COS 514: Fundamentals of Deep Learning

Fall 2025

Instructor: Prof. Sanjeev Arora
TA: Gon Buzaglo

Assignment 2

Instructions:

- Submission deadline is October 6.
- We recommend start reading all questions as soon as possible, as some are harder than others.
- You may collaborate in groups of up to **3** students.
- If you collaborate on a problem, you must clearly state the names of your collaborators at the beginning of the solution to that problem.
- All group members must declare that they contributed equally to the solutions.
- You must write up your own solutions independently in LaTeX. **Handwritten or scanned solutions will not be accepted.**
- Cite any resources (papers, textbooks, websites) that you use.
- Submit your assignment as a single PDF on gradescope.

Problems

Problem 1: PAC-Bayes with Gaussian Distributions Implies Generalization of Min-Norm Solution

Consider linear classifiers in $\mathbb{R}^k$ parameterized by $w \in \mathbb{R}^k$. Let the prior be $P = \mathcal{N}(0, I_k)$. After training on a sample S of size n you obtain $\hat{w}$, and take the posterior to be $Q = \mathcal{N}(\hat{w}, I_k)$. Assume the loss $\ell(h_w, (x, y)) \in [0, 1]$ for all (x, y) and let $\hat{L}_S(Q)$ and $L_{\mathcal{D}}(Q)$ denote, respectively, the empirical and population risks of the Gibbs classifier that draws $w \sim Q$ and predicts with h_w.

1. (KL divergence calculation, step by step.) We want to compute the KL divergence between

$$Q = \mathcal{N}(\hat{w}, I_k) \quad \text{and} \quad P = \mathcal{N}(0, I_k).$$

Recall the pdf of a k-dimensional Gaussian:

$$p(w) = \frac{1}{(2\pi)^{k/2}(\det \Sigma)^{1/2}} \exp\Big(-\tfrac{1}{2}(w-\mu)^\top \Sigma^{-1}(w-\mu)\Big).$$

Show that

$$D(Q\|P) = \tfrac{1}{2}\,\|\hat{w}\|_2^2.$$

2. (PAC-Bayes bound with explicit target inequality.) Recall the standard PAC-Bayes kl-bound, which states that with probability at least $1-\delta$ over the draw of S,

$$L_{\mathcal{D}}(Q) \;\le\; \hat{L}_S(Q) \;+\; \sqrt{\frac{D(Q\|P) + \ln\!\big(\frac{2\sqrt{n}}{\delta}\big)}{2\,(n-1)}}.$$

Conclude that in our Gaussian setup,

$$L_{\mathcal{D}}(Q) \;\le\; \hat{L}_S(Q) \;+\; \sqrt{\frac{\frac{1}{2}\,\|\hat{w}\|_2^2 + \ln\!\big(\frac{2\sqrt{n}}{\delta}\big)}{2\,(n-1)}}.$$

3. Briefly explain what this bound suggests about why low-ℓ_2-norm solutions (e.g., via L_2-regularization or max-margin bias) are conducive to good generalization.

Problem 2: Typical Generalization in Neural Networks

We study how a simple "Guess–and–Check" algorithm can lead to generalization when training neural networks.

Setup.

- A *student network* has weights $w \in \mathbb{R}^k$.
- Inputs are vectors $x \in \mathbb{R}^{d_0}$.
- A training dataset is

$$S = \{x^{(n)}\}_{n=1}^{N}, \quad x^{(n)} \sim P_X \text{ i.i.d.}$$

- The labels are generated by a *teacher network* $f_{w^\star}(x)$.
- The student and teacher have the same depth, but the teacher is *narrower* (fewer hidden neurons).

The student has a two-layer form:

$$f_w(x) = \operatorname{sign}\big(w_2^\top [W_1 x]_+\big),$$

where $[\,\cdot\,]_+$ is the ReLU activation. The teacher is defined in the same form, but only the first $d_1^\star$ hidden neurons are active, so its effective width is $d_1^\star < d_1$.

The *Guess–and–Check algorithm* samples candidate weights from q possible values independently, tests them on S, and stops once a perfect fit is found.

Questions.

1. What is the probability of guessing the exact teacher network?
2. What is the probability of matching the teacher's function (not necessarily its exact weights)?
3. After T trials, what is the probability that no perfect fit is found?
4. Derive a generalization bound for the hypothesis chosen by Guess–and–Check, and compare it to the naive bound obtained by considering the entire class of student networks.

Problem 3: Implicit Bias of Logistic Regression on Separable Data

We study implicit bias in logistic regression on linearly separable data. Although the loss has no finite minimizer, gradient descent converges in direction to the *maximum-margin separator* (the solution of the Hard SVM problem).

Setup. We are given data $\{(x_n, y_n)\}_{n=1}^N$ with $x_n \in \mathbb{R}^d$ and $y_n \in \{-1, 1\}$. A linear classifier predicts $\text{sign}(w^\top x)$. The data are linearly separable if there exists $w^\star$ such that $y_n (w^\star)^\top x_n > 0$ for all n. The Hard SVM problem is

$$\min_w \|w\| \quad \text{s.t. } y_n w^\top x_n \geq 1 \ \ \forall n.$$

The logistic regression loss is

$$\mathcal{L}(w) = \sum_{n=1}^N \log\bigl(1 + \exp(-y_n w^\top x_n)\bigr).$$

Questions.

(a) Define what it means for an algorithm to have an *implicit bias*, and why this matters for generalization.

