Introduction to Machine Learning 2019

Review Session

Ilija Bogunovic, Philippe Wenk

Tuesday 30th July, 2019

ETH Zürich

Naive Bayes

Generative Modeling

$\mathbf{x} = \text{Pixels}$

$y \in \{\text{cat}, \text{dog}\}$

Generative Modeling

Discriminative Modeling

Model $p(y|\boldsymbol{x})$

Generative Modeling

Model $p(y)$

Model $p(\boldsymbol{x}|y)$

Calculate $p(y|\boldsymbol{x}) = \frac{p(\boldsymbol{x}|y)p(y)}{\Sigma_y p(\boldsymbol{x}|y)p(y)}$

$\mathbf{x} = \text{Pixels}$

$y \in \{\text{cat}, \text{dog}\}$

Generative Modeling

Discriminative Modeling

Model $p(y|\boldsymbol{x})$

Generative Modeling

Model $p(y)$

Model $p(\boldsymbol{x}|y)$

Calculate $p(y|\boldsymbol{x}) = \frac{p(\boldsymbol{x}|y)p(y)}{\Sigma_y p(\boldsymbol{x}|y)p(y)}$

Naive Bayes

Assume $p(\boldsymbol{x}|y) = \Pi_{i=1}^{N} p(x_i|y)$

$\mathbf{x} = \text{Pixels}$

$y \in \{\text{cat}, \text{dog}\}$

4. Poisson Naive Bayes (21 points)

In this task we will use the Naive Bayes model for binary classification. Let $\mathcal{Y} = \{0, 1\}$ be the set of labels and $\mathcal{X} = \mathbb{N}^d$ a d-dimensional features space ($\mathbb{N} = \{0, 1, 2, \dots\}$). You are given a training set $D = \{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_n, y_n)\}$ of n labeled examples $(\mathbf{x}_i, y_i) \in \mathcal{X} \times \mathcal{Y}$.

(i) Is the Naive Bayes model a generative or a discriminative model? Justify your answer.

4. Poisson Naive Bayes (21 points)

In this task we will use the Naive Bayes model for binary classification. Let $\mathcal{Y} = \{0, 1\}$ be the set of labels and $\mathcal{X} = \mathbb{N}^d$ a d-dimensional features space ($\mathbb{N} = \{0, 1, 2, \dots\}$). You are given a training set $D = \{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_n, y_n)\}$ of n labeled examples $(\mathbf{x}_i, y_i) \in \mathcal{X} \times \mathcal{Y}$.

(i) Is the Naive Bayes model a generative or a discriminative model? Justify your answer.

Generative, since we model the joint density.

4. Poisson Naive Bayes (21 points)

In this task we will use the Naive Bayes model for binary classification. Let $\mathcal{Y} = \{0, 1\}$ be the set of labels and $\mathcal{X} = \mathbb{N}^d$ a d-dimensional features space ($\mathbb{N} = \{0, 1, 2, \dots\}$). You are given a training set $D = \{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_n, y_n)\}$ of n labeled examples $(\mathbf{x}_i, y_i) \in \mathcal{X} \times \mathcal{Y}$.

(i) Is the Naive Bayes model a generative or a discriminative model? Justify your answer.

Generative, since we model the joint density.

(ii) Let λ be a positive scalar, and assume that $z_1, \dots, z_m \in \mathbb{N}$ are m iid observations of a λ-Poisson distributed random variable. Find the maximum likelihood estimator for λ in this model. *(Hint: A λ-Poisson distributed random variable Z takes values $k \in \mathbb{N}$ with probability $P(Z = k) = e^{-\lambda}\frac{\lambda^k}{k!}$.)*

4. Poisson Naive Bayes (21 points)

In this task we will use the Naive Bayes model for binary classification. Let $\mathcal{Y} = \{0, 1\}$ be the set of labels and $\mathcal{X} = \mathbb{N}^d$ a d-dimensional features space ($\mathbb{N} = \{0, 1, 2, \dots\}$). You are given a training set $D = \{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_n, y_n)\}$ of n labeled examples $(\mathbf{x}_i, y_i) \in \mathcal{X} \times \mathcal{Y}$.

(i) Is the Naive Bayes model a generative or a discriminative model? Justify your answer.

Generative, since we model the joint density.

(ii) Let λ be a positive scalar, and assume that $z_1, \dots, z_m \in \mathbb{N}$ are m iid observations of a λ-Poisson distributed random variable. Find the maximum likelihood estimator for λ in this model. *(Hint: A λ-Poisson distributed random variable Z takes values $k \in \mathbb{N}$ with probability $P(Z = k) = e^{-\lambda}\frac{\lambda^k}{k!}$.)*

Likelihood: $p(z_1, \dots, z_m) = \Pi_{i=1}^{m} e^{-\lambda}\frac{\lambda^{z_i}}{z_i!}$

Hint: Can maximize directly, easier with log-likelihood.

ML-estimate: $\lambda = \frac{1}{m}\sum_{i=1}^{m} z_i$

(iii) Let's train a Poisson Naive Bayes classifier using maximum likelihood estimation. Define appropriate parameters $p_0, p_1 \in [0, 1]$, and vectors $\lambda_0, \lambda_1 \in \mathbb{R}^d$, and write down the joint distribution $P(X, Y)$ of the resulting model. *(Note that the following should be satisfied for the parameters:* $p_0 + p_1 = 1$, *and* λ_0, λ_1 *are vectors with non-negative components.)*

(iii) Let's train a Poisson Naive Bayes classifier using maximum likelihood estimation. Define appropriate parameters $p_0, p_1 \in [0, 1]$, and vectors $\lambda_0, \lambda_1 \in \mathbb{R}^d$, and write down the joint distribution $P(X, Y)$ of the resulting model. *(Note that the following should be satisfied for the parameters:* $p_0 + p_1 = 1$, *and* λ_0, λ_1 *are vectors with non-negative components.)*

Want to maximize data likelihood, i.e. $p(\mathcal{D}) = \Pi p(\boldsymbol{x}_i, y_i)$.

