CS 446 MJT — Homework 3

haoyuan9

Version 1

Instructions.

- Homework is due **Tuesday, March 12, at 11:59pm**; no late homework accepted.
- Everyone must submit individually at gradescope under `hw3` and `hw3code`.
- The "written" submission at `hw3` **must be typed**, and submitted in any format gradescope accepts (to be safe, submit a PDF). You may use LaTeX, markdown, google docs, MS word, whatever you like; but it must be typed!
- When submitting at `hw3`, gradescope will ask you to mark out boxes around each of your answers; please do this precisely!
- Please make sure your NetID is clear and large on the first page of the homework.
- Your solution **must** be written in your own words. Please see the course webpage for full academic integrity information. Briefly, you may have high-level discussions with at most 3 classmates, whose NetIDs you should place on the first page of your solutions, and you should cite any external reference you use; despite all this, your solution must be written in your own words.
- We reserve the right to reduce the auto-graded score for `hw3code` if we detect funny business (e.g., rather than implementing an algorithm, you keep re-submitting the assignment to the auto-grader, eventually completing a binary search for the answers).
- There are **no regrade requests** on `hw3code`, which is the code auto-grader; however, you can re-submit and re-grade as many times as you like before the deadline! Start early and report any issues on piazza!
- Methods and functions in the template and utility code include docstrings describing the inputs and outputs. The autograder relies on correct implementations of these methods. Follow the docstrings to avoid failing tests.

1. **The ln-sum-exp and cross entropy.**

(a) Given $\boldsymbol{z} \in \mathbb{R}^k$, prove that $g(\boldsymbol{z}) = \ln \sum_{j=1}^k \exp(z_j)$ is convex.

Hint: prove that the Hessian matrix is positive semi-definite, meaning $\nabla^2 g(\boldsymbol{z}) \succeq 0$.

(b) Recall that given a data example $(\boldsymbol{x}, y)$ where $\boldsymbol{x} \in \mathbb{R}^d$ and $y \in \{1, 2, \ldots, k\}$, and a classifier $f : \mathbb{R}^d \to \mathbb{R}^k$, the cross entropy loss is defined as

$$\ell_{\text{ce}}\big(f(\boldsymbol{x}), y\big) = -\ln\left(\frac{\exp\big(f(\boldsymbol{x})_y\big)}{\sum_{j=1}^k \exp\big(f(\boldsymbol{x})_j\big)}\right) = -f(\boldsymbol{x})_y + \ln \sum_{j=1}^k \exp\big(f(\boldsymbol{x})_j\big).$$

Let data examples $\big((\boldsymbol{x}_i, y_i)\big)_{i=1}^n$ be given, where $\boldsymbol{x}_i \in \mathbb{R}^d$ and $y_i \in \{1, 2, \ldots, k\}$. Consider the linear predictor $\boldsymbol{x} \mapsto \boldsymbol{W}\boldsymbol{x}$, where $\boldsymbol{W} \in \mathbb{R}^{k \times d}$. Prove that the empirical risk

$$\widehat{\mathcal{R}}(\boldsymbol{W}) = \frac{1}{n} \sum_{i=1}^n \ell_{\text{ce}}(\boldsymbol{W}\boldsymbol{x}, y)$$

is convex.

Hint: use part (a) and the fact that convexity is preserved under affine composition and nonnegative combination. (It doesn't matter that this part uses matrix variables; this affine composition property holds for convex functions over matrices, and then when applying the previous part, its input is a vector after the affine transformation.)

(c) For $\boldsymbol{z} \in \mathbb{R}^k$ and $r > 0$, let $g_r(\boldsymbol{z}) = \frac{1}{r} \ln \sum_{j=1}^k \exp(r z_j)$. Prove that

$$\lim_{r \to \infty} g_r(\boldsymbol{z}) = \max_{1 \le j \le k} z_j.$$

(d) As a corollary, for $z \in \mathbb{R}$ and $r > 0$, the logistic loss $\ell(z) = \ln(1 + \exp(-z))$, and $\ell_r(z) = \frac{1}{r} \ln(1 + \exp(-rz))$, prove that

$$\lim_{r \to \infty} \ell_r(z) = \max\{0, -z\} = \text{ReLU}(-z).$$

Solution. *(Your solution here.)*

(a) **Proof:**

$H(z) = \bigtriangledown^2 g(z) = [\frac{-\exp(z_i)\exp(z_j)}{[\sum_{i=1}^k \exp(z_i)]^2}]_{ij}$, where $H(z)_{ij} = \frac{-\exp(z_i)\exp(z_j)}{[\sum_{i=1}^k \exp(z_i)]^2}$

For any $x = \begin{bmatrix} x_1 & x_2 & \cdots & x_k \end{bmatrix}^\intercal$, we have

$$x^\intercal H(z) x = \frac{\sum_{i=1}^k \sum_{j=1}^k x_i x_j \exp(z_i)\exp(z_j)}{(\sum_{i=1}^k \exp(z_i))^2} = (\frac{\sum_{i=1}^k x_i \exp(z_i)}{\sum_{i=1}^k \exp(z_i)})^2 \ge 0$$

$H(z)$ is positive semi-definite.

