathbf{c}^\star \| \mathbf{c}\right).$$

Thus maximizing the objective is equivalent to minimizing $\mathrm{D}_{\mathrm{KL}}\left(\mathbf{c}^\star \| \mathbf{c}\right)$.

(d) Since KL Divergence is always non-negative and is zero if and only if the two distributions are equal, we get that the optimal solution to the optimization problem is indeed $\mathbf{c} = \mathbf{c}^\star$.

Problem 47. (Sigmoidal function)

Let $Y \in \{0, 1\}$ denote a binary random variable that depends on k other random variables X_i as:

$$P\left(Y=1 \mid X_1=x_1, X_2=x_2, \ldots, X_k=x_k\right)=\sigma\left(\sum_{i=1}^k w_i x_i\right) \quad \text{where} \quad \sigma(z)=\frac{1}{1+e^{-z}}.$$

The real-valued parameters w_i in this CPT are known as weights. The so-called sigmoid function $\sigma(z)$ arises in many contexts. In neural networks, it models the probability that a neuron Y fires given its input from other neurons X_i; the weights w_i describe the connections between neurons. In statistics, the sigmoid function appears in models of logistic regression. Sketch the function $\sigma(z)$, and verify the following properties:

(a) $\sigma'(z) = \sigma(z)\sigma(-z)$.

(b) $\sigma(-z) + \sigma(z) = 1$.

(c) $L(\sigma(z)) = z$, where $L(p) = \log\left(\frac{p}{1-p}\right)$ is the log-odds function.

(d) $w_i = L\left(p_i\right)$, where $p_i = P\left(Y=1 \mid X_i=1, X_j=0 \text{ for all } j \neq i\right)$.

Solution:

(a)

$$\begin{aligned}\sigma'(z) &= \frac{0+1\cdot -e^{-z}}{\left(1+e^{-z}\right)^2}=\frac{1}{1+e^{-z}}\cdot\frac{-e^{-z}}{1+e^{-z}} \\ &= \frac{1}{1+e^{-z}}\cdot\frac{1}{e^{z}+1}=\sigma(z)\cdot\sigma(-z)\end{aligned}$$

(b)

$$\sigma(-z)+\sigma(z)=\frac{1}{1+e^{z}}+\frac{1}{1+e^{-z}}=\frac{1+e^{-z}+1+e^{z}}{1+e^{z}+e^{-z}+e^{0}}=1$$

(c)

$$L(\sigma(z))=\log\left(\frac{\sigma(z)}{1-\sigma(z)}\right)=\log\left(\frac{\frac{1}{1+e^{-z}}}{\frac{e^{-z}}{1+e^{-z}}}\right)=\log\left(\frac{1}{e^{-z}}\right)=\log\left(e^{z}\right)=z$$

(d)

$$L\left(p_i\right)=\log\left(\frac{P\left(Y=1 \mid X_i=1, X_j=0, \forall j\neq i\right)}{1-P\left(Y=1 \mid X_i=1, X_j=0, \forall j\neq i\right)}\right)=\log\left(\frac{\sigma\left(w_i\right)}{1-\sigma\left(w_i\right)}\right)=L\left(\sigma\left(w_i\right)\right)$$

According to the conclusion from (c), $L\left(\sigma\left(w_i\right)\right)=w_i$.

Problem 48. (A General Activation Function)

Consider the following activation function: $h(x) = x\sigma(\beta x)$ where σ denotes the sigmoid function $\sigma(z) = \frac{1}{1+\exp(-z)}$. Show that, for appropriately chosen values of β, this activation function can approximate (1) the linear activation function and (2) the ReLU activation function.

Solution:

Given $h(x) = x\sigma(\beta x)$ where σ is the usual sigmoid activation function.

$$h(x)=\frac{x}{1+\exp(-\beta x)}$$

Case 1: Linear approximation
Take $\beta = 0$, we will have

$$h(x) = \frac{x}{2} \quad \text{[Linear]}$$

Case 2: Approximating ReLU
Take $\beta \to \infty$, we will have $\exp(-\beta x) \to \infty$ for all $x < 0$ and $\exp(-\beta x) \to 0$ for $x \geq 0$. Thus, we get

$$h(x) = \begin{cases} 0 & \forall x < 0 \\ x & \forall x \geq 0 \end{cases}$$

Hence, we can approximate ReLU using the given activation function $h(x)$.

Problem 49. (Recurrent Neural Networks)

We saw feedforward artificial neural networks, which do not contain any cycles and for which the nodes do not maintain a persistent state over several runs. This exercise considers artificial neural networks with nodes that maintain a persistent state that can be updated. This kind of neural network is called a recurrent neural network (RNN). As an example, consider the following RNN with

$$y_t = Wx_t + Vs_t$$
$$s_{t+1} = y_t$$

from some initial state s_0, where t denotes the t th call of the RNN, i.e., x_t is the t^{th} input.