(iii) Let's train a Poisson Naive Bayes classifier using maximum likelihood estimation. Define appropriate parameters $p_0, p_1 \in [0, 1]$, and vectors $\lambda_0, \lambda_1 \in \mathbb{R}^d$, and write down the joint distribution $P(X, Y)$ of the resulting model. *(Note that the following should be satisfied for the parameters:* $p_0 + p_1 = 1$, *and* λ_0, λ_1 *are vectors with non-negative components.)*

Want to maximize data likelihood, i.e. $p(\mathcal{D}) = \Pi p(\boldsymbol{x}_i, y_i)$.

Key tricks:

1. Logarithm transforms products into sums
2. Create independent subproblems.

$\rightarrow$ Blackboard

one point: $p(x_i, y_i) = p_{y_i} \Pi_{j=1}^{d} \exp(-\lambda_{j,y_i}) \frac{\lambda_{j,y_i}^{x_{i,j}}}{x_{i,j}!}$

one point: $p(x_i, y_i) = p_{y_i} \Pi_{j=1}^{d} \exp(-\lambda_{j,y_i}) \frac{\lambda_{j,y_i}^{x_{i,j}}}{x_{i,j}!}$

all data:

$$\begin{aligned}\log(p(\mathcal{D})) = &\Sigma_{i:y_i=1} \left[\log p_1 + \Sigma_{j=1}^{d}[-\lambda_{j,1} + x_{i,j} \log(\lambda_{j,1}) - \log(x_{i,j}!)]\right] \\ &+ \Sigma_{i:y_i=0} \left[\log p_0 + \Sigma_{j=1}^{d}[-\lambda_{j,0} + x_{i,j} \log(\lambda_{j,0}) - \log(x_{i,j}!)]\right]\end{aligned}$$

one point: $p(x_i, y_i) = p_{y_i} \Pi_{j=1}^{d} \exp(-\lambda_{j,y_i}) \frac{\lambda_{j,y_i}^{x_{i,j}}}{x_{i,j}!}$

all data:

$$\log(p(\mathcal{D})) = \Sigma_{i:y_i=1} \left[\log p_1 + \Sigma_{j=1}^{d}[-\lambda_{j,1} + x_{i,j} \log(\lambda_{j,1}) - \log(x_{i,j}!)]\right] + \Sigma_{i:y_i=0} \left[\log p_0 + \Sigma_{j=1}^{d}[-\lambda_{j,0} + x_{i,j} \log(\lambda_{j,0}) - \log(x_{i,j}!)]\right]$$

Both d-dimensional sums are exponential distributions (see (ii))

$\lambda_{j,0} = \frac{1}{n_0} \Sigma_{i:y_i=0} x_{i,j}$
$\lambda_{j,1} = \frac{1}{n_1} \Sigma_{i:y_i=1} x_{i,j}$

one point: $p(x_i, y_i) = p_{y_i} \Pi_{j=1}^{d} \exp(-\lambda_{j,y_i}) \frac{\lambda_{j,y_i}^{x_{i,j}}}{x_{i,j}!}$

all data:

$$\log(p(\mathcal{D})) = \Sigma_{i:y_i=1} \left[\log p_1 + \Sigma_{j=1}^{d}[-\lambda_{j,1} + x_{i,j} \log(\lambda_{j,1}) - \log(x_{i,j}!)]\right] + \Sigma_{i:y_i=0} \left[\log p_0 + \Sigma_{j=1}^{d}[-\lambda_{j,0} + x_{i,j} \log(\lambda_{j,0}) - \log(x_{i,j}!)]\right]$$

Both d-dimensional sums are exponential distributions (see (ii))

$\lambda_{j,0} = \frac{1}{n_0} \Sigma_{i:y_i=0} x_{i,j}$
$\lambda_{j,1} = \frac{1}{n_1} \Sigma_{i:y_i=1} x_{i,j}$

p_0 and $p_1 = 1 - p_0$ can be found independently

$p_0 = \frac{n_1}{n_0+n_1}$
$p_1 = \frac{n_0}{n_0+n_1}$

(iv) Now, we want to use our trained model from (iii) to minimize the misclassification probability of a new observation $\mathbf{x} \in \mathcal{X}$, i.e. $y_{\text{pred}} = \text{argmax}_{y \in \mathcal{Y}} P(y|X = \mathbf{x})$. Show that the predicted label y_{pred} for $\mathbf{x}$ is determined by a hyperplane, i.e., that $y_{\text{pred}} = \left[\mathbf{a}^\top \mathbf{x} \geq b\right]$ for some $\mathbf{a} \in \mathbb{R}^d, b \in \mathbb{R}$.

How many labels do we have?

What happens at the decision boundary?

At boundary:

$$p(y=0|x) = p(y=1|x)$$

$$\frac{p(y=0,x)}{p(x)} = \frac{p(y=1,x)}{p(x)}$$

$$p(y=0,x) = p(y=1,x)$$

$$p_0 \Pi_{j=1}^{d} \exp(-\lambda_{j,0}) \frac{\lambda_{j,0}^{x_j}}{x_j!} = p_1 \Pi_{j=1}^{d} \exp(-\lambda_{j,1}) \frac{\lambda_{j,1}^{x_j}}{x_j!}$$

Take logarithm and reorder to obtain solution

2017 - 4

(v) Instead of simply predicting the most likely label, one can define a cost function $c : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$, such that $c(y_{\text{pred}}, y_{\text{true}})$ is the cost of predicting y_{pred} given that the true label is y_{true}. Define the Bayes optimal decision rule for a cost function $c(\cdot, \cdot)$, with respect to a distribution $P(X, Y)$.

(v) Instead of simply predicting the most likely label, one can define a cost function $c : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$, such that $c(y_{\text{pred}}, y_{\text{true}})$ is the cost of predicting y_{pred} given that the true label is y_{true}. Define the Bayes optimal decision rule for a cost function $c(\cdot, \cdot)$, with respect to a distribution $P(X, Y)$.

Bayes optimal decision rule: Minimize expected cost under your model

Formally: $y_{\text{Bayes}} = \arg\min_y \mathbb{E}_Y[c(Y, y)|X = x]$

(v) Instead of simply predicting the most likely label, one can define a cost function $c : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$, such that $c(y_{\text{pred}}, y_{\text{true}})$ is the cost of predicting y_{pred} given that the true label is y_{true}. Define the Bayes optimal decision rule for a cost function $c(\cdot, \cdot)$, with respect to a distribution $P(X, Y)$.

