Labs
Machine Learning Course
Fall 2025

Westlake University
Department of Artificial Intelligence, SOE
Tao Lin
https://github.com/LINs-lab/course_machine_learning

Problem Set Lab 09, Nov 27, 2025 (Kernels & Neural Network Introduction)

Goals. The goals of this exercise are to:

- Gain a better understanding of properties of valid kernel functions.
- Familiarize you with the cross-entropy loss for multi-class classification.
- Introduce you to the PyTorch deep learning framework.
- Explore the representational capacity of neural networks by approximating 2d functions.

Theory Exercises

Problem 1 (Kernels):

In class we have seen that many kernel functions $k(\boldsymbol{x}, \boldsymbol{x}')$ can be written as inner products $\phi(\boldsymbol{x})^\top \phi(\boldsymbol{x}')$, for a suitably chosen vector-function $\phi(\cdot)$ (often called a feature map). Let us say that such a kernel function is *valid*. We further discussed many operations on valid kernel functions that result again in valid kernel functions. Here are two more.

1. Let $k_1(\boldsymbol{x}, \boldsymbol{x}')$ be a valid kernel function. Let f be a polynomial with positive coefficients. Show that $k(\boldsymbol{x}, \boldsymbol{x}') = f(k_1(\boldsymbol{x}, \boldsymbol{x}'))$ is a valid kernel.
2. Show that $k(\boldsymbol{x}, \boldsymbol{x}') = \exp(k_1(\boldsymbol{x}, \boldsymbol{x}'))$ is a valid kernel assuming that $k_1(\boldsymbol{x}, \boldsymbol{x}')$ is a valid kernel. *Hint*: You can use the following property: if $(K_n)_{n\geq 0}$ is a sequence of valid kernels and if there exists a function $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ such that for all $(x, x') \in \mathcal{X}^2$, $K_n(x, x') \underset{n\to+\infty}{\longrightarrow} K(x, x')$, then K is a valid kernel.

Problem 2 (Softmax Cross Entropy):

In this exercise, we study multi-class classification with the *softmax-cross-entropy* loss (or simply *cross-entropy*) which can be seen as a generalization of the logistic loss to more than 2 classes. First, we define the *softmax* of a vector $\mathbf{x} = [x_1, ..., x_d]^\top$ is a vector $\mathbf{z} = [z_1, ..., z_d]^\top$ with:

$$z_k = \frac{\exp(x_k)}{\sum_{i=1}^{d} \exp(x_i)}. \tag{1}$$

The label y is an integer denoting the target class. To turn y into a probability distribution for use with cross-entropy, we use one-hot encoding:

$$\text{onehot}(y) = \mathbf{y} = [y_1, ..., y_d]^\top \text{ where } y_k = \begin{cases} 1, & \text{if } k = y \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

The cross-entropy is given by:

$$H(\mathbf{y}, \mathbf{z}) = -\sum_{i=1}^{d} y_i \ln(z_i) \tag{3}$$

We ask you to do the following:

1. Equation **??** potentially computes $\exp$ of large positive numbers which is numerically unstable. Modify Eq. **??** to avoid positive numbers in $\exp$. Hint: Use $\max_j(x_j)$.

2. Derive $\frac{\partial H(\mathbf{y},\mathbf{z})}{\partial x_j}$. You may assume that $\mathbf{y}$ is a one-hot vector.

3. What values of x_i minimize the softmax-cross-entropy loss? To avoid complications, practitioners sometimes use a trick called label smoothing where $\mathbf{y}$ is replaced by $\hat{\mathbf{y}} = (1-\epsilon)\mathbf{y} + \frac{\epsilon}{d}\mathbf{1}$ for some small value e.g. $\epsilon = 0.1$, with $\mathbf{1} \in \mathbb{R}^d$ defined as the vector of ones, i.e., $\mathbf{1} = [1, \ldots, 1]^\top$.

Programming Exercises

Problem 3 (PyTorch Introduction and Neural Network Training):

The accompanying Jupyter Notebook (`labs/lab09/template/ex09.ipynb`) contains a brief introduction to PyTorch along with two neural network exercises. You will explore the representational capacity of neural networks by approximating 2d functions and train a digit classifier. Note that some details like the backpropagation algorithm will be explained in detail next week. For now, you can use the PyTorch autograd as a black box that returns you the gradients needed for optimization.

We recommend running the notebook on **Google Colab** which provides you with a free GPU and does not require installing any packages. Alternatively you can download the notebook from GitHub and install PyTorch locally, see the instructions on **pytorch.org**.

Additional Tutorials: If you plan on using PyTorch in your own projects, we recommend additionally going through the official tutorials after the exercise session:

- Deep Learning with PyTorch: a 60-minute Blitz
- Learning PyTorch with Examples