TTIC 31020: Introduction to Machine Learning **Winter 2025**

Homework 9

Due: March 6. No late submissions after noon on March 10.

Note *You may discuss these problems in groups. However, you must write up your own solutions and mention the names of the people in your group. Also, please do mention any books, papers or other sources you refer to. We prefer and encourage that you typeset solutions in LaTeX for written assignments. Handwritten solutions that are not neat, organized and with clear and easy-to-read handwriting will not be graded. Please submit your solutions as a PDF document on Canvas.*

Challenge and Optional Questions *Challenge questions are marked with* challenge *and optional questions are marked with* optional *. You can get extra credit for solving* challenge *questions. You are not required to turn in* optional *questions, but we encourage you to solve them for your understanding. Course staff will help with* optional *questions but will prioritize queries on non-*optional *questions.*

1. **Back Propagation.** In this question, our goal is to consider neural networks which have either sigmoid or softmax activation units and calculate the gradients with respect to model parameters.

   Consider a neural network architecture, which is represented by a directed acyclic graph $G(V, E)$. Each node has an output value associated with it, which are defined recursively as follows.

   The nodes with no incoming edges are the input nodes $v_1, \dots v_d$. Their values are given by the input $x \in \mathbb{R}^d$. In particular, we have $o[v_i] = x[i]$ for $i \in [d]$. For any other node $v \in V$, having computed output values of its parent nodes $\{o[u] : (u, v) \in E\}$, the output value of v is computed using either one of the two parametric functions below.

   - **Sigmoid activation.** We have parameters $w(u, v) \in \mathbb{R}$ for each edge (u, v)[1]. Then

   $$o[v] = \text{sigmoid}\left(\sum_{u:(u,v)\in E} w(u, v)o[u]\right), \text{ and } \text{sigmoid}(z) = \frac{1}{1 + e^{-z}},$$

     where sigmoid : $\mathbb{R} \to \mathbb{R}$ is a continuous differentiable function.

   - **Softmax activation.** Let $k := |\{u : (u, v) \in E\}|$, i.e. k is the number of parents of v. We have k^2 parameters $\{w(i, u, v) : i \in [k], (u, v) \in E\}$[2]. The output of v is then computed by

   $$o[v] = \text{softmax}\left(\sum_{u:(u,v)\in E} w(1, u, v)o[u], \ \dots, \ \sum_{u:(u,v)\in E} w(k, u, v)o[u]\right).$$

     Here softmax : $\mathbb{R}^k \to \mathbb{R}$ is defined as follows.

   $$\text{softmax}(z_1, \dots, z_k) := \frac{\sum_{i=1}^k z_i e^{z_i}}{\sum_{i=1}^k e^{z_i}}.$$

   Finally, there is an output node v_{out} with no outgoing edges whose output value is the final output $\hat{y} = o[v_{\text{out}}]$.

[1] You may think of it as a vector of parameters; one associated with each parent node u.
[2] You may think of it as a matrix of parameters in $\mathbb{R}^{k\times k}$.

(a) Calculate $\frac{\partial \hat{y}}{\partial w}$ for any parameter w in the system. In other words: for any node v, whose output is calculated using sigmoid activation, calculate $\frac{\partial \hat{y}}{\partial w(u,v)}$. Similarly, for any v whose value is calculated using softmax activation, calculate $\frac{\partial \hat{y}}{\partial w(i,u,v)}$. You may define stimuli of different nodes recursively and provide your answer in terms of them.

(b) Now consider the special case of the layered graph with depth two. In particular,

$$\hat{y}(x) = \text{softmax}\left(W^{(2)}\text{sigmoid}\left(W^{(1)}x\right)\right), \text{ where } W^{(1)} \in \mathbb{R}^{k\times d}, W^{(2)} \in \mathbb{R}^{k\times k}.$$

Here sigmoid is applied to a vector entrywise. Moreover, consider the squared loss $\ell^{\text{sq}}(\hat{y}, y) = \frac{1}{2}(\hat{y} - y)^2$. For an example $(x, y) \in (\mathbb{R}^d \times \mathbb{R})$, compute

$$\nabla_{W^{(2)}} \ell^{\text{sq}}(\hat{y}(x), y) \text{ and } \nabla_{W^{(1)}} \ell^{\text{sq}}(\hat{y}(x), y).$$

Simplify your answer so that the final expression is only in terms matrix/vector multiplications, without using summation notation (over indices).

2. **Multiclass and Structured Prediction.** Consider a multi-class prediction problem over a (possibly very large) label set $\mathcal{Y}$, with cost sensitive error metric $\Delta : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}_+$, where for every pair of labels $\hat{y}, y$, $\Delta(\hat{y}, y)$ is the cost of predicting $\hat{y}$ when the correct label is y, and we always have $\forall_{y\in\mathcal{Y}}, \Delta(y, y) = 0$. You should think of $\mathcal{Y}$ as potentially extremely large, e.g. the set of all sentences or images. The task loss for a predictor $h : \mathcal{X} \to \mathcal{Y}$ is then given by:

$$\ell^{\Delta}(h, (x, y)) = \Delta(h(x), y)$$

We consider an approach to multi-class prediction based on a joint feature map $\Psi : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}^d$. A multi-class predictor $h_w : \mathcal{X} \to \mathcal{Y}$ based on weights $w \in \mathbb{R}$ is the defined as

$$h_w(x) = \arg\max_{y\in\mathcal{Y}} \langle w, \Psi(x, y)\rangle .$$

The Δ-multiclass hinge loss of such a predictor is given by:

$$\ell^{\text{hinge}}(w, (x, y)) = \max_{y'\in\mathcal{Y}} \left(\Delta(y', y) + \langle w, \Psi(x, y')\rangle - \langle w, \Psi(x, y)\rangle\right).$$

For a training set S, We will refer to the empirical population task loss $L_S^{\Delta}(h)$ and empirical multi-class hinge loss $L_S^{\text{hinge}}(w)$.