Bayes optimal decision rule: Minimize expected cost under your model

Formally: $y_{\text{Bayes}} = \arg\min_y \mathbb{E}_Y[c(Y, y)|X = x]$

(vi) Write down a cost function such that the corresponding decision rule that you have defined in (v) for this cost coincides with a decision rule that minimizes the misclassification probability, i.e., $y_{\text{pred}} = \text{argmax}_{y \in \mathcal{Y}} P(y|X = \mathbf{x})$.

(v) Instead of simply predicting the most likely label, one can define a cost function $c : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$, such that $c(y_{\text{pred}}, y_{\text{true}})$ is the cost of predicting y_{pred} given that the true label is y_{true}. Define the Bayes optimal decision rule for a cost function $c(\cdot, \cdot)$, with respect to a distribution $P(X, Y)$.

Bayes optimal decision rule: Minimize expected cost under your model

Formally: $y_{\text{Bayes}} = \arg\min_y \mathbb{E}_Y[c(Y, y)|X = x]$

(vi) Write down a cost function such that the corresponding decision rule that you have defined in (v) for this cost coincides with a decision rule that minimizes the misclassification probability, i.e., $y_{\text{pred}} = \text{argmax}_{y \in \mathcal{Y}} P(y|X = \mathbf{x})$.

1. Expectation must collapse to $Y = y$.
2. min has to become max.

Thus: $c(Y, y) = -\delta(Y - y)$

Alternatively: $c(Y, y) = \mathbb{1}(Y \neq y) = 1 - \delta(Y - y)$

Expectation Maximization

Dealing with Missing Data

Discriminative Modeling

Model $p(y|\boldsymbol{x})$

Generative Modeling

Model $p(y)$

Model $p(\boldsymbol{x}|y)$

Calculate $p(y|\boldsymbol{x}) = \frac{p(\boldsymbol{x}|y)p(y)}{\sum_y p(\boldsymbol{x}|y)p(y)}$

Unobserved y

Model $p(\boldsymbol{x}|\theta) = \sum_y p(y|\theta)p(\boldsymbol{x}|y,\theta)$

$\mathbf{x} = \text{Pixels}$

y is missing!

Dealing with Missing Data

Discriminative Modeling

Model $p(y|\boldsymbol{x})$

Generative Modeling

Model $p(y)$

Model $p(\boldsymbol{x}|y)$

Calculate $p(y|\boldsymbol{x}) = \frac{p(\boldsymbol{x}|y)p(y)}{\sum_y p(\boldsymbol{x}|y)p(y)}$

$\mathbf{x} = \text{Pixels}$

y is missing!

Unobserved y

Model $p(\boldsymbol{x}|\theta) = \sum_y p(y|\theta)p(\boldsymbol{x}|y,\theta)$

GMM $p(\boldsymbol{x}|\mu,\Sigma,\boldsymbol{w}) = \prod_{i=1}^{N}\sum_j w_j \mathcal{N}(x_i|\mu_j,\Sigma_j), \quad w_j \geq 0, \sum_j w_j = 1$

Dealing with Missing Data

Discriminative Modeling

Model $p(y|\boldsymbol{x})$

Generative Modeling

Model $p(y)$

Model $p(\boldsymbol{x}|y)$

Calculate $p(y|\boldsymbol{x}) = \frac{p(\boldsymbol{x}|y)p(y)}{\sum_y p(\boldsymbol{x}|y)p(y)}$

$\mathbf{x} = \text{Pixels}$

y is missing!

Unobserved y

Model $p(\boldsymbol{x}|\theta) = \sum_y p(y|\theta)p(\boldsymbol{x}|y,\theta)$

GMM $p(\boldsymbol{x}|\mu,\Sigma,\boldsymbol{w}) = \prod_{i=1}^{N}\sum_j w_j \mathcal{N}(x_i|\mu_j,\Sigma_j), \quad w_j \geq 0, \sum_j w_j = 1$

Maximize $\log p(\boldsymbol{x}|\theta)$ w.r.t. θ

Expectation-Maximization (EM)

Notation: Observed variables $\boldsymbol{x}$, latent variables $\boldsymbol{z}$, model parameters θ.

Summary: At every round k:

1. **E-step:** Calculate the expected complete data log-likelihood
   1.1 Define $Q^{(k)}(\boldsymbol{z}) := P(\boldsymbol{z}|\boldsymbol{x}, \theta_{k-1})$
   1.2 Set $\mathcal{L}^{(k)}(\theta) := \mathbb{E}_{Q^{(k)}}[\log P(\boldsymbol{x}, \boldsymbol{z}|\theta)]$
2. **M-step:** Solve

$$\theta_k = \arg\max_{\theta} \mathcal{L}^{(k)}(\theta)$$

Repeat until obtained model parameters converge.

7. Expectation Maximization (12 points)

Assume that we have a categorical distribution that represents drawing a red, green, or blue ball with probabilities $p_{\mathrm{r}} = 0.5, p_{\mathrm{g}} = \theta, p_{\mathrm{b}} = 0.5 - \theta$ respectively, where $\theta \in [0, 0.5]$ is an unknown parameter. After repeatedly and independently drawing n balls from this distribution, we know that the sum of red and green balls is $X_{\mathrm{r}} + X_{\mathrm{g}} = \alpha$, and the number of blue balls is $X_{\mathrm{b}} = n - \alpha := \beta$. We would like to use expectation maximization to estimate the value of θ.

If we knew the number of balls of each color to be x_{r}, x_{g}, x_{b}, we could compute the likelihood of our data set by noting that the counts follow a multinomial distribution,

$$P(X_{\mathrm{r}} = x_{\mathrm{r}}, X_{\mathrm{g}} = x_{\mathrm{g}}, X_{\mathrm{b}} = x_{\mathrm{b}} \mid \theta) = \frac{1}{Z} 0.5^{x_{\mathrm{r}}} \, \theta^{x_{\mathrm{g}}} \, (0.5 - \theta)^{x_{\mathrm{b}}},$$

and maximize this likelihood with respect to θ. Z is a normalization constant independent of θ.