(b) **Proof:**

$$R(W) = \frac{1}{n} \sum_{i=1}^n \{[-(Wx_i)_{y_i}] + \ln \sum_{j=1}^k \exp((Wx_i)_j)\}$$

As stated above, $\ln \sum_{j=1}^k \exp(x_i)$ is convex.

Assume $h(x) = -x$, obviously, $\bigtriangledown^2 h(x) \ge 0$.

So, -x is convex. And because convexity preserves after affine composition and summation, We have that R(W) is also convex.

(c) **Proof:**

Assume $\max_{1\leq j\leq k} z_j = z_{max}$. We have

$$\begin{aligned}
\lim_{r\to\infty} g_r(z) &= \lim_{r\to\infty} \frac{1}{r} \ln \exp{(rz_{max})} \sum_{j=1}^{k} \exp{(r(z_j - z_{max}))} \\
&= \lim_{r\to\infty} \frac{1}{r} [\ln \exp{(rz_{max})} + \ln \sum_{j=1}^{k} \exp{(r(z_j - z_{max}))}] \\
&= \lim_{r\to\infty} [z_{max} + \frac{1}{r} \ln \sum_{j=1}^{k} \exp{(r(z_j - z_{max}))}] \\
&= z_max + \lim_{r\to\infty} \frac{1}{r} \ln(\sum_{j\neq max} \exp{(r(z_j - z_{max}))} + 1) \\
&= z_{max}
\end{aligned}$$

(d) **Proof:**

When $Z \geq 0$, then $-rz \leq 0$.

Then we have $\lim_{r\to\infty} l_r(z) = \frac{1}{r}\ln(1) = 0$

When $Z < 0$, $-rz > 0$.

Then we have

$$\begin{aligned}
\lim_{r\to\infty} l_r(z) &= \lim_{r\to\infty} \frac{1}{r} \ln(1 + \exp{(-rz)}) \\
&= \lim_{r\to\infty} \frac{1}{r} \ln(\exp{(-rz)}) \\
&= -z
\end{aligned}$$

Q.E.D.

2. **On initialization.**

Consider a 2-layer network

$$f(\boldsymbol{x}; \boldsymbol{W}, \boldsymbol{v}) = \sum_{j=1}^{m} v_j \sigma\left(\langle \boldsymbol{w}_j, \boldsymbol{x} \rangle\right),$$

where $\boldsymbol{x} \in \mathbb{R}^d$, $\boldsymbol{W} \in \mathbb{R}^{m \times d}$ with rows $\boldsymbol{w}_j^\top$, and $\boldsymbol{v} \in \mathbb{R}^m$. For simplicity, the network has a single output, and bias terms are omitted.

Given a data example $(\boldsymbol{x}, y)$ and a loss function ℓ, consider the empirical risk

$$\widehat{\mathcal{R}}(\boldsymbol{W}, \boldsymbol{v}) = \ell\left(f(\boldsymbol{x}; \boldsymbol{W}, \boldsymbol{v}), y\right).$$

Only a single data example will be considered in this problem; the same analysis extends to multiple examples by taking averages.

(a) For each $1 \leq j \leq m$, derive $\partial \widehat{\mathcal{R}} / \partial v_j$ and $\partial \widehat{\mathcal{R}} / \partial \boldsymbol{w}_j$.

(b) Consider gradient descent which starts from some $\boldsymbol{W}^{(0)}$ and $\boldsymbol{v}^{(0)}$, and at step $t \geq 0$, updates the weights for each $1 \leq j \leq m$ as follows:

$$\boldsymbol{w}_j^{(t+1)} = \boldsymbol{w}_j^{(t)} - \eta \frac{\partial \widehat{\mathcal{R}}}{\partial \boldsymbol{w}_j^{(t)}}, \qquad \text{and} \qquad v_j^{(t+1)} = v_j^{(t)} - \eta \frac{\partial \widehat{\mathcal{R}}}{\partial v_j^{(t)}}.$$

Suppose there exists two hidden units $p, q \in \{1, 2, \ldots, m\}$ such that $\boldsymbol{w}_p^{(0)} = \boldsymbol{w}_q^{(0)}$ and $v_p^{(0)} = v_q^{(0)}$. Prove by induction that for any step $t \geq 0$, it holds that $\boldsymbol{w}_p^{(t)} = \boldsymbol{w}_q^{(t)}$ and $v_p^{(t)} = v_q^{(t)}$.

Remark: as a result, if the neural network is initialized symmetrically, then such a symmetry may persist during gradient descent, and thus the representation power of the network will be limited.

(c) Random initialization is a good way to break symmetry. Moreover, proper random initialization also preserves the squared norm of the input, as formalized below.