(a) First, let's verify that the multi-class hinge loss generalizes the binary hinge loss: Consider $\mathcal{Y} = \{\pm 1\}$, $\Delta(-1, +1) = \Delta(+1, -1) = 1$ and $\Psi(x, y) = \frac{1}{2}y\phi(x)$, for some feature map $\phi : \mathcal{X} \to \mathbb{R}^d$. Show that $h_w(x) = \text{sign}(\langle \tilde{w}, \phi(x)\rangle)$ corresponds to standard linear classification, and $L_S^{\text{hinge}}(w)$ is exactly the standard binary hinge loss.

(b) Next, let's study the multi-class hinge loss as a convex surrogate:

i. Prove that the multiclass hinge loss $L_S^{\text{hinge}}(w)$ is convex in w.

ii. Prove that multiclass hinge loss upper bounds the task loss, i.e. $L_S^{\text{hinge}}(w) \geq L_S^{\Delta}(h_w)$.

iii. Prove that if there is a weight vector w with zero task loss, $L_S^{\Delta}(h_w) = 0$, then there is some $\tilde{w}$ with zero multi-class hinge loss, $L_S^{\text{hinge}}(\tilde{w}) = 0$.

(c) Calculate a subgradient of the multiclass hinge loss with respect to w. What is the computational complexity in terms of d and $|\mathcal{Y}|$ of computing a subgradient?

(d) challenge Consider a structured problem where the input and output are strings, so $\mathcal{X} = \mathcal{Y} = \Sigma^T$ for fixed T and finite alphabet Σ. Let $\Delta(\hat{y}, y) = |\{i|\hat{y}[i] \neq y[i]\}|$ be the Hamming distance. Define the $|\Sigma|^4$-dimensional feature map

$$\Psi(x, y)[a, b, c, d] = |\{i \in [T-1] \mid x[i], x[i+1], y[i], y[i+1] = a, b, c, d\}|$$

which represents joint bigrams of x and y. What is the computational complexity of the naïve approach (as in item 2c above) for calculating subgradients? Now suggest a more efficient dynamic programming procedure, both for computing $h_w(x)$ itself, as well as subgradients of the multi-class hinge loss, both with runtime linear in T.

(e) challenge Alternatively, for the same input and output spaces $\mathcal{X} = \mathcal{Y} = \Sigma^T$ and hamming distance Δ, and a per-symbol feature map $\phi : \Sigma \times \Sigma \to \mathbb{R}^d$, consider the "product" feature map $\Psi(x, y) \in \mathbb{R}^{T \times d}$ where $\Psi(x, y)[i, :] = \phi(x[i], y[i])$. Describe how to compute $h_w(x)$ and subgradients of the multi-class hinge loss in time linear in T, and show that this can also be expressed as T independent learning problems.

3. challenge **Expressive Power of Neural Networks.** Let $\mathcal{X} = \{0, 1\}^d$ and the hypothesis class of all parity functions in $\{0, 1\}^{\mathcal{X}}$. Formally,

$$\text{PARITIES}_d = \left\{ h_I(x) = \bigoplus_{i \in I} x_i : I \subseteq [d] \right\}.$$

We additionally define a hypothesis class of two layer ReLU networks with at most $2d$ hidden units.

$$\mathcal{H}_{\text{ReLU}} := \left\{ h_\theta(x) = \text{sign}\left(\sum_{i=1}^{2d} a_i \text{ReLU}(\langle w_i, x \rangle + b_i) \right) : \theta = (a_i \in \mathbb{R}, b_i \in \mathbb{R}, w_i \in \mathbb{R}^d)_{i \in [d]} \right\}.$$

(a) Show that $\text{PARITIES}_d \subseteq \mathcal{H}_{\text{ReLU}}$. In other words, for each parity function $h_I(x)$, there exists a setting of the weights of the ReLU network that realizes it.

(b) Recall from the lecture that VCdim of fully connected feed forward ReLU networks of a certain depth is $O(|E| \cdot \text{depth} \cdot \log |E|)$, where $|E|$ is the total number of edges in the architecture graph, or alternatively, the number of parameters. Relying on this fact, propose a neural network based learning rule (not necessarily with efficient runtime) that can learn PARITIES_d but using only $\text{poly}(d)$ samples. What is the best sample complexity guarantee of your learning rule? Contrast this with $\text{VCdim}(\text{PARITIES}_d)$, which is the optimal sample complexity using any learning rule.