7. Expectation Maximization (12 points)

Assume that we have a categorical distribution that represents drawing a red, green, or blue ball with probabilities $p_{\mathrm{r}} = 0.5, p_{\mathrm{g}} = \theta, p_{\mathrm{b}} = 0.5 - \theta$ respectively, where $\theta \in [0, 0.5]$ is an unknown parameter. After repeatedly and independently drawing n balls from this distribution, we know that the sum of red and green balls is $X_{\mathrm{r}} + X_{\mathrm{g}} = \alpha$, and the number of blue balls is $X_{\mathrm{b}} = n - \alpha := \beta$. We would like to use expectation maximization to estimate the value of θ.

If we knew the number of balls of each color to be $x_{\mathrm{r}}, x_{\mathrm{g}}, x_{\mathrm{b}}$, we could compute the likelihood of our data set by noting that the counts follow a multinomial distribution,

$$P(X_{\mathrm{r}} = x_{\mathrm{r}}, X_{\mathrm{g}} = x_{\mathrm{g}}, X_{\mathrm{b}} = x_{\mathrm{b}} \mid \theta) = \frac{1}{Z} 0.5^{x_{\mathrm{r}}} \, \theta^{x_{\mathrm{g}}} \, (0.5 - \theta)^{x_{\mathrm{b}}},$$

and maximize this likelihood with respect to θ. Z is a normalization constant independent of θ.

If we define the conditional distribution

$$Q^{(k)}(X_{\mathrm{r}}, X_{\mathrm{g}}, X_{\mathrm{b}}) := \mathbb{P}\left[X_{\mathrm{r}}, X_{\mathrm{g}}, X_{\mathrm{b}} \,\middle|\, \alpha, \beta, \theta^{(k)} \right],$$

then the expected log-likelihood mentioned above can be written as

$$\mathcal{L}^{(k)}(\theta) := \mathbb{E}_{Q^{(k)}} \left[\log P(X_{\mathrm{r}}, X_{\mathrm{g}}, X_{\mathrm{b}} \mid \theta) \right].$$

(i) If we define $\xi_{\mathrm{r}}^{(k)} := \mathbb{E}_{Q^{(k)}}[X_{\mathrm{r}}]$ and $\xi_{\mathrm{g}}^{(k)} := \mathbb{E}_{Q^{(k)}}[X_{\mathrm{g}}]$, write $\mathcal{L}^{(k)}$ as a function of $\xi_{\mathrm{r}}^{(k)}$, $\xi_{\mathrm{g}}^{(k)}$, α, β and θ.

(i) If we define $\xi_{\mathrm{r}}^{(k)} := \mathbb{E}_{Q^{(k)}}[X_{\mathrm{r}}]$ and $\xi_{\mathrm{g}}^{(k)} := \mathbb{E}_{Q^{(k)}}[X_{\mathrm{g}}]$, write $\mathcal{L}^{(k)}$ as a function of $\xi_{\mathrm{r}}^{(k)}$, $\xi_{\mathrm{g}}^{(k)}$, α, β and θ.

$$
\begin{aligned}
\mathcal{L}^{(k)}(\theta) &= \mathbb{E}_{Q^{(k)}}[\log P(X_r, X_g, X_b|\theta)] \\
&= \mathbb{E}_{Q^{(k)}}[-\log(Z) + X_r \log(0.5) + X_g \log(\theta) + X_b \log(0.5-\theta)] \\
&= \mathbb{E}_{Q^{(k)}}[-\log(Z)] + \log(0.5)\mathbb{E}_{Q^{(k)}}[X_r] + \log(\theta)\mathbb{E}_{Q^{(k)}}[X_g] \\
&\quad + \log(0.5-\theta)\mathbb{E}_{Q^{(k)}}[X_b] \\
&= \mathbb{E}_{Q^{(k)}}[-\log(Z)] + \xi_r^{(k)} \log(0.5) + \xi_g^{(k)} \log(\theta) + \beta \log(0.5-\theta)
\end{aligned}
$$

Key tricks:

1. Logarithm transforms products into sums.
2. Linearity of expectation.

(ii) Compute $\theta^{(k+1)}$ by maximizing $\mathcal{L}^{(k)}$ with respect to θ. (*M-step*)

(ii) Compute $\theta^{(k+1)}$ by maximizing $\mathcal{L}^{(k)}$ with respect to θ. (*M-step*)

$$\theta^{(k+1)} = \arg\max_{\theta} \mathcal{L}^{(k)}(\theta)$$

$$\frac{d\mathcal{L}^{(k)}(\theta)}{d\theta} = \xi_g^{(k)} \frac{1}{\theta} - \beta \frac{1}{0.5 - \theta}$$

$$\frac{d\mathcal{L}^{(k)}(\theta)}{d\theta} = 0 \implies \theta^{(k+1)} = \frac{0.5\xi_g^{(k)}}{\xi_g^{(k)} + \beta}.$$

(iii) Compute $\xi_{\mathrm{g}}^{(k)}$ by noting that X_{g} follows a binomial distribution. (*E-step*)

[*Hint: The mean of a binomial distribution with* m *trials and success probability* p *is* pm.]

(iii) Compute $\xi_g^{(k)}$ by noting that X_g follows a binomial distribution. (*E-step*)

[*Hint: The mean of a binomial distribution with m trials and success probability p is pm.*]

$X_g|X_g + X_r = \alpha$ follows a binomial distribution (green ball "success", red ball "failure") with $m = \alpha$ trials and probability of success:

$$p = \frac{p_g}{p_g + p_r} = \frac{\theta^{(k)}}{(0.5 + \theta^{(k)})}.$$

Hence, we have: $\xi_g^{(k)} = pm = \frac{\alpha\theta^{(k)}}{(0.5+\theta^{(k)})}$.

