Foundations of Deep Learning **2020-2021 Spring Semester**

Assignment 3: Optimization

Due: May 31, 2021

Important Guidelines

1. Work can be done in pairs.
2. Solution should be submitted as a single .zip file containing: (i) a PDF that includes answers to theoretical questions and reports for experimental tasks; and (ii) all source files required to reproduce experimental results. It is highly recommended to typeset the solutions in LaTeX or Word. If you choose to submit scanned handwritten solutions, please make sure they are clearly written and the scan is of high quality.

Part 1: Optimization in deep learning is non-convex

1. **(10 points)** Prove the following proposition from class.

   Proposition: Consider a feed-forward fully connected neural network with depth $N \geq 2$ and activation $\sigma(\cdot)$:

   $$\mathcal{H} = \Big\{ x \mapsto y = W_N \sigma(W_{N-1}\sigma(W_{N-2}\sigma(\ldots W_2\sigma(W_1 x))\ldots)\ ;\ W_n \in \mathbb{R}^{d_n \times d_{n-1}}, n \in [N] \Big\}$$

   Let $L(W_1, W_2, \ldots, W_N)$ be a continuously differentiable loss function that depends on $W_1, W_2, \ldots, W_N$ only through the input-output mapping of the network. Assume that the global minimum of $L(\cdot)$ is **not** attained at $W_1 = 0, W_2 = 0, \ldots, W_n = 0$. Assume also that $\sigma(\cdot)$ is continuously differentiable and $\sigma(0) = 0$. Then, $L(\cdot)$ is non-convex.

Part 2: Landscape approach

1. **(10 points)** In this question we extend the convergence to stationary point result delivered in class from gradient descent to stochastic gradient descent.

   Let $f : \mathbb{R}^d \to \mathbb{R}$ be a twice continuously differentiable and β-smooth function that attains its global minimum: $f^* = \min_{\underline{w} \in \mathbb{R}^d} f(\underline{w})$. Suppose we run stochastic gradient descent over $f(\cdot)$:

   $$\underline{w}_{t+1} \leftarrow \underline{w}_t - \eta_t(\nabla f(\underline{w}_t) + \underset{\sim}{\xi}_t) \quad , \ t = 1, 2, 3, \ldots$$

   where η_t is a (possibly time-varying) learning rate, and $\underset{\sim}{\xi}_t$ stands for time-independent noise with zero mean ($\mathbb{E}[\underset{\sim}{\xi}_t] = \underline{0}$) and σ^2 variance ($\mathbb{E}[\|\underset{\sim}{\xi}_t\|^2] = \sigma^2$). Let $\epsilon > 0$.

   Assume $\eta_t = \frac{1}{\beta}$, and derive an upper bound on the number of iterations needed for reaching an expected $(\epsilon + \sigma)$-stationary point, i.e. on $T \in \mathbb{N}$ that will ensure:

   $$\min_{t \in [T]} \mathbb{E}\left[\|\nabla f(\underline{w}_t)\|\right] \leq \epsilon + \sigma$$

2. **(Bonus 8 points)** Show that by decaying learning rate it is possible to guarantee convergence to expected ϵ-stationary point. In particular, define a scheme for η_t and a respective upper bound on $T \in \mathbb{N}$ that will ensure:

$$\min_{t\in[T]} \mathbb{E}\left[\|\nabla f(\underline{w}_t)\|\right] \leq \epsilon$$

3. **(10 points)** Let $\ell : \mathbb{R}^{d,d} \to \mathbb{R}$ be a twice continuously differentiable convex loss, overparameterized by a depth N linear neural network with hidden widths d:

$$\phi : \mathbb{R}^{d,d} \times \mathbb{R}^{d,d} \times \cdots \times \mathbb{R}^{d,d} \to \mathbb{R} \quad , \quad \phi(W_1, W_2, \ldots, W_N) = \ell(W_N W_{N-1} \ldots W_1)$$

   Prove the following claims:

   - If we restrict the domain of $\phi(\cdot)$ to $B^{d,d} \times B^{d,d} \times \cdots \times B^{d,d}$, where $B^{d,d} := \left\{\underline{x} \in \mathbb{R}^{d,d} \;:\; \|\underline{x}\| \leq 1\right\}$, then we obtain a smooth (Lipschitz gradient) function.
   - If we do not restrict the domain of $\phi(\cdot)$, the function is smooth if and only if at least one of the following conditions holds: (i) $\ell(\cdot)$ is constant; or (ii) $\ell(\cdot)$ is affine and the depth is $N = 2$.

4. **Experiment (10 points):** On a regression dataset of your choice, train a depth N linear neural network (with hidden widths no smaller than the minimum between input dimension and output dimension) by minimizing ℓ_2 loss via (full batch) gradient descent with small learning rate and initialization. Plot the objective value, the magnitude of its gradient, and the maximal and minimal eigenvalues of its Hessian throughout the run. Explain your results. Repeat the experiment with depths $N = 2, 3, 4$.

