ECE421: Introduction to Machine Learning — Fall 2024

Worksheet 2: Gradient, Logistic Regression, and Non-linear Transformation

Q1. (Gradient Computation) For a scalar-valued function $f : \mathbb{R}^d \to \mathbb{R}$, the gradient evaluated at $\underline{w} \in \mathbb{R}^d$ is

$$\nabla f(\underline{w}) = \begin{bmatrix} \frac{\partial f(\underline{w})}{\partial w_1} & \cdots & \frac{\partial f(\underline{w})}{\partial w_d} \end{bmatrix}^\top \in \mathbb{R}^d.$$

Using this definition, compute the gradients of the following functions, where $A \in \mathbb{R}^{d \times d}$ is not necessarily a symmetric matrix.

1.a. $f(\underline{w}) = \underline{w}^\top A \underline{v} + \underline{w}^\top A^\top \underline{v} + \underline{v}^\top A \underline{w} + \underline{v}^\top A^\top \underline{w}$, where $\underline{v} \in \mathbb{R}^d$.

1.b. $f(\underline{w}) = \sum_{i=1}^d \log(1 + \exp(w_i))$

1.c. $f(\underline{w}) = \sqrt{1 + \|\underline{w}\|_2^2}$

Q2. (Logistic Regression) You are given a dataset $\mathcal{D} = \{(\underline{x}_n, y_n)\}_{n=1}^N$, where $\underline{x}_n \in \mathbb{R}^{d+1}, d \geq 1$, and $y_n \in \{+1, -1\}$. For $\underline{w} \in \mathbb{R}^{d+1}$ and $\underline{x} \in \mathbb{R}^{d+1}$, we wish to train a logistic regression model $h(\underline{x}) = \theta(\underline{w}^\top \underline{x})$, where $\theta(.)$ is the logistic function defined as $\theta(z) = \frac{e^z}{1+e^z}$, for $z \in \mathbb{R}$. Following the arguments on page 91 of LFD, the in-sample error can be written as $E_{\text{in}}(\underline{w}) = \frac{1}{N} \sum_{n=1}^N \log \left[\frac{1}{P_{\underline{w}}(y_n | \underline{x}_n)} \right]$ where

$$P_{\underline{w}}(y \mid \underline{x}) = \begin{cases} h(\underline{x}), & \text{if } y = +1, \\ 1 - h(\underline{x}), & \text{if } y = -1. \end{cases}$$

2.a. Show that $E_{\text{in}}(\underline{w})$ can be expressed as

$$E_{\text{in}}(\underline{w}) = \frac{1}{N} \left(\sum_{n=1}^N \mathbb{I}(y_n = +1) \log \left[\frac{1}{h(\underline{x}_n)} \right] + \mathbb{I}(y_n = -1) \log \left[\frac{1}{1 - h(\underline{x}_n)} \right] \right),$$

where $\mathbb{I}$(argument) evaluates to 1 if the argument is true and 0 if it is false.

2.b. Show that $E_{\text{in}}(\underline{w})$ can also be expressed as

$$E_{\text{in}}(\underline{w}) = \frac{1}{N} \sum_{n=1}^N \log(1 + \exp(-y_n \underline{w}^\top \underline{x}_n)).$$

2.c. Use (b) to show that

$$\nabla E_{\text{in}}(\underline{w}) = \frac{1}{N} \sum_{n=1}^N -y_n \underline{x}_n \theta(-y_n \underline{w}^\top \underline{x}_n),$$

and argue that a "misclassified" example contributes more to the gradient than a correctly classified one.

Q3. (Problem 4, Midterm 2017) Consider the logistic regression setup as in the previous question. Suppose we are given a dataset $D = \{(\underline{x}_1, y_1), (\underline{x}_2, y_2)\}$ with

$$\underline{x}_1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \quad y_1 = 1, \quad \underline{x}_2 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \quad y_2 = -1.$$

For any $\underline{w} \in \mathbb{R}^2$, we consider the ℓ_2-regularized error as

$$E_{\text{in}}(\underline{w}) = -\sum_{n=1}^N \log \left[P_{\underline{w}}(y_n \mid \underline{x}_n) \right] + \lambda \|\underline{w}\|_2^2, \quad \lambda > 0,$$

where

$$P_{\underline{w}}(y \mid \underline{x}) = \begin{cases} h(\underline{x}) & y = +1 \\ 1 - h(\underline{x}) & y = -1 \end{cases},$$

and

$$h(x) = \frac{e^{\underline{w}^\top \underline{x}}}{1 + e^{\underline{w}^\top \underline{x}}} = \frac{1}{1 + e^{-\underline{w}^\top \underline{x}}}.$$

3.a. For $\lambda = 0$, find the optimal $\underline{w}$ that minimizes $E_{\text{in}}(\underline{w})$ and the minimum value of $E_{\text{in}}(\underline{w})$. (Hint: you are given $(\underline{x}_n, y_n)$, so plug those values into the expression of the in-sample error).