7. Expectation Maximization (15 points)

Kate is developing a new system and would like to understand the number of requests it has to serve. She counted the number of requests in 100 *ms* intervals and observed that these counts follow a bimodal distribution. Hence, she thought about using a mixture model with *two* components. As her observations are positive (they are counts), she decided to model the components with Poisson distributions. Remember that a Poisson distribution X with parameter $\lambda > 0$ assigns the following probabilities

$$P(X = x) = \begin{cases} \frac{\lambda^x}{x!} e^{-\lambda} & \text{if } \lambda \in \{0, 1, 2, 3, \ldots\} \\ 0 & \text{otherwise} \end{cases}.$$

7. Expectation Maximization (15 points)

Kate is developing a new system and would like to understand the number of requests it has to serve. She counted the number of requests in 100 *ms* intervals and observed that these counts follow a bimodal distribution. Hence, she thought about using a mixture model with *two* components. As her observations are positive (they are counts), she decided to model the components with Poisson distributions. Remember that a Poisson distribution X with parameter $\lambda > 0$ assigns the following probabilities

$$P(X = x) = \begin{cases} \frac{\lambda^x}{x!} e^{-\lambda} & \text{if } \lambda \in \{0, 1, 2, 3, \ldots\} \\ 0 & \text{otherwise} \end{cases}.$$

Parameter	Description
$\pi \in [0, 1]$	The probability of the point being sampled from the first component
$\lambda_1 > 0$	The parameter of the first mixture
$\lambda_2 > 0$	The parameter of the second mixture

(i) After running several iterations of the EM algorithm, Kate obtained the following posterior probabilities in the E-step using the current set of parameters (please note that we denote the probabilities by $Q(\cdot)$).

Observation x_i	$Q(Z = 1 \mid X = x_i)$	$Q(Z = 2 \mid X = x_i)$
$x_1 = 2$	0.9	0.1
$x_2 = 4$	0.8	0.2
$x_3 = 8$	0.3	0.7

Write down the objective that the M-step is optimizing with respect to the parameters π, λ_1 and λ_2. Remember that this is the expected log-likelihood of the complete data under the computed posterior distribution $Q(Z \mid X)$, or written formally

$$\mathbb{E}_{Q(Z|X)}[\log P(x_1, z_1, x_2, z_2, x_3, z_3)],$$

$$\mathbb{E}_{Q(Z|X)}[\log P(x_{1:3}, z_{1:3})]$$

$$= \mathbb{E}_{Q(Z|X)}\left[\log \prod_{i=1}^{3} P(x_i, z_i)\right] = \sum_{i=1}^{3} \mathbb{E}_{Q(Z|X)}[\log P(x_i, z_i)]$$

$$\mathbb{E}_{Q(Z|X)}[\log P(x_{1:3}, z_{1:3})]$$

$$= \mathbb{E}_{Q(Z|X)}\left[\log \prod_{i=1}^{3} P(x_i, z_i)\right] = \sum_{i=1}^{3} \mathbb{E}_{Q(Z|X)}[\log P(x_i, z_i)]$$

$$= \sum_{i=1}^{3}\sum_{j=1}^{2} \underbrace{Q(Z = z_i = j|X = x_i)}_{Q_{j|x_i}} \log \underbrace{P(x_i, z_i = j)}_{\pi_j P(X=x_i|\lambda_j)}$$

$$
\begin{aligned}
&\mathbb{E}_{Q(Z|X)}[\log P(x_{1:3}, z_{1:3})] \\
&= \mathbb{E}_{Q(Z|X)}\left[\log \prod_{i=1}^{3} P(x_i, z_i)\right] = \sum_{i=1}^{3} \mathbb{E}_{Q(Z|X)}[\log P(x_i, z_i)] \\
&= \sum_{i=1}^{3}\sum_{j=1}^{2} \underbrace{Q(Z = z_i = j|X = x_i)}_{Q_{j|x_i}} \log \underbrace{P(x_i, z_i = j)}_{\pi_j P(X=x_i|\lambda_j)} \\
&= \sum_{i=1}^{3} Q_{1|x_i} \log\left(\pi \frac{\lambda_1^{x_i}}{x_i!} e^{-\lambda_1}\right) + Q_{2|x_i} \log\left((1-\pi)\frac{\lambda_2^{x_i}}{x_i!} e^{-\lambda_2}\right)
\end{aligned}
$$

$$
\begin{aligned}
&\mathbb{E}_{Q(Z|X)}[\log P(x_{1:3}, z_{1:3})] \\
&= \mathbb{E}_{Q(Z|X)}\left[\log \prod_{i=1}^{3} P(x_i, z_i)\right] = \sum_{i=1}^{3} \mathbb{E}_{Q(Z|X)}[\log P(x_i, z_i)] \\
&= \sum_{i=1}^{3}\sum_{j=1}^{2} \underbrace{Q(Z = z_i = j|X = x_i)}_{Q_{j|x_i}} \log \underbrace{P(x_i, z_i = j)}_{\pi_j P(X=x_i|\lambda_j)} \\
&= \sum_{i=1}^{3} Q_{1|x_i} \log\left(\pi \frac{\lambda_1^{x_i}}{x_i!} e^{-\lambda_1}\right) + Q_{2|x_i} \log\left((1-\pi)\frac{\lambda_2^{x_i}}{x_i!} e^{-\lambda_2}\right) \\
&= \sum_{i=1}^{3} Q_{1|x_i}\left(\log(\pi) + x_i \log(\lambda_1) - \log(x_i!) - \lambda_1\right) \\
&\qquad + Q_{2|x_i}\left(\log(1-\pi) + x_i \log(\lambda_2) - \log(x_i!) - \lambda_2\right) \\
&:= \mathcal{L}(\lambda_1, \lambda_2, \pi)
\end{aligned}
$$

(ii) Perform the M-step update and compute the new parameters π, λ_1 and λ_2.

(ii) Perform the M-step update and compute the new parameters π, λ_1 and λ_2.

$$\frac{\partial \mathcal{L}(\lambda_1, \lambda_2, \pi)}{\partial \lambda_1} = \sum_{i=1}^{3} Q_{1|x_i} \left(\frac{x_i}{\lambda_1} - 1 \right) \tag{1}$$

$$\frac{\partial \mathcal{L}(\lambda_1, \lambda_2, \pi)}{\partial \lambda_2} = \sum_{i=1}^{3} Q_{2|x_i} \left(\frac{x_i}{\lambda_2} - 1 \right) \tag{2}$$

$$\frac{\partial \mathcal{L}(\lambda_1, \lambda_2, \pi)}{\partial \pi} = \frac{\sum_{i=1}^{3} Q_{1|x_i}}{\pi} - \frac{\sum_{i=1}^{3} Q_{2|x_i}}{1 - \pi} \tag{3}$$

By setting them to 0 (and substituting x_i's), we obtain $\lambda_1 = 3.7$, $\lambda_2 = 6.6$ and $\pi = 2/3$.