(b) Show that gradient flow on $\mathcal{L}(w)$ drives $\|w(t)\| \to \infty$.

(c) Argue that only support vectors influence the asymptotic direction of $w(t)$.

(d) Normalize by the minimum margin $\gamma(t) = \min_n w(t)^\top x_n$ and analyze the limit points of $v(t) = w(t)/\gamma(t)$.

(e) Conclude that the direction of $w(t)$ converges to the maximum-margin separator, i.e. the solution of the Hard SVM.

Takeaway. Gradient descent on logistic regression, despite the absence of a finite minimizer, converges in direction to the maximum-margin solution.

Problem 4: Influence Functions

At first sight, computing influence functions appears difficult due to the inverse Hessian computation, which naively has cubic complexity in the number of parameters. In this question, we will see how to use the matrix power series for fast but approximate computation of the form $H^{-1}v$. The key idea is a simple identify in the following question.

1. If A is any positive definite matrix with full rank and maximum eigenvalue less than 1, then show that

$$A^{-1} = \sum_{i=0}^{\infty} (I - A)^i.$$

   Hint: Note that A is diagonalizable. How do eigenvectors and eigenvalues of A^i relate to those of A?

2. If S_r denotes the truncation of the series to its first r terms, then show that $\lambda_{max}(A^{-1} - S_r) \leq \lambda_{min}(A)^{-1}(1 - \lambda_{min}(A))^{r+1}$. Compute a value of r for which the r-term approximation to the Hessian-vector product is within $(1 + \epsilon)$ multiplicative factor of the correct value.

Problem 5: Shapley Values

The *Shapley value* is a way to distribute credit among N players in a cooperative game with utility function $U(\cdot)$. Formally, the Shapley value of player i is

$$s_i = \mathbb{E}_{\pi}[U(\pi_{\leq i}) - U(\pi_{<i})],$$

where π is a random permutation of $\{1, \ldots, N\}$ and $\pi_{<i}$ are the players before i in π.

The following are natural axioms for any credit-attribution scheme:

1. **Efficiency:** $\sum_i s_i = U([N])$.
2. **Symmetry:** If $U(S \cup \{i\}) = U(S \cup \{j\})$ for all S not containing i, j, then $s_i = s_j$.
3. **Linearity:** For utilities U_1, U_2, the values for $U_1 + U_2$ equal the sum of the values for U_1 and U_2.

4. **Null Player:** If $U(S \cup \{i\}) = U(S)$ for all S not containing i, then $s_i = 0$.

By linearity of expectation in the definition of Shapley value, all four axioms are satisfied.

1. In the first part of the exercise we will use a toy model to make the non-linearity issue of neural networks concrete and easy to analyze. The goal is to see how the Shapley axioms produce a mathematically "fair" but potentially misleading explanation for a model with strong feature interactions.

   Consider a simple 2x2 binary image with four pixels: Top-Left (P_{TL}), Top-Right (P_{TR}), Bottom-Left (P_{BL}), and Bottom-Right (P_{BR}).

   A simple "corner detection" model is defined as follows. It outputs a high score (logit) if and only if three specific pixels are "on" (value 1): P_{TL}, P_{TR}, and P_{BL}.

   The utility function $U(S)$ for a set of pixels S is:

   $$U(S) = \begin{cases} 1 & \text{if } \{P_{TL}, P_{TR}, P_{BL}\} \subseteq S, \\ 0 & \text{otherwise.} \end{cases}$$

   The image we want to explain is $\{P_{TL} = 1, P_{TR} = 1, P_{BL} = 1, P_{BR} = 0\}$. For this image, the total utility is $U(\{P_{TL}, P_{TR}, P_{BL}\}) = 1$. We assume the baseline (empty set) utility $U(\emptyset) = 0$.

   (a) Calculate the Shapley values for all four pixels: s_{TL}, s_{TR}, s_{BL}, and s_{BR}.

   (b) Analyze the result:

   - How does the model distribute the credit for the prediction?
   - Does the Null Player axiom hold for the P_{BR} pixel?
   - Do the Symmetry and Efficiency axioms hold? How do they help you find the solution?
   - Discuss whether the resulting Shapley values provide a "true" explanation of how this AND-gate-like model works. What insight do the values provide, and what do they obscure about the model's logic?

2. Computing Shapley values exactly is NP-hard, since it requires evaluating $U(S)$ for all 2^N subsets. A simple approximation method is to sample $O(R^2 N \log N/\epsilon^2)$ random permutations and estimate the expectation in the definition

$$s_i = \mathbb{E}_\pi[U(\pi_{\leq i}) - U(\pi_{<i})]\,.$$

With high probability this approximates the vector of Shapley values within ℓ_2 error at most ϵ.

A more efficient approach is as follows:

(a) Approximate s_1 within error $\epsilon/(2\sqrt{N})$ using the naive method above.

(b) Use the following fact to estimate the differences $s_1 - s_j$ for all j, within error $\epsilon/2\sqrt{N}$:

$$s_i - s_j \;=\; \frac{1}{N-1} \sum_{S \subseteq [N] \setminus \{i,j\}} \frac{U(S \cup \{i\}) - U(S \cup \{j\})}{\binom{N-2}{|S|}}.$$

Design and analyze step the above approach in detail. The key insight is that a single randomly chosen subset S can be used to estimate the marginal contribution of a player i versus all other players $j \notin S \cup i$ simultaneously, leading to significant computational savings