First consider the identity activation $\sigma(z) = z$. For each $1 \leq j \leq m$ and $1 \leq k \leq d$, initialize $w_{j,k}^{(0)} \sim \mathcal{N}(0, 1/m)$ (i.e., normal distribution with mean $\mu = 0$ and variance $\sigma^2 = 1/m$). Prove that

$$\mathbb{E}\left[\left\|\boldsymbol{W}^{(0)}\boldsymbol{x}\right\|_2^2\right] = \|\boldsymbol{x}\|_2^2.$$

Next consider the ReLU activation $\sigma_r(z) = \max\{0, z\}$. For each $1 \leq j \leq m$ and $1 \leq k \leq d$, initialize $w_{j,k}^{(0)} \sim \mathcal{N}(0, 2/m)$. Prove that

$$\mathbb{E}\left[\left\|\sigma_r(\boldsymbol{W}^{(0)}\boldsymbol{x})\right\|_2^2\right] = \|\boldsymbol{x}\|_2^2.$$

Hint: linear combinations of Gaussians are again Gaussian! For the second part (with ReLU), consider the symmetry of a Gaussian around 0.

Solution. *(Your solution here.)*

(a) **Proof:**

$$\frac{\partial R}{\partial v_j} = \frac{\partial R}{\partial f}\frac{\partial f}{\partial v_j} = \frac{\partial R}{\partial f}\sigma(< w_j, x >)$$

$$\frac{\partial R}{\partial w_j} = \frac{\partial R}{\partial f}\frac{\partial f}{\partial w_j} = \frac{\partial R}{\partial f} v_j \sigma^{'}(< w_j, x >)x$$

(b) **Proof:** When $t = 0$

$$\frac{\partial R}{\partial v_p^0} = \frac{\partial R}{\partial f}\sigma(< w_p^0, x >) = \frac{\partial R}{\partial f}\sigma(< w_q^0, x >) = \frac{\partial R}{\partial v_q^0}$$

So we get $\frac{\partial R}{\partial v_p^1} = \frac{\partial R}{\partial v_q^1}$.

$$\frac{\partial R}{\partial w_p^0} = \frac{\partial R}{\partial f} v_p^0 \sigma^{'}(< w_p^0, x >)x = \frac{\partial R}{\partial f} v_q^0 \sigma^{'}(< w_q^0, x >)x = \frac{\partial R}{\partial w_q^0}$$

So we get $\frac{\partial R}{\partial w_p^1} = \frac{\partial R}{\partial w_q^1}$.

Assume when $t = k, w_p^k = w_q^k, v_p^k = v_q^k$.

When $t = k + 1$

$$\frac{\partial R}{\partial v_p^k} = \frac{\partial R}{\partial f}\sigma(< w_p^k, x >) = \frac{\partial R}{\partial f}\sigma(< w_q^k, x >) = \frac{\partial R}{\partial v_q^k}$$

$$\frac{\partial R}{\partial w_p^k} = \frac{\partial R}{\partial f} v_p^k \sigma^{'}(< w_p^k, x >)x = \frac{\partial R}{\partial f} v_q^k \sigma^{'}(< w_q^k, x >)x = \frac{\partial R}{\partial w_q^k}$$

Hence,

$$v_p^{k+1} = v_q^{k+1}$$
$$w_p^{k+1} = w_q^{k+1}$$

Q.E.D.

(c) **Proof:**

$$\begin{aligned}
\mathbb{E}\left[\left\|\boldsymbol{W}^{(0)}\boldsymbol{x}\right\|_2^2\right] &= \mathbb{E}\Big[\sum_{j=1}^{m}(\sum_{k=1}^{d} w_{jk}x_k)^2\Big] \\
&= \sum_{j=1}^{m}\Big[Var(\sum_{k=1}^{d} w_{jk}x_k) + \mathbb{E}^2(\sum_{k=1}^{d} w_{jk}x_k)\Big] \\
&= \sum_{j=1}^{m}\sum_{k=1}^{d} x_k^2 Var(w_{jk}) \\
&= \sum_{j=1}^{m}\sum_{k=1}^{d} x_k^2 \frac{1}{m} \\
&= \sum_{k=1}^{d} x_k^2 \\
&= \left\|x\right\|_2^2
\end{aligned}$$

$$\mathbb{E}\left[\left\|\boldsymbol{W}^{(0)}\boldsymbol{x}\right\|_2^2\right] = \mathbb{E}\Big[\sum_{j=1}^{m}\sigma_r^2(\sum_{k=1}^{d} w_{jk}x_k)\Big]$$

Because

$$\mathbb{E}\left[\sum_{k=1}^{d} w_{jk}x_k\right] = 0$$

Distribution of $\sum_{k=1}^{d} w_{jk}x_k$ is symmetric around zero.
So

$$\mathbb{E}\left[\sigma_r^2(\sum_{k=1}^{d} w_{jk}x_k)\right] = \frac{1}{2}\mathbb{E}\left[(\sum_{k=1}^{d} w_{jk}x_k)^2\right] = \frac{1}{2}Var\left[\sum_{k=1}^{d} w_{jk}x_k\right]$$

So we have

$$\begin{aligned}
\mathbb{E}\left[\left\|\boldsymbol{W}^{(0)}\boldsymbol{x}\right\|_2^2\right] &= \frac{1}{2}Var\left[\sum_{k=1}^{d} w_{jk}x_k\right] \\
&= \frac{1}{2}\sum_{j=1}^{m}\sum_{k=1}^{d} x_k^2 Var(w_{jk}) \\
&= \sum_{k=1}^{d} x_k^2 \\
&= \left\|x\right\|_2^2
\end{aligned}$$

3. **ResNet.**

In this problem, you will implement a simplified ResNet. You do not need to change arguments which are not mentioned here (but you of course could try and see what happens).