(ii) Perform the M-step update and compute the new parameters π, λ_1 and λ_2.

$$\frac{\partial \mathcal{L}(\lambda_1, \lambda_2, \pi)}{\partial \lambda_1} = \sum_{i=1}^{3} Q_{1|x_i} \left(\frac{x_i}{\lambda_1} - 1 \right) \tag{1}$$

$$\frac{\partial \mathcal{L}(\lambda_1, \lambda_2, \pi)}{\partial \lambda_2} = \sum_{i=1}^{3} Q_{2|x_i} \left(\frac{x_i}{\lambda_2} - 1 \right) \tag{2}$$

$$\frac{\partial \mathcal{L}(\lambda_1, \lambda_2, \pi)}{\partial \pi} = \frac{\sum_{i=1}^{3} Q_{1|x_i}}{\pi} - \frac{\sum_{i=1}^{3} Q_{2|x_i}}{1 - \pi} \tag{3}$$

By setting them to 0 (and substituting x_i's), we obtain $\lambda_1 = 3.7$, $\lambda_2 = 6.6$ and $\pi = 2/3$.

(iii) Kate would also like to experiment with different mixture components, but she does not know how to *compare* the resulting models. What strategy would you suggest to Kate? How should she choose a model? Please provide a principled approach.

(ii) Perform the M-step update and compute the new parameters π, λ_1 and λ_2.

$$\frac{\partial \mathcal{L}(\lambda_1, \lambda_2, \pi)}{\partial \lambda_1} = \sum_{i=1}^{3} Q_{1|x_i} \left(\frac{x_i}{\lambda_1} - 1 \right) \tag{1}$$

$$\frac{\partial \mathcal{L}(\lambda_1, \lambda_2, \pi)}{\partial \lambda_2} = \sum_{i=1}^{3} Q_{2|x_i} \left(\frac{x_i}{\lambda_2} - 1 \right) \tag{2}$$

$$\frac{\partial \mathcal{L}(\lambda_1, \lambda_2, \pi)}{\partial \pi} = \frac{\sum_{i=1}^{3} Q_{1|x_i}}{\pi} - \frac{\sum_{i=1}^{3} Q_{2|x_i}}{1 - \pi} \tag{3}$$

By setting them to 0 (and substituting x_i's), we obtain $\lambda_1 = 3.7$, $\lambda_2 = 6.6$ and $\pi = 2/3$.

(iii) Kate would also like to experiment with different mixture components, but she does not know how to *compare* the resulting models. What strategy would you suggest to Kate? How should she choose a model? Please provide a principled approach.

- Aim to maximize log-likelihood on validation set

Kernels

What are kernels?

Kernels represent a scalar product in a (potentially infinitely dimensional) feature space.

What are kernels?

Kernels represent a scalar product in a (potentially infinitely dimensional) feature space.

Requirements:

1. k has to be symmetric, i.e. $k(\boldsymbol{x}, \boldsymbol{x'}) = k(\boldsymbol{x'}, \boldsymbol{x})$.
2. k has to be positive definite, i.e. the Gram matrix $\boldsymbol{K}$ with $K_{i,j} = k(\boldsymbol{x}_i, \boldsymbol{x}_j)$ is positive semi-definite for all $\{x_1, \ldots, x_N\}, N \in \mathbb{N}$.

What are kernels?

Kernels represent a scalar product in a (potentially infinitely dimensional) feature space.

Requirements:

1. k has to be symmetric, i.e. $k(\boldsymbol{x}, \boldsymbol{x'}) = k(\boldsymbol{x'}, \boldsymbol{x})$.
2. k has to be positive definite, i.e. the Gram matrix $\boldsymbol{K}$ with $K_{i,j} = k(\boldsymbol{x}_i, \boldsymbol{x}_j)$ is positive semi-definite for all $\{x_1, \ldots, x_N\}, N \in \mathbb{N}$.

Equivalent to 2.: $\forall N \in \mathbb{N}, \forall \alpha_i \in \mathbb{R}, \forall \boldsymbol{x}_i \in \mathcal{X} : \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j k(\boldsymbol{x}_i, \boldsymbol{x}_j) \geq 0$

What are kernels?

Kernels represent a scalar product in a (potentially infinitely dimensional) feature space.

Requirements:

1. k has to be symmetric, i.e. $k(\boldsymbol{x}, \boldsymbol{x'}) = k(\boldsymbol{x'}, \boldsymbol{x})$.
2. k has to be positive definite, i.e. the Gram matrix $\boldsymbol{K}$ with $K_{i,j} = k(\boldsymbol{x}_i, \boldsymbol{x}_j)$ is positive semi-definite for all $\{x_1, \ldots, x_N\}, N \in \mathbb{N}$.

Equivalent to 2.: $\forall N \in \mathbb{N}, \forall \alpha_i \in \mathbb{R}, \forall \boldsymbol{x}_i \in \mathcal{X} : \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j k(\boldsymbol{x}_i, \boldsymbol{x}_j) \geq 0$

Mercer's theorem

There exist a feature expansion for every kernel.

The kernel trick

Idea: Make linear methods nonlinear by an implicit feature transformation

1. Reformulate original method to only contain scalar products $\boldsymbol{x}^T \boldsymbol{x}'$.
2. Subsitute scalar products by kernel function $\boldsymbol{x}^T \boldsymbol{x}' \to k(\boldsymbol{x}, \boldsymbol{x}')$.

What do we gain?

Can deal with a lot (potentially infinitely many) features quickly.

What do we lose?

Limited control of feature space. Features not even unique.

How do we construct kernels?

Start with legit kernel

- Constant kernel $k(\boldsymbol{x}, \boldsymbol{x}') = c > 0$
- Linear kernel $k(\boldsymbol{x}, \boldsymbol{x}') = \boldsymbol{x}^T \boldsymbol{x}'$
- Gaussian kernel $k(\boldsymbol{x}, \boldsymbol{x}') = \sigma_F^2 \exp(-\frac{1}{2l^2}||x - x'||^2)$

How do we construct kernels?