3.b. Suppose λ is a very large constant such that it suffices to consider weights that satisfy $\|\underline{w}\|_2 \ll 1$. Since $\underline{w}$ has a small magnitude, we may use the Taylor series approximation

$$\log(1 + \exp(-y_n \underline{w}^\top \underline{x}_n)) \approx \log(2) - \frac{1}{2} y_n \underline{w}^\top \underline{x}_n.$$

Assuming the above approximation is exact, find $\underline{w}$ that minimizes $E_{\text{in}}(\underline{w})$ (it should be expressed in terms of λ).

Q4. (Hinge Loss) Here are two reviews of "Perfect Blue", from *Rotten Tomatoes*:

> **Panos Kotzathanasis** (Asian Movie Plus): "Perfect Blue" is an artistic and technical masterpiece; however, what is of outmost importance is the fact that Satoshi Kon never deteriorate from the high standards he set here, in the first project that was entirely his own.
>
> **Derek Smith** (Cinematic Reflections): [An] nime thriller [that] often plays as an examination of identity and celebrity, but ultimately gets so lost in its own complex structure that it doesn't end up saying much at all.

Rotten Tomatoes has classified these reviews as "positive" and "negative," respectively.

In this assignment, you will create a simple text classification system that can perform this task automatically. We'll warm up with the following set of four mini-reviews, each labeled positive $(+1)$ or negative (-1):

1. $\underline{x}_1$: not good; label: (-1)
2. $\underline{x}_2$: pretty bad; label: (-1)
3. $\underline{x}_3$: good plot; label: $(+1)$
4. $\underline{x}_4$: pretty scenery; label: $(+1)$

Each review $\underline{x}$ is mapped onto a feature vector $\phi(\underline{x})$, which maps each word to the number of occurrences of that word in the review and adds a 1 to account for bias. For example, the second review maps to the (sparse) feature vector $\phi(\underline{x}_2) = \{\textbf{extra added 1} : 1, \textbf{pretty} : 1, \textbf{bad} : 1\}$. The hinge loss for a single datapoint is defined as

$$L_{\text{hinge}}(\underline{x}, y, \underline{w}) = \max\{0, 1 - y\underline{w}^\top \phi(\underline{x})\},$$

where $\underline{x}$ is the review text, y is the correct label, $\underline{w}$ is the weight vector.

4.a. (Linearly Inseparable) Given the following dataset of reviews:

1. $\underline{x}_1$: bad; label: (-1)
2. $\underline{x}_2$: good; label: $(+1)$
3. $\underline{x}_3$: not bad; label: $(+1)$
4. $\underline{x}_4$: not good; label: (-1)

Prove that no linear classifier using word features (*i.e.*, word count) can get zero error on this dataset. Remember that this is a question about classifiers, not optimization algorithms; your proof should be true for any linear classifier of the form $f_{\underline{w}}(\underline{x}) = \text{sign}(\underline{w}^\top \phi(\underline{x}))$, regardless of how the weights are learned.

Propose a single additional feature for your dataset that we could augment the feature vector with that to fix this problem.

Q5. **(Squared Loss)** Suppose that we are now interested in predicting a numeric rating for movie reviews. We will use a non-linear predictor that takes a movie review $\underline{x}$ and returns $\sigma(\underline{w}^\top \phi(\underline{x}))$, where $\sigma(z) = (1 + e^{-z})^{-1}$ is the logistic function that squashes a real number to the range $(0, 1)$. For this problem, assume that the movie rating y is a real-valued variable in the range $[0, 1]$.

5.a. Suppose that we wish to use squared loss. Write out the expression of the loss $L(\underline{x}, y, \underline{w})$ for a single datapoint $(\underline{x}, y)$.

5.b. Given $L(\underline{x}, y, \underline{w})$ from the previous part, compute the gradient of the loss with respect to $\underline{w}$, $\nabla_{\underline{w}} L(\underline{x}, y, \underline{w})$. Write the answer in terms of the predicted value $p = \sigma(\underline{w}^\top \phi(\underline{x}))$.

5.c. Suppose there is one datapoint $(\underline{x}, y)$ with some arbitrary non-zero $\phi(\underline{x})$ and $y = 1$. Specify conditions for $\underline{w}$ to make the magnitude of the gradient of the loss with respect to $\underline{w}$ arbitrarily small (*i.e.*, minimize the magnitude of the gradient). Can the magnitude of the gradient with respect to $\underline{w}$ ever be exactly zero? You are allowed to make the magnitude of $\underline{w}$ arbitrarily large but not infinity.

Why does it matter? the reason why we're interested in the magnitude of the gradients is because it governs how far gradient descent will step. For example, if the gradient is close to zero when $\underline{w}$ is very far from the optimum, then it could take a long time for gradient descent to reach the optimum (if at all). This is known as the vanishing gradient problem when training neural networks.