(a) Implement a class `Block`, which is a building block of ResNet. It is described in (He et al., 2016) Figure 2.

The input to `Block` is of shape (N, C, H, W), where N denotes the batch size, C denotes the number of channels, and H and W are the height and width of each channel. For each data example $\boldsymbol{x}$ with shape (C, H, W), the output of `block` is

$$\texttt{Block}(\boldsymbol{x}) = \sigma_r\left(\boldsymbol{x} + f(\boldsymbol{x})\right),$$

where σ_r denotes the ReLU activation, and $f(\boldsymbol{x})$ also has shape (C, H, W) and thus can be added to $\boldsymbol{x}$. In detail, f contains the following layers.

i. A `Conv2d` with C input channels, C output channels, kernel size 3, stride 1, padding 1, and no bias term.

ii. A `BatchNorm2d` with C features.

iii. A ReLU layer.

iv. Another `Conv2d` with the same arguments as i above.

v. Another `BatchNorm2d` with C features.

Because 3×3 kernels and padding 1 are used, the convolutional layers do not change the shape of each channel. Moreover, the number of channels are also kept unchanged. Therefore $f(\boldsymbol{x})$ does have the same shape as $\boldsymbol{x}$.

Additional instructions are given in doscstrings in `hw3.py`.

(b) Explain why a `Conv2d` layer does not need a bias term if it is followed by a `BatchNorm2d` layer.

(c) Implement a (shallow) `ResNet` consists of the following parts:

i. A `Conv2d` with 1 input channel, C output channels, kernel size 3, stride 2, padding 1, and no bias term.

ii. A `BatchNorm2d` with C features.

iii. A ReLU layer.

iv. A `MaxPool2d` with kernel size 2.

v. A `Block` with C channels.

vi. An `AdaptiveAvgPool2d` which for each channel takes the average of all elements.

vii. A `Linear` with C inputs and 10 outputs.

Additional instructions are given in doscstrings in `hw3.py`.

(d) Train a `ResNet` with 16 channels on the data given by `hw3_utils.torch_digits()`, using the cross entropy loss and SGD with learning rate 0.005 and batch size 16, for 30 epochs. You can just use your `fit_and_validate()` in `hw2`. Plot the epochs vs training and validation cross entropy losses. Since there is some inconsistency due to random initialization, try 3 runs and have 3 plots.

Solution. *(Your solution here.)*

(a)

(b) From the expression of batch normalization, we can see it detract the mean of input from the output. So the bias term will be diminished, and it has no effect on the output.

(c)

(d)