Part 3: Trajectory approach

Linear neural networks

1. **(10 points)** Complete the proof sketch of the end-to-end dynamics given in class, by showing that under the conditions of the respective theorem:

$$W_{1:j}(t)^T W_{1:j}(t) = \left[W_{1:N}(t)^T W_{1:N}(t)\right]^{\frac{j}{N}} \qquad \forall t \in \mathbb{R}_{\geq 0} \;,\; j \in [N]$$

2. **(10 points)** Extend the end-to-end dynamics delivered in class to the case of a depth 2 **symmetric** linear neural network. Namely, for a continuously differentiable loss $\ell : \mathbb{R}^{d,d} \to \mathbb{R}$ parameterized by:

$$\phi : \mathbb{R}^{d,d} \to \mathbb{R} \;, \quad \phi(U) = \ell(UU^T)$$

   develop an expression for the dynamics of $W(t) := U(t)U(t)^T$ that are induced by gradient flow over $\phi(\cdot)$:

$$\dot{U}(t) := \frac{d}{dt}U(t) = -\nabla\phi(U(t))$$

3. **(10 points)** Simplify the end-to-end dynamics delivered in class for the special case of output dimension 1, i.e., $d_N = 1$. Explain how this resonates with our interpretation of the end-to-end dynamics "promoting movement in direction already taken".

4. **Experiment (10 points):** On a scalar regression dataset of your choice, train a depth N linear neural network (with hidden width no smaller than the minimum between input dimension and output dimension) by minimizing ℓ_2 loss via (full batch) gradient descent with small learning rate and initialization close to zero. Compare the trajectory taken by the end-to-end matrix to that obtained by directly applying the (discrete version of the) end-to-end dynamics to a linear model:

$$W_{t+1} \leftarrow W_t - \eta \sum_{j=1}^{N} \left[W_t W_t^T\right]^{\frac{j-1}{N}} \nabla \ell(W_t) \left[W_t^T W_t\right]^{\frac{N-j}{N}} \qquad t = 1, 2, 3, \dots ,$$

where η is the learning rate used for gradient descent over the linear neural network. Repeat the experiment with depths $N = 2$ and 3.

Ultra wide neural networks

1. **(10 points)** In class we have shown that if:

$$\dot{\underline{u}}(t) = -H^*(\underline{u}(t) - \underline{y}) \quad , t \in \mathbb{R}_{\geq 0} \, , \tag{1}$$

where

- $\underline{u}(t) \in \mathbb{R}^m$ holds neural network predictions on training instances at time t of optimization
- $H^* \in \mathbb{R}^{m,m}$ is the Neural Tangent Kernel's Gram matrix, positive definite with eigenvalues $\geq \lambda > 0$
- $\underline{y} \in \mathbb{R}^m$ holds training labels

then $\underline{u}(t) \to \underline{y}$ exponentially fast. This was done by applying a change of variables to $\underline{u}(t)$ based on the eigendecomposition of H^*.

Establish the same result without applying change of variables (hint: derive an expression for $\frac{d}{dt}\|\underline{u}(t) - \underline{y}\|^2$).

2. **(Bonus 8 points)** The dynamics in (1) are actually an idealization of:

$$\dot{\underline{u}}(t) = -H(t)(\underline{u}(t) - \underline{y}) \quad , t \in \mathbb{R}_{\geq 0} \, , \tag{2}$$

where $H(t) \in \mathbb{R}^{m,m}$ satisfies $\|H(t) - H^*\|_{spectral} \leq \epsilon$. Assuming identical initialization $\underline{u}(0)$, show that under (2), the value of $\underline{u}(t)$ is at most $O(\sqrt{t\epsilon})$ away (in Euclidean distance) from what it would be under (1).

3. **Experiment (10 points):** In class we analyzed a shallow network and derived the following formula for the Neural Tangent Kernel:

$$k(\underline{x}, \underline{x}') = \underline{x}^T \underline{x}' \cdot \mathbb{E}_{\underline{W} \sim \mathcal{N}(0,I)} \left[\dot{\sigma}(\underline{W}^T \underline{x}) \cdot \dot{\sigma}(\underline{W}^T \underline{x}')\right] .$$

For the special case of ReLU activation this amounts to:

$$k(\underline{x}, \underline{x}') = \underline{x}^T \underline{x}' \cdot \frac{1}{2\pi} \cdot \left[\pi - arccos\left(\frac{\underline{x}^T \underline{x}'}{\|\underline{x}\| \cdot \|\underline{x}'\|}\right)\right] . \tag{3}$$

For a scalar regression dataset of your choice, and with activation being ReLU, compare $\underline{u}(t)$ during gradient flow over the shallow network analyzed in class, to a direct implementation of the dynamics (1) (use the analytic expression for the Neural Tangent Kernel given in (3)). For emulating continuous optimization use small learning rate in your discrete implementations. Repeat the experiment with varying widths for the shallow network. Is there a closer match with larger width?