This written part of HW0 covers some of the fundamental concepts in probability theory and linear algebra, the prerequisites of the course. In addition to this written part, there are programming exercises for dynamic programming and PyTorch on Colab.

---

## Problem 1

(conditional probability, Bayes' rule)

There is a multiple choice exam with 5 choices for each question. Suppose there is a 0.5 probability of a student knowing the answer, and a 0.25 probability that they can eliminate a choice, otherwise all 5 choices seem equally plausible. When a student does not know the answer, they would guess randomly from the 4 or 5 choices. If a student answers a question correctly, what is the probability they knew the answer?

## Problem 2

(joint probability, marginal probability, independence, expectation)

Let $X \sim Ber(0.3)$ and $Y \sim Ber(0.7)$ be independent random variables. Define $S$ and $T$ by: $S = X + Y$ and $T = X - Y$.

(a) Find the joint and marginal PMFs for $S$ and $T$.

(b) Are S and T independent?

(c) Find $\mathbb{E}[S]$ and $\mathrm{Var}(2T)$.

## Problem 3

In this problem we will use the problem of linear least squares to review several concepts from linear algebra. We are given a data matrix $X \in \mathbb{R}^{n\times d}$ and the corresponding labels $y \in \mathbb{R}^n$, and the ordinary least squares problem is defined as:

$$\min_{w\in\mathbb{R}^d} \frac{1}{n}\sum_{i=1}^{n}(y_i - x_i^T w)^2 = \min_{w\in\mathbb{R}^d} \frac{1}{n}\|Xw - y\|^2.$$

The above formulation is associated with the linear system

$$Xw = y.$$

(a) (rank) Assume that the columns of $X$ are linearly independent. What is the rank of $X$?

(b) (solutions of linear systems) Following (a), suppose that $n > d$. In this case, how many solutions can we obtain for the linear system? Please explain.

(c) (singular value decomposition) Now assume that $n < d$ and the rows of $X$ are linearly independent. In this case, the linear system will have infinitely many solutions. A classic way is to select the solution with minimal norm, which would be $w = X^\dagger y = X^T(XX^T)^{-1}y$ (we won't prove this here). $X^\dagger$ is also called the Moore-Penrose inverse of $X$, which in practice can be calculated by the singular value decomposition (SVD). Recall that the SVD finds a decomposition $X = U\Sigma V^T$ with orthogonal matrices $U \in \mathbb{R}^{n\times n}, V \in \mathbb{R}^{d\times d}$ and rectangular diagonal matrix $\Sigma \in \mathbb{R}^{n\times d}$. Show with SVD that the solution can be written in the form of

$$w = X^\dagger y = V\Sigma^\dagger U^T y,$$

where $\Sigma^\dagger \in \mathbb{R}^{d\times n}$ has reciprocals of $\Sigma$ in its diagonal entries.

(d) (matrix calculus) Another way to solve linear least squares is with (batch) gradient descent (GD), which is an iterative optimization algorithm for finding the local minimum of a function. Its stochastic approximation, the (mini-batch) stochastic gradient descent (SGD), is commonly used in training modern neural networks. For linear least squares, the objective is convex and differentiable, so we can use the update rule

$$w_t = w_{t-1} - \gamma \nabla_w \mathcal{L}(w),$$

where $\mathcal{L}(w) = \frac{1}{n}\|Xw - y\|^2$, with a $\gamma$ that is small enough to achieve the optimal solution. Derive $\nabla_w \mathcal{L}$ and write down the gradient descent iteration.