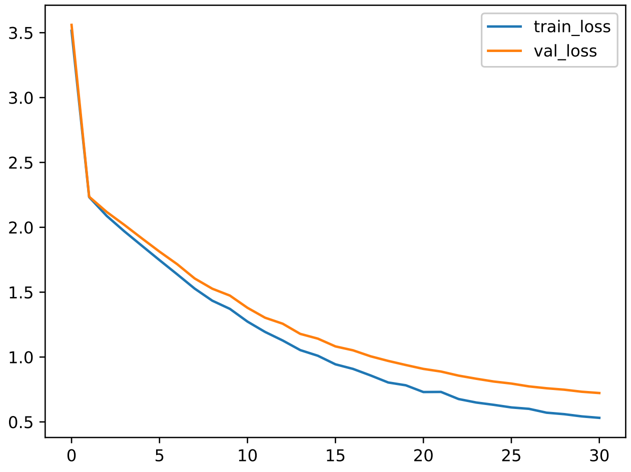

Figure 1: First run

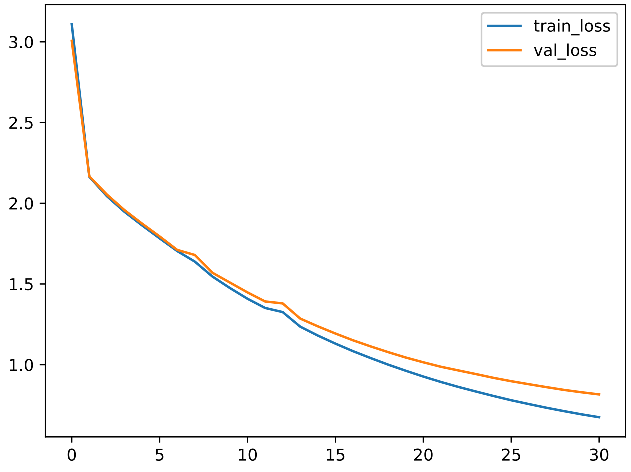

Figure 2: Second run

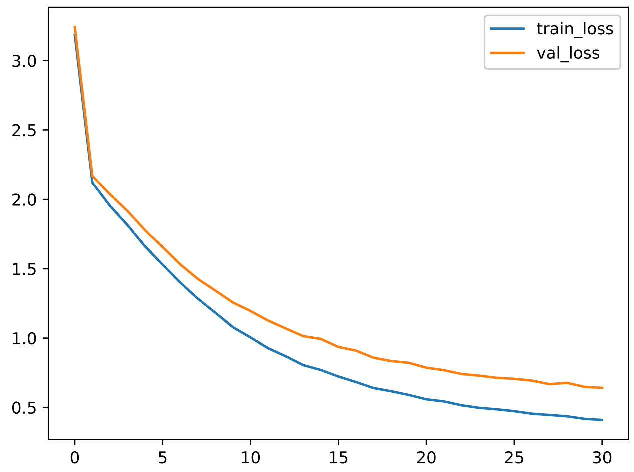

Figure 3: Third run

4. **Kernel properties.**

(a) Let $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ denote a symmetric positive semi-definite matrix of rank r. Prove that there exists n vectors $\boldsymbol{z}_1, \boldsymbol{z}_2, \ldots, \boldsymbol{z}_n \in \mathbb{R}^r$, such that $A_{i,j} = \langle \boldsymbol{z}_i, \boldsymbol{z}_j \rangle$.

Hint: use the eigendecomposition $\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^\top$.

(b) On the other hand, given n vectors $\boldsymbol{z}_1, \boldsymbol{z}_2, \ldots, \boldsymbol{z}_n \in \mathbb{R}^m$, prove that the matrix $\boldsymbol{A}$ defined by $A_{i,j} = \langle \boldsymbol{z}_i, \boldsymbol{z}_j \rangle$ is positive semi-definite.

Remark: note that the rank of $\boldsymbol{A}$ is at most m, and it could be strictly less than m. In particular, m could be larger than n.

(c) Using (a) and (b), prove that if $K_1(\boldsymbol{x}, \boldsymbol{y})$ and $K_2(\boldsymbol{x}, \boldsymbol{y})$ are kernels, then $K(\boldsymbol{x}, \boldsymbol{y}) = K_1(\boldsymbol{x}, \boldsymbol{y})K_2(\boldsymbol{x}, \boldsymbol{y})$ is also a kernel.

(d) Using (c), prove that $K(\boldsymbol{x}, \boldsymbol{y}) = (1 + \langle \boldsymbol{x}, \boldsymbol{y} \rangle)^r$ is a kernel for any positive integer r. (It is the polynomial kernel with degree r.)

(e) Assume $K(\boldsymbol{x}, \boldsymbol{y}) = \exp(-\|\boldsymbol{x} - \boldsymbol{y}\|_2^2/2\sigma^2)$ is a kernel in the 1-dimensional case (i.e., $x, y \in \mathbb{R}$). Using (c), prove that $K(\boldsymbol{x}, \boldsymbol{y})$ is indeed a kernel for any dimension. (It is the Gaussian / RBF kernel.)

Solution. *(Your solution here.)*

(a) **Proof:**

Because $rank(A) = r$, A has r distinct eigenvalues.

And because A is a real symmetric positive semi-definite matrix, we can do the eigendecomposition of it as

$$A = USU^\intercal$$

where S is a diagonal matrix whose elements $\lambda_i \geq 0$ are eigenvalues of A, $U \in R^{n\times n}, S \in R^{n\times n}$.

Assume the last $n - r$ eigenvalues are 0, then

$$A = USU^\intercal$$

where $U \in R^{n\times r}, S \in R^{r\times r}$

Because all diagonal elements in S are greater than 0, we can decompose S as

$$S = XX^\intercal$$

where $X = \left[\sqrt{\lambda_i}\right]_{diag}$

So

$$A = UX(UX)^\intercal$$

where

$$UX = \begin{bmatrix} z_1 \\ z_2 \\ \vdots \\ z_n \end{bmatrix}$$

(b) **Proof:** Assume

$$U = \begin{bmatrix} z_1 \\ z_2 \\ \vdots \\ z_n \end{bmatrix}$$

and $A = UU^\intercal$.

Then for any $x = \begin{bmatrix} x_1 & x_2 & \cdots & x_n \end{bmatrix}^\intercal$

$$x^\intercal Ax = x^\intercal UU^\intercal x = (x^\intercal U)(x^\intercal U)^\intercal = \sum_{j=1}^{m}(\sum_{i=1}^{n} x_i z_{ij})^2 \geq 0$$

So A is positive semi-definite.

(c) **Proof:**

Because $K_1(\boldsymbol{x}, \boldsymbol{y})$ is a valid kernel, we have $K_1(\boldsymbol{x}, \boldsymbol{y}) = \boldsymbol{p}_i^\intercal \boldsymbol{p}_j$, where $\boldsymbol{p}_i \in R^{r_1}$. Similarly, $K_2(\boldsymbol{x}, \boldsymbol{y}) = \boldsymbol{q}_i^\intercal \boldsymbol{q}_j$ where $\boldsymbol{q}_i \in R^{r_2}$.