Start with legit kernel

- Constant kernel $k(\boldsymbol{x}, \boldsymbol{x}') = c > 0$
- Linear kernel $k(\boldsymbol{x}, \boldsymbol{x}') = \boldsymbol{x}^T \boldsymbol{x}'$
- Gaussian kernel $k(\boldsymbol{x}, \boldsymbol{x}') = \sigma_F^2 \exp(-\frac{1}{2l^2}||x - x'||^2)$

Combine using

- Positive scaling rule:
  k legit, $\alpha > 0 \implies \alpha k$ legit.
- Sum rule:
  k_1 and k_2 legit $\implies k_1 + k_2$ legit.
- Product rule:
  k_1 and k_2 legit $\implies k_1 k_2$ legit.
- Mapping rule:
  $k : (\mathcal{X} x \mathcal{X}) \to \mathbb{R}$ legit, $A : \tilde{\mathcal{X}} \to \mathcal{X}$ a map, then $k(A(x), A(x'))$ legit.

2017 - 5

a) For any kernel there exists an equivalent feature map.

2017 - 5

a) For any kernel there exists an equivalent feature map.

True (see Mercer's theorem)

2017 - 5

a) For any kernel there exists an equivalent feature map.

True (see Mercer's theorem)

b) For any feature map there exists an equivalent kernel.

2017 - 5

a) For any kernel there exists an equivalent feature map.

True (see Mercer's theorem)

b) For any feature map there exists an equivalent kernel.

True (see definitions)

a) For any kernel there exists an equivalent feature map.

True (see Mercer's theorem)

b) For any feature map there exists an equivalent kernel.

True (see definitions)

c) Let $M \in \mathbb{R}^{d \times d}$ be a diagonal matrix with non-zero diagonal elements, and define:

$$k(\mathbf{x}, \mathbf{x}') = \mathbf{x}^\top M \mathbf{x}', \ \forall \mathbf{x}, \mathbf{x}' \in \mathbb{R}^d,$$

then k is always a valid kernel.

a) For any kernel there exists an equivalent feature map.

True (see Mercer's theorem)

b) For any feature map there exists an equivalent kernel.

True (see definitions)

c) Let $M \in \mathbb{R}^{d \times d}$ be a diagonal matrix with non-zero diagonal elements, and define:

$$k(\mathbf{x}, \mathbf{x}') = \mathbf{x}^\top M \mathbf{x}', \ \forall \mathbf{x}, \mathbf{x}' \in \mathbb{R}^d,$$

then k is always a valid kernel.

False, e.g. negative identity

a) For any kernel there exists an equivalent feature map.

True (see Mercer's theorem)

b) For any feature map there exists an equivalent kernel.

True (see definitions)

c) Let $M \in \mathbb{R}^{d \times d}$ be a diagonal matrix with non-zero diagonal elements, and define:

$$k(\mathbf{x}, \mathbf{x}') = \mathbf{x}^\top M \mathbf{x}', \ \forall \mathbf{x}, \mathbf{x}' \in \mathbb{R}^d,$$

then k is always a valid kernel.

False, e.g. negative identity

Assume: SVM-classification and lower = lower or equal

For the rest of this question recall the definitions of the linear kernel and the polynomial kernel of degree 2,

$$k_{\text{linear}}(\mathbf{x}, \mathbf{x}') = \mathbf{x}^\top \mathbf{x}', \qquad k_{\text{poly}}(\mathbf{x}, \mathbf{x}') = (\mathbf{x}^\top \mathbf{x}' + 1)^2, \qquad \forall \mathbf{x}, \mathbf{x}' \in \mathbb{R}^d$$

d) Using the polynomial kernel we are always ensured to obtain a lower *training* error compared to the linear kernel.

Assume: SVM-classification and lower = lower or equal

For the rest of this question recall the definitions of the linear kernel and the polynomial kernel of degree 2,

$$k_{\text{linear}}(\mathbf{x}, \mathbf{x}') = \mathbf{x}^\top \mathbf{x}', \qquad k_{\text{poly}}(\mathbf{x}, \mathbf{x}') = (\mathbf{x}^\top \mathbf{x}' + 1)^2, \qquad \forall \mathbf{x}, \mathbf{x}' \in \mathbb{R}^d$$

d) Using the polynomial kernel we are always ensured to obtain a lower *training* error compared to the linear kernel.

True, since the polynomial kernel contains all linear features.

Assume: SVM-classification and lower = lower or equal

For the rest of this question recall the definitions of the linear kernel and the polynomial kernel of degree 2,

$$k_{\text{linear}}(\mathbf{x}, \mathbf{x}') = \mathbf{x}^\top \mathbf{x}', \qquad k_{\text{poly}}(\mathbf{x}, \mathbf{x}') = (\mathbf{x}^\top \mathbf{x}' + 1)^2, \qquad \forall \mathbf{x}, \mathbf{x}' \in \mathbb{R}^d$$

d) Using the polynomial kernel we are always ensured to obtain a lower *training* error compared to the linear kernel.

True, since the polynomial kernel contains all linear features.

e) Using the polynomial kernel we are always ensured to obtain a lower *generalization* error compared to the linear kernel.

Assume: SVM-classification and lower = lower or equal

For the rest of this question recall the definitions of the linear kernel and the polynomial kernel of degree 2,

$$k_{\text{linear}}(\mathbf{x}, \mathbf{x}') = \mathbf{x}^\top \mathbf{x}', \qquad k_{\text{poly}}(\mathbf{x}, \mathbf{x}') = (\mathbf{x}^\top \mathbf{x}' + 1)^2, \qquad \forall \mathbf{x}, \mathbf{x}' \in \mathbb{R}^d$$

d) Using the polynomial kernel we are always ensured to obtain a lower *training* error compared to the linear kernel.

True, since the polynomial kernel contains all linear features.

e) Using the polynomial kernel we are always ensured to obtain a lower *generalization* error compared to the linear kernel.

False, since the polynomial kernel contains more than just linear features.

Assume: SVM-classification and lower = lower or equal

For the rest of this question recall the definitions of the linear kernel and the polynomial kernel of degree 2,

$$k_{\text{linear}}(\mathbf{x}, \mathbf{x}') = \mathbf{x}^\top \mathbf{x}', \qquad k_{\text{poly}}(\mathbf{x}, \mathbf{x}') = (\mathbf{x}^\top \mathbf{x}' + 1)^2, \qquad \forall \mathbf{x}, \mathbf{x}' \in \mathbb{R}^d$$

d) Using the polynomial kernel we are always ensured to obtain a lower *training* error compared to the linear kernel.

