CS480/680, Spring 2025

Assignment 2

Designer: Alireza Fathollah Pour; Instructor: Hongyang Zhang

Released: June 4; Due: July 2, noon

Instructions

- We do not accept hand-written submissions.
- This assignment is due by noon on July 2, 2025.
- For questions labelled as "**coding**", please follow the instructions provided and implement the required features. Unless otherwise specified, all implementations should be in *Python* using a *Jupyter Notebook*. Before submission, please make sure that your code can run without any errors. Also, be sure to save the output of each cell, as any missing output may not be graded.
- Please submit the following TWO files to LEARN:
  - A write-up in PDF format: the written answers to ALL questions, including the reported results and plots of coding questions, in a single PDF file.
  - An IPYNB file: your implementations for ALL coding questions. Please save the output of each cell, or your coding questions may NOT be graded.

Question 1 (Kernels (10 pts)). *In this question we investigate some closure properties of kernels and how they can be used to design interesting kernels.*

1. *(2 pts) Show that if k_1 is a kernel, then $k(x_1, x_2) = \alpha k_1(x_1, x_2)$ is also a kernel for any $\alpha \geq 0$.*
2. *(2 pts) Show that if k_1 and k_2 are kernels, then $k(x_1, x_2) = k_1(x_1, x_2) + k_2(x_1, x_2)$ is also a kernel.*
3. *(2 pts) Show that if k_1 and k_2 are kernels, then $k(x_1, x_2) = k_1(x_1, x_2)k_2(x_1, x_2)$ is also a kernel.*
4. *(2 pts) Assume k_1 is a kernel. Show that for any polynomial p with positive coefficients, $p(k_1(x_1, x_2))$ is a kernel.*
5. *(2 pts) Show that the Radial Basis Function (RBF) kernel*

$$k(x_1, x_2) = \exp\left(-\frac{\|x_1 - x_2\|^2}{2\sigma^2}\right)$$

is a valid kernel [Hint: Try using Taylor series of exponential function. You can use the fact that if $(k_i)_{i\in\mathbb{N}}$ are kernels then so is $\lim_{n\to\infty} k_n$ without proving it].

Question 2 (Regularization (10 pts)). *Overfitting to the training set is a major concern in machine learning. A simple remedy is to inject noise into each training datum before feeding it to the learner. In this exercise you will prove that such noise injection is equivalent to adding a particular form of regularization.*

1. *Recall that least–squares regression solves the following objective function*

$$\min_{\mathbf{w}\in\mathbb{R}^d} \sum_{i=1}^{n} \big(y_i - \langle \mathbf{w}, \mathbf{x}_i \rangle\big)^2, \tag{1}$$

where $\mathbf{x_i} \in \mathbb{R}^d$ and $y_i \in \mathbb{R}$ are training data. Instead of using $\mathbf{x}_i$ directly, we perturb it by independent noise $\boldsymbol{\varepsilon}_i$ to obtain $\tilde{\mathbf{x}}_i = f(\mathbf{x}_i, \boldsymbol{\varepsilon}_i)$ with different choices of the perturbation function f. We then optimize the following expected *least-squares objective*

$$\min_{\mathbf{w}\in\mathbb{R}^d} \sum_{i=1}^{n} \mathbb{E}\big[\big(y_i - \langle \mathbf{w}, \tilde{\mathbf{x}}_i \rangle\big)^2\big], \tag{2}$$

where the expectation is w.r.t to the noise and removes the randomness in $\tilde{\mathbf{x}}_i$ (we treat $\mathbf{x}_i, y_i$ as fixed).

   (a) *(3 pts) Let $\tilde{\mathbf{x}}_i = f(\mathbf{x}_i, \boldsymbol{\varepsilon}_i) = \mathbf{x}_i + \boldsymbol{\varepsilon}_i$ with $\boldsymbol{\varepsilon}_i \sim \mathcal{N}(\mathbf{0}, \lambda I)$ following the standard Gaussian distribution. Show that (2) is equal to the usual least-squares problem (1) plus an ℓ_2 regularizer on $\mathbf{w}$. State the exact regularization coefficient.*