Hence

$$\begin{aligned} K(\boldsymbol{x}, \boldsymbol{y}) &= K_1(\boldsymbol{x}, \boldsymbol{y}) K_2(\boldsymbol{x}, \boldsymbol{y}) \\ &= \sum_{m=1}^{r_1} \boldsymbol{p}_{im} \boldsymbol{p}_{jm} \sum_{n=1}^{r_2} \boldsymbol{p}_{in} \boldsymbol{p}_{jn}) \\ &= \sum_{m=1}^{r_1} \sum_{n=1}^{r_2} (\boldsymbol{p}_{im} \boldsymbol{p}_{in})(\boldsymbol{p}_{jm} \boldsymbol{p}_{jn}) \\ &= \sum_{l=1}^{r_1 r_2} \boldsymbol{a}_{il} \boldsymbol{a}_{jl} \end{aligned}$$

So $K(\boldsymbol{x}, \boldsymbol{y})$ is also a valid kernel.

(d) **Proof:**

Because

$$K_1(\boldsymbol{x}, \boldsymbol{y}) = \begin{bmatrix} x_1 & x_2 & \cdots & x_n & 1 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ \\ y_n \\ 1 \end{bmatrix} = 1 + < \boldsymbol{x}, \boldsymbol{y} >$$

So $K_1(\boldsymbol{x}, \boldsymbol{y})$ is a valid kernel. Because $K(\boldsymbol{x}, \boldsymbol{y}) = K_1(\boldsymbol{x}, \boldsymbol{y})^r$, according to (c), it is also a valid kernel.

(e) **Proof:**

$$\begin{aligned} K(\boldsymbol{x}, \boldsymbol{y}) &= \exp{(-||\boldsymbol{x} - \boldsymbol{y}||_2^2 / 2\sigma^2)} \\ &= \exp{(-\sum_{i=1}^{n} (\boldsymbol{x}_i - \boldsymbol{y}_i)^2 / 2\sigma^2)} \\ &= \prod_{i=1}^{n} \exp{(-(\boldsymbol{x}_i - \boldsymbol{y}_i)^2 / 2\sigma^2)} \end{aligned}$$

Because $K(\boldsymbol{x}_i, \boldsymbol{y}_i)$ is a valid kernel, according to (c), $K(\boldsymbol{x}, \boldsymbol{y})$ is also a valid kernel.

5. **RBF kernel and nearest neighbors.**

(a) Recall that given data examples $((\boldsymbol{x}_i, y_i))_{i=1}^n$ and an optimal dual solution $(\hat{\alpha}_i)_{i=1}^n$, the RBF kernel SVM makes a prediction as follows:

$$f_\sigma(\boldsymbol{x}) = \sum_{i=1}^n \hat{\alpha}_i y_i \exp\left(-\frac{\|\boldsymbol{x}-\boldsymbol{x}_i\|_2^2}{2\sigma^2}\right) = \sum_{i\in S} \hat{\alpha}_i y_i \exp\left(-\frac{\|\boldsymbol{x}-\boldsymbol{x}_i\|_2^2}{2\sigma^2}\right),$$

where $S \subset \{1, 2, \dots, n\}$ is the set of indices of support vectors.

Given an input $\boldsymbol{x}$, let $T := \arg\min_{i\in S} \|\boldsymbol{x}-\boldsymbol{x}_i\|_2$ denote the set of closest support vectors to $\boldsymbol{x}$, and let $\rho := \min_{i\in S} \|\boldsymbol{x}-\boldsymbol{x}_i\|_2$ denote this smallest distance. (In other words, $T := \{i \in S : \|\boldsymbol{x}-\boldsymbol{x}_i\| = \rho\}$.) Prove that

$$\lim_{\sigma\to 0} \frac{f_\sigma(\boldsymbol{x})}{\exp\left(-\rho^2/2\sigma^2\right)} = \sum_{i\in T} \hat{\alpha}_i y_i.$$

Remark: in other words, when the bandwidth σ becomes small enough, RBF kernel SVM is almost the 1-nearest neighbor predictor with the set of support vectors as the training set.

(b) Consider the XOR dataset:

$$\begin{aligned}
\boldsymbol{x}_1 &= (+1, +1), \quad y_1 = +1,\\
\boldsymbol{x}_2 &= (-1, +1), \quad y_2 = -1,\\
\boldsymbol{x}_3 &= (-1, -1), \quad y_3 = +1,\\
\boldsymbol{x}_4 &= (+1, -1), \quad y_4 = -1.
\end{aligned}$$

Verify that $\hat{\boldsymbol{\alpha}} = (1/\alpha, 1/\alpha, 1/\alpha, 1/\alpha)$ is an optimal dual solution to the RBF kernel SVM, where

$$\alpha = \left(1 - \exp\left(-\frac{\|\boldsymbol{x}_1-\boldsymbol{x}_2\|_2^2}{2\sigma^2}\right)\right)^2 = \left(1 - \exp\left(-\frac{2}{\sigma^2}\right)\right)^2 > 0.$$

Hint: prove that the gradient of the dual function is $\mathbf{0}$ at $\hat{\boldsymbol{\alpha}}$. Since the dual function is concave, and $\hat{\boldsymbol{\alpha}} > \mathbf{0}$, it follows that $\hat{\boldsymbol{\alpha}}$ is an optimal dual solution.