True, since the polynomial kernel contains all linear features.

e) Using the polynomial kernel we are always ensured to obtain a lower *generalization* error compared to the linear kernel.

False, since the polynomial kernel contains more than just linear features.

f) Using any kernel we are always ensured to obtain a lower *training* error compared to the linear kernel.

Assume: SVM-classification and lower = lower or equal

For the rest of this question recall the definitions of the linear kernel and the polynomial kernel of degree 2,

$$k_{\text{linear}}(\mathbf{x}, \mathbf{x}') = \mathbf{x}^\top \mathbf{x}', \qquad k_{\text{poly}}(\mathbf{x}, \mathbf{x}') = (\mathbf{x}^\top \mathbf{x}' + 1)^2, \qquad \forall \mathbf{x}, \mathbf{x}' \in \mathbb{R}^d$$

d) Using the polynomial kernel we are always ensured to obtain a lower *training* error compared to the linear kernel.

True, since the polynomial kernel contains all linear features.

e) Using the polynomial kernel we are always ensured to obtain a lower *generalization* error compared to the linear kernel.

False, since the polynomial kernel contains more than just linear features.

f) Using any kernel we are always ensured to obtain a lower *training* error compared to the linear kernel.

g) Using any kernel we are always ensured to obtain a lower *generalization* error compared to the linear kernel.

Assume: SVM-classification and lower = lower or equal

For the rest of this question recall the definitions of the linear kernel and the polynomial kernel of degree 2,

$$k_{\text{linear}}(\mathbf{x}, \mathbf{x}') = \mathbf{x}^\top \mathbf{x}', \qquad k_{\text{poly}}(\mathbf{x}, \mathbf{x}') = (\mathbf{x}^\top \mathbf{x}' + 1)^2, \qquad \forall \mathbf{x}, \mathbf{x}' \in \mathbb{R}^d$$

d) Using the polynomial kernel we are always ensured to obtain a lower *training* error compared to the linear kernel.

True, since the polynomial kernel contains all linear features.

e) Using the polynomial kernel we are always ensured to obtain a lower *generalization* error compared to the linear kernel.

False, since the polynomial kernel contains more than just linear features.

f) Using any kernel we are always ensured to obtain a lower *training* error compared to the linear kernel.

g) Using any kernel we are always ensured to obtain a lower *generalization* error compared to the linear kernel.

Both False, since there exist both simpler (e.g. just linear kernel on one dimension) and more complex (e.g. polynomial) kernels.

h) For any kernel, the optimal solution to the kernelized SVM problem can always be written as a linear combination of the training points $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$.

h) For any kernel, the optimal solution to the kernelized SVM problem can always be written as a linear combination of the training points $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$.

$$\mathbf{k}_i = [y_1 k(\mathbf{x}_i, \mathbf{x}_1), \ldots, y_n k(\mathbf{x}_i, \mathbf{x}_n)]$$

- Prediction: For data point ***x*** predict label *y* as

$$\hat{y} = \text{sign}\left(\sum_{i=1}^{n} \alpha_i y_i k(\mathbf{x}_i, \mathbf{x})\right)$$

False

h) For any kernel, the optimal solution to the kernelized SVM problem can always be written as a linear combination of the training points $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$.

$$\mathbf{k}_i = [y_1 k(\mathbf{x}_i, \mathbf{x}_1), \ldots, y_n k(\mathbf{x}_i, \mathbf{x}_n)]$$

- Prediction: For data point ***x*** predict label *y* as

$$\hat{y} = \text{sign}\left(\sum_{i=1}^{n} \alpha_i y_i k(\mathbf{x}_i, \mathbf{x})\right)$$

False

i) The optimal solution to the original problem (without kernel trick or feature map) is the same as the optimal solution we would get using the linear kernel.

h) For any kernel, the optimal solution to the kernelized SVM problem can always be written as a linear combination of the training points $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$.

$$\mathbf{k}_i = [y_1 k(\mathbf{x}_i, \mathbf{x}_1), \ldots, y_n k(\mathbf{x}_i, \mathbf{x}_n)]$$

- Prediction: For data point ***x*** predict label *y* as

$$\hat{y} = \text{sign}\left(\sum_{i=1}^{n} \alpha_i y_i k(\mathbf{x}_i, \mathbf{x})\right)$$

False

i) The optimal solution to the original problem (without kernel trick or feature map) is the same as the optimal solution we would get using the linear kernel.

True, since the linear kernel is just the standard scalar product.

2017 - 5

(i) For $\mathbf{x}, \mathbf{x}' \in \mathbb{R}^d$, and $K(\mathbf{x}, \mathbf{x}') = (\mathbf{x}^T\mathbf{x}' + 1)^2$, find a feature map $\phi(\mathbf{x})$, such that $k(\mathbf{x}, \mathbf{x}') = \phi(\mathbf{x})^\top \phi(\mathbf{x}')$.

(i) For $\mathbf{x}, \mathbf{x}' \in \mathbb{R}^d$, and $K(\mathbf{x}, \mathbf{x}') = (\mathbf{x}^T\mathbf{x}' + 1)^2$, find a feature map $\phi(\mathbf{x})$, such that $k(\mathbf{x}, \mathbf{x}') = \phi(\mathbf{x})^\top \phi(\mathbf{x}')$.

Key trick: Look for symmetric groups. Easier with notation $\boldsymbol{x}' \to \boldsymbol{y}$.

$$
\begin{aligned}
k(\boldsymbol{x}, \boldsymbol{y}) &= (\boldsymbol{x}^T \boldsymbol{y} + 1)^2 \\
&= (\Sigma_i x_i y_i + 1)^2 \\
&= 1 + 2\Sigma_i x_i y_i + \Sigma_{i,j}(x_i y_i)(x_j y_j) \\
&= 1 + \Sigma_i(\sqrt{2}x_i)(\sqrt{2}y_i) + \Sigma_{i,j}(x_i x_j)(y_i y_j)
\end{aligned}
$$

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

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

Gram matrix: $\boldsymbol{K}_{i,j} = k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \boldsymbol{\phi}(\boldsymbol{x}_i)^T \boldsymbol{\phi}(\boldsymbol{x}_j)$

Calculate to get: $\boldsymbol{K} = \begin{pmatrix} 50 & 2 \\ 2 & 2 \end{pmatrix}$