   (b) *(4 pts) For a vector $\mathbf{x}_i \in \mathbb{R}^d$ we write x_i^j for its j-th entry. Let $\tilde{\mathbf{x}}_i = f(\mathbf{x}_i, \boldsymbol{\varepsilon}_i) = \mathbf{x}_i \odot \boldsymbol{\varepsilon}_i$, where $\odot$ denotes element-wise multiplication and $p\varepsilon_i^j$ are i.i.d. from $\sim$ BERNOULLI(p), i.e.,*

$$\varepsilon_i^j \;=\; \begin{cases} 0 & \text{with probability } 1-p, \\ 1/p & \text{with probability } p. \end{cases}$$

   Note that for different training data $\mathbf{x}_i, \boldsymbol{\varepsilon}_i$'s are independent. Simplify (2) and identify the induced (weighted) ℓ_2 regularizer on $\mathbf{w}$. Explain what happens when the features are normalized so that $\sum_i (x_i^j)^2 = 1$ for every j.

2. *(3 pts) Let (X, Y) be a random example with $Y \in \{\pm 1\}$. Define a label-flipped variable*

$$\tilde{Y} \;=\; \begin{cases} Y & \text{with probability } p, \\ -Y & \text{with probability } 1-p. \end{cases}$$

Show that for any classifier $f : \mathcal{X} \to \{\pm 1\}$,

$$\mathbb{E}\big[\,[f(X) \neq \tilde{Y}]\,\big] \;=\; (2p-1)\,\mathbb{E}\big[\,[f(X) \neq Y]\,\big] + 1 - p.$$

Question 3 (Coding (10 pts))**.** *We first implement a VGG-11 architecture and train it on MNIST dataset. The VGG-11 architecture is listed below, where each convolutional layer is denoted as* `Conv(number of input channels, number of output channels, kernel size, stride, padding)`*, the batch norms are denoted as* `BatchNorm(number of channels)`*, max-pools as* `MaxPool(kernel size, stride)`*, fully connected layers as* `FC(number of input features, number of output features)`*, and dropout as* `Dropout( droput ratio)`*.*

VGG-11 architecture.

- `Conv(001, 064, 3, 1, 1) - BatchNorm(064) - ReLU - MaxPool(2, 2)`
- `Conv(064, 128, 3, 1, 1) - BatchNorm(128) - ReLU - MaxPool(2, 2)`
- `Conv(128, 256, 3, 1, 1) - BatchNorm(256) - ReLU`
- `Conv(256, 256, 3, 1, 1) - BatchNorm(256) - ReLU -MaxPool(2, 2)`
- `Conv(256, 512, 3, 1, 1) - BatchNorm(512) - ReLU`
- `Conv(512, 512, 3, 1, 1) - BatchNorm(512) - ReLU - MaxPool(2, 2)`
- `Conv(512, 512, 3, 1, 1) - BatchNorm(512) - ReLU`
- `Conv(512, 512, 3, 1, 1) - BatchNorm(512) - ReLU - MaxPool(2, 2)`
- `FC(0512, 4096) - ReLU - Dropout(0.5)`
- `FC(4096, 4096) - ReLU - Dropout(0.5)`
- `FC(4096, 10)`

1. *Implement the above architecture and train it on MNIST dataset using* `torch.nn.CrossEntropyLoss`*. You can train it for about 10 epochs if compute resource is an issue.* ***Note.*** *VGG expects 32×32 inputs, and the original size of MNIST is 28×28. You can use* `transforms.Resize(32)` *to resize the images when loading MNIST dataset.*

   You need to create the following plots and insert them in the PDF file you submit.

   (a) *(1 pt) test accuracy vs the number of epochs*
   (b) *(1 pt) training accuracy vs the number of epochs*
   (c) *(1 pt) test loss vs the number of epochs*
   (d) *(1 pt) training loss vs the number of epochs*

2. *(1 pt) Try the following two types of flipping the images from left to right and from top to bottom in the test data set. Report the test accuracy after each type flip. What is the effect on the prediction?*

   `torchvision.transforms.RandomHorizontalFlip(p=1)`

```
torchvision.transforms.RandomVerticalFlip(p=1)
```

3. *(1 pt) Try adding Gaussian noise to test images with variance* $0.01, 0.1, 1$. *What is the effect on the prediction? You can use the following to add Gaussian noise.*

```
t = torchvision.transforms.Lambda(lambda x :  x + 0.1*torch.randn_like(x))
```

4. *(2pts) Finally, we investigate the effect of regularization on model performance. Retrain your model with data augmentation and test the above items again. Report the test accuracy and explain what kind of data augmentation you use in retraining.*