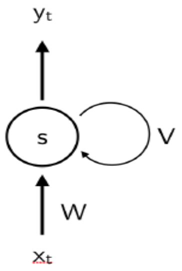


Figure 1

(a) What is the recurrent state in the RNN from Figure 1? Name one example that can be more naturally modeled with RNNs than with feedforward neural networks.

(b) As the state of an RNN changes over different runs of the RNN, the loss functions that we use for feedforward neural networks do not yield consistent results. For the given dataset X, please propose a loss function (based on the mean square loss function) for RNNs and justify why you chose this loss function.

(c) For a dataset $X := (x_t, y_t)_1^k$ (for some $k \in \mathbb{N}$), show how information is propagated by drawing a feedforward neural network that corresponds to the RNN from Figure 1 for $k = 3$. Recall that a feedforward neural network does not contain nodes with a persistent state. (Hint: unfold the RNN.)

Solution:

(a) The recurrent state is denoted s. In this case, it coincides with the output. Recurrent models are used to model data with temporal structure, e.g., time series, speech, and sound.

(b) We have a data $X = \{(x_t, y_t)\}$, where we assume that the data is ordered temporily. Thus, we define the loss function to be $L(U, W, s_0) = \sum_{t=1}^{T} (y(t) - f(x_t, s_{t-1}(U, W), U, W)$, where s_t is the previous reccurent state. The initial state s_0 needs to be specified, and the problem also depends on it.

(c) Check Figure 2.

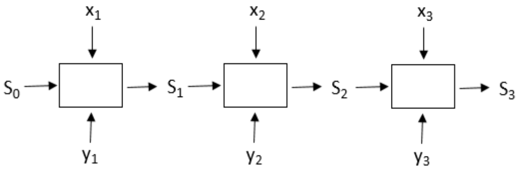


Figure 2

Problem 50. (Mixtures meet Neural Nets!)

Consider modeling some data $\{(\boldsymbol{x}_n, y_n)\}_{n=1}^{N}, \boldsymbol{x}_n \in \mathbb{R}^D, y_n \in \{0,1\}$, using a mixture of logistic regression models, where we model each binary label y_n by first picking one of the K logistic regression models, based on the value of a latent variable $z_n \sim$ multinoulli $(\pi_1, \ldots, \pi_K)$, and then generating y_n conditioned on z_n as $y_n \sim$ Bernoulli $\left[\sigma\left(\boldsymbol{w}_{z_n}^{\top} \boldsymbol{x}_n\right)\right]$.

Now, consider the marginal probability of the label $y_n = 1$, given $\boldsymbol{x}_n$, i.e., $p\left(y_n = 1 \mid \boldsymbol{x}_n\right)$, and show that this can also be thought of as the output of a neural network. Clearly specify the input layer, hidden layer(s), activations, output layer, and connection weights of this neural network.

Solution:

We have been given data $\{(\mathbf{x}_n, y_n)\}_{n=1}^{N}, \mathbf{x}_n \in \mathbb{R}^2, y_n \in \{0,1\}$. We are given

$$z_n \sim \text{multinoulli}\left(\pi_1, \pi_2, .., \pi_K\right)$$
$$y_n \sim \text{Bernoulli}\left(\sigma\left(w_{z_n}^T \mathbf{x}_n\right)\right)$$

We want to estimate $p\left(y_n = 1 \mid \mathbf{x}_n\right)$:

$$p\left(y_n = 1 \mid \mathbf{x}_n\right) = \sum_{k=1}^{K} p\left(y_n = 1, z_n = k \mid \mathbf{x}_n\right) = \sum_{k=1}^{K} p\left(y_n = 1 \mid z_n = k, \mathbf{x}_n\right) p\left(z_n = k\right) = \sum_{k=1}^{K} \sigma\left(w_k^T \mathbf{x}_n\right) \pi_k$$

$$p\left(y_n = 1 \mid \mathbf{x}_n\right) = \sum_{k=1}^{K} \sigma\left(w_k^T \mathbf{x}_n\right) \pi_k$$

We can think of this as a neural network in the following way:

- Input layer: $\left(x_1, x_2, .., x_D\right), \mathbf{x} \in \mathbb{R}^D$ is the input example.
- Hidden layer: We consider a single hidden layer of size K with each hidden node k having output $\sigma\left(w_k^T \mathbf{x}\right)$. The weight matrix will be $\mathbf{W} = \left[w_{ij}\right], i \in \{1, 2, \ldots, D\}, j \in \{1, 2, \ldots, K\}$ i.e. $\mathbf{W} = \begin{bmatrix} w_1^T & w_2^T & \ldots & w_K^T \end{bmatrix}$
- Final layer: The output layer will consist of only a single node whose output will be $p(y = 1 \mid \mathbf{x})$. The weight vector $\mathbf{U} = \begin{bmatrix} \pi_1 & \pi_2 & \ldots & \pi_K \end{bmatrix}^T$.