Problem Set 1

Physics 198: (Deep) Learning Mechanics

Covers Chapter 1: Introduction to Neural Networks

Chapter 1 Exercises

The exercises in Chapter 1 serve to confront the reader with the perils of relying on naive analytical methods to understand deep learning.

1.1 Coupled Dynamics

In this problem, we will see why the combination of coupled dynamics and discrete updates results in an analytically intractable nightmare.

Consider a 2-layer linear neural network $\hat{f}(x) = W_2 W_1 x$. For simplicity, let's look at a single training example (x, y). Under the MSE loss $\mathcal{L}(x) = \frac{1}{2}||y - W_2 W_1 x||^2$, the gradients with respect to each weight matrix are:

$$\nabla_{W_2} \mathcal{L} = -(y - W_2 W_1 x)(W_1 x)^\top$$
$$\nabla_{W_1} \mathcal{L} = -W_2^\top (y - W_2 W_1 x) x^\top$$

a) Write down the gradient descent update equations for $W_1^{(1)}$ and $W_2^{(1)}$ in terms of the initializations $W_1^{(0)}$, $W_2^{(0)}$ and learning rate η.

b) Now, substitute $W_1^{(1)}$ and $W_2^{(1)}$ into the gradient expression to write out the update for the second step, $W_2^{(2)}$. Do not fully expand the polynomial, but simply count: if you were to fully distribute the multiplication, how many distinct terms would comprise $W_2^{(2)}$

c) Based on part b) briefly explain why analytically predicting the exact state of $W_2^{(t)}$ at an arbitrary time step t using discrete updates is a hopeless endeavor. How does taking the continuous-time limit (gradient flow) elegantly bypass this specific algebraic nightmare?

1.2 Pointwise Nonlinearities

In this problem, we will see how the introduction of pointwise nonlinearities makes even simple networks hard to analyze.

Let's reintroduce a nonlinearity but strip the network down to a single weight vector to eliminate coupled dynamics. Consider the model $\hat{f}(x) = \text{ReLU}(w^\top x) = \max(0, w^\top x)$. We train this weight vector on a dataset of P examples $\{(x_i, y_i)\}_{i=1}^P$ using MSE loss: $\mathcal{L} = \frac{1}{2}\sum_{i=1}^P (y_i - \text{ReLU}(w^\top x_i))$.

a) Write down the gradient flow differential equation $\dot{w}$. Now, define the active set of data points at time t as $\mathcal{A}(w(t)) = \{i \mid w(t)^\top x_i > 0\}$. Rewrite your differential equation strictly in terms of the sum over the active set $\mathcal{A}(w(t))$.

b) Look closely at your equation from part a). If the active set $\mathcal{A}(w(t))$ magically remained constant for all $t > 0$, does this seem like a solvable problem? Why is this realization simultaneously a relief and a trap?

In reality, the active set is not constant. The weight space $\mathbb{R}^d$ is sliced by P distinct hyperplanes defined by $w^\top x_i = 0$. Every time the trajectory $w(t)$ crosses one of these hyperplanes, a data point enters or exits the active set. Given that P can be absurdly large, tracking the angle between w and every single data point x_i quickly gets out of hand.

1.3 Data Complexity

In this problem, we will explore how the structure of the data can affect a model's learning dynamics.

Let's return to the exact gradient flow solution for over-parameterized linear regression in terms of the residual $r = y - Xw$: $r(t) = e^{-2XX^\top t} r(0)$. To understand how fast the network learns, we need to analyze the exponent $-2XX^\top t$. Because the matrix $XX^\top \in \mathbb{R}^{P \times P}$ is symmetric and positive-semidefinite, we can decompose it via eigendecomposition as $XX^\top = Q\Lambda Q^\top$, where Q is an orthogonal matrix of eigenvectors and Λ is a diagonal matrix of eigenvalues $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_P > 0$.

a) Using the property that $e^{Q\Lambda Q^\top} = Qe^{\Lambda}Q^\top = Q\text{diag}(e^{\lambda_1}, \ldots, e^{\lambda_P})Q^\top$, rewrite the gradient flow solution for the residual $r(t) = y - Xw(t)$ strictly in terms of Q, Λ, t, and the initial residual $r(0)$.

b) Suppose our data is perfectly isotropic white noise, such that $XX^\top \approx I_P$. What can you say about the rate at which the residual decays for different components of the target vector?

c) Now suppose our data consists of natural images. Natural data is highly correlated, meaning Λ will have a few massive eigenvalues and a long tail of very small eigenvalues. Based on your equation from part (a), explain the concept of spectral learning order: which components of the data does the network memorize immediately, and which take an exponentially long time to learn? Why does this make predicting the general training time of a neural network impossible without strict assumptions about the data distribution?