Remark: in other words, all four data examples are mapped to support vectors in the reproducing kernel Hilbert space. In light of (a), when σ is small enough, $f_\sigma(\boldsymbol{x})$ is almost the 1-nearest neighbor predictor on the XOR dataset. In fact, it is also true for large σ, due to the symmetry of the XOR data.

Solution. *(Your solution here.)*

(a)

$$\begin{aligned}
\lim_{\sigma\to 0} \frac{f_\sigma(\boldsymbol{x})}{\exp\left(-\rho^2/2\sigma^2\right)} &= \lim_{\sigma\to 0} \sum_{i\notin T} \hat{\alpha}_i y_i \exp\left(-\frac{\|\boldsymbol{x}-\boldsymbol{x}_i\|_2^2 - \rho^2}{2\sigma^2}\right) + \sum_{i\in T} \hat{\alpha}_i y_i\\
&= \sum_{i\in T} \hat{\alpha}_i y_i.
\end{aligned}$$

(b) Given $L(\alpha) = \max_{\boldsymbol{\alpha}\in\mathcal{C}} \sum_{i=1}^n \alpha_i - \frac{1}{2}\sum_{i,j=1}^n \alpha_i\alpha_j y_i y_j K(\boldsymbol{x}_i, \boldsymbol{x}_j)$

$$\begin{aligned}
\frac{\partial L}{\partial \alpha_i} &= 1 - \alpha_j \sum_{j \neq i} K(\boldsymbol{x}_i, \boldsymbol{x}_j) y_i y_j - \alpha_i K(\boldsymbol{x}_i, \boldsymbol{x}_i) y_i^2 \\
&= 1 - \alpha_i(-2\exp{(-\frac{2}{\sigma^2})} + \exp{(-\frac{4}{\sigma^2})}) - \alpha_i \\
&= 1 - \alpha_i(\exp{(-\frac{4}{\sigma^2})} - 2\exp{(-\frac{2}{\sigma^2})} + 1) \\
&= 1 - \frac{1}{\alpha}(1 - \exp{(-\frac{2}{\sigma^2})})^2 \\
&= 0
\end{aligned}$$

Q.E.D.

6. **SVM implementation.**

Recall that the dual problem of SVM is

$$\max_{\boldsymbol{\alpha}\in\mathcal{C}} \sum_{i=1}^{n} \alpha_i - \frac{1}{2}\sum_{i,j=1}^{n} \alpha_i\alpha_j y_i y_j K(\boldsymbol{x}_i, \boldsymbol{x}_j).$$

where the domain $\mathcal{C} = [0,\infty)^n = \{\boldsymbol{\alpha} : \alpha_i \geq 0\}$ for hard-margin SVM, and $\mathcal{C} = [0,C]^n = \{\boldsymbol{\alpha} : 0 \leq \alpha_i \leq C\}$ for soft-margin SVM.

Equivalently, it can be formulated as a minimization problem

$$\min_{\boldsymbol{\alpha}\in\mathcal{C}} f(\boldsymbol{\alpha}) := \frac{1}{2}\sum_{i,j=1}^{n} \alpha_i\alpha_j y_i y_j K(\boldsymbol{x}_i, \boldsymbol{x}_j) - \sum_{i=1}^{n} \alpha_i.$$

It can be solved by projected gradient descent, which starts from some $\boldsymbol{\alpha}_0 \in \mathcal{C}$ (e.g., $\mathbf{0}$) and updates as follows

$$\boldsymbol{\alpha}_{t+1} = \Pi_{\mathcal{C}}\left[\boldsymbol{\alpha}_t - \eta\nabla f(\boldsymbol{\alpha}_t)\right].$$

Here $\Pi_{\mathcal{C}}[\boldsymbol{\alpha}]$ is the *projection* of $\boldsymbol{\alpha}$ onto $\mathcal{C}$, defined as the closet point to $\boldsymbol{\alpha}$ in $\mathcal{C}$:

$$\Pi_{\mathcal{C}}[\boldsymbol{\alpha}] := \arg\min_{\boldsymbol{\alpha}'\in\mathcal{C}} \|\boldsymbol{\alpha}' - \boldsymbol{\alpha}\|_2.$$

If $\mathcal{C}$ is convex, the projection is uniquely defined.

(a) Prove that

$$\left(\Pi_{[0,\infty)^n}[\boldsymbol{\alpha}]\right)_i = \max\{\alpha_i, 0\},$$

and

$$\left(\Pi_{[0,C]^n}[\boldsymbol{\alpha}]\right)_i = \min\{\max\{0, \alpha_i\}, C\}.$$

(b) Implement an `svm_solver()`, using projected gradient descent formulated as above. See the docstrings in `hw3.py` for details.

(c) Implement an `svm_predictor()`, using an optimal dual solution, the training set, and an input. See the docstrings in `hw3.py` for details.

(d) On the area $[-5,5]\times[-5,5]$, plot the contour lines of the following kernel SVMs, trained on the XOR data. Different kernels and the XOR data are provided in `hw3_utils.py`. Learning rate 0.1 and 10000 steps should be enough. To draw the contour lines, you can use `hw3.svm_contour()`.

- The polynomial kernel with degree 2.
- The RBF kernel with $\sigma = 1$.
- The RBF kernel with $\sigma = 2$.
- The RBF kernel with $\sigma = 4$.

Solution. *(Your solution here.)*

(a)

(b)

(c)

(d)

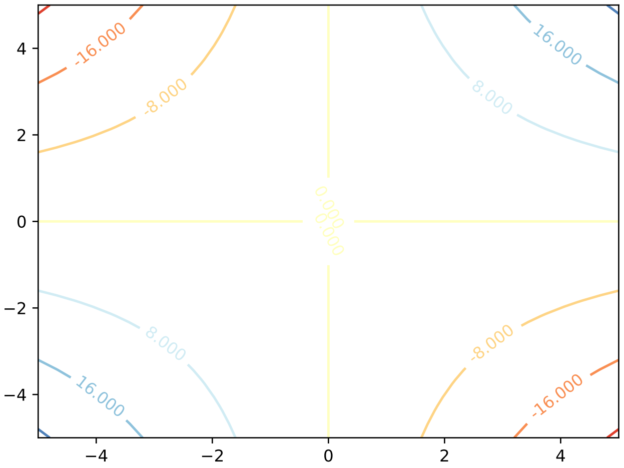

Figure 4: ploynomial kernel with degree 2

References

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

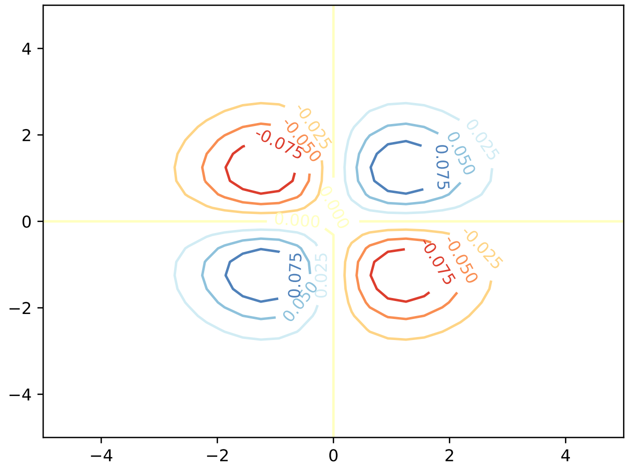

Figure 5: RBF kernel with sigma 1

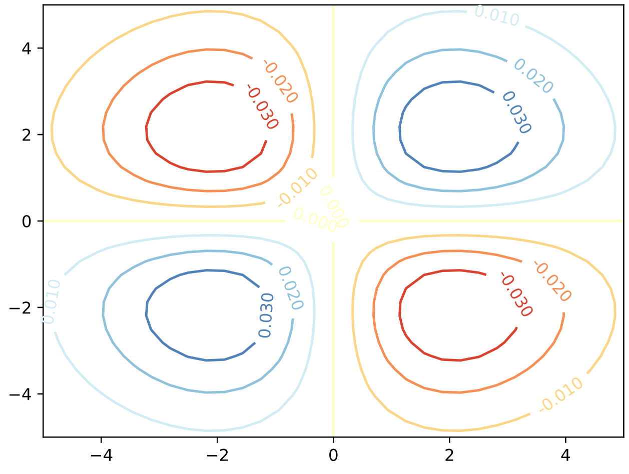

Figure 6: RBF kernel with sigma 2

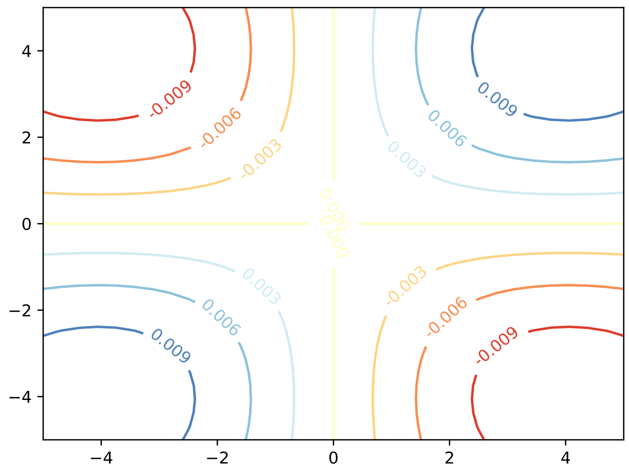

Figure 7: RBF kernel with sigma 4