EECS 16B Designing Information Devices and Systems II
Fall 2021 Discussion Worksheet Discussion 13A

1. Linearization to help classification: Discussion Version

This discussion is designed to help you get started on a closely related homework problem.

Consider trying to classify a set of measurements $\vec{x}_i$ with given labels ℓ_i. We consider the binary case of two possible labels: "+" and "-" and fold our threshold implicitly into the weights by augmenting the constant "1" in the first position of each $\vec{x}_i$. **We want to learn a vector of weights $\vec{w}$ so that we can deem any point with $\vec{x}_i^\top \vec{w} > 0$ as being a member of the "+" category and anything with $\vec{x}_i^\top \vec{w} < 0$ as being a member of the "-" category.**

We will do this using a minimization in the spirit of least squares. Except, instead of necessarily using some sort of squared loss function, we will just consider a generic cost function that can depend on the label and the prediction score for the point. For the i-th data point in our training data, we will incur a cost $c(\vec{x}_i^\top \vec{w}, \ell_i)$ for a total cost that we want to minimize by picking the best $\vec{w}$:

$$\underset{\vec{w}}{\text{argmin}} \quad c_{\text{total}}(\vec{w}) = \sum_{i=1}^{m} c(\vec{x}_i^\top \vec{w}, \ell_i) \tag{1}$$

Because this can be a nonlinear nonquadratic function, our goal is to solve this iteratively as a sequence of least-squares problems that we know how to solve.

Consider the following algorithm:

1: $\vec{w} = \vec{0}$ ▷ Initialize the weights to $\vec{0}$
2: **while** Not done **do** ▷ Iterate towards solution
3: Compute $\vec{w}^\top \vec{x}_i$ ▷ Generate current estimated labels
4: Compute $\frac{\mathrm{d}}{\mathrm{d}\vec{w}} c(\vec{w}^\top \vec{x}_i, \ell_i)$ ▷ Generate derivatives with respect to $\vec{w}$ of the cost for update step
5: Compute $\frac{\mathrm{d}^2}{\mathrm{d}\vec{w}^2} c(\vec{w}^\top \vec{x}_i, \ell_i)$ ▷ Generate second derivatives of the cost for update step
6: $\delta\vec{w} = \texttt{LeastSquares}(\cdot, \cdot)$ ▷ We will derive what to call least squares on
7: $\vec{w} = \vec{w} + \delta\vec{w}$ ▷ Update parameters
8: **end while**
9: Return $\vec{w}$

The key step above is figuring out with what arguments to call `LeastSquares` while only having the labels ℓ_i and the points $\vec{x}_i$.

Recall that when the function $\vec{f}(\vec{x}, \vec{y}) : \mathbb{R}^n \times \mathbb{R}^k \to \mathbb{R}^m$ takes in vectors and outputs a vector, the relevant derivatives for linearization are also represented by matrices:

$$\frac{\partial \vec{f}}{\partial \vec{x}} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial y[1]} & \cdots & \frac{\partial f_m}{\partial y[k]} \end{bmatrix}.$$

where we will use python style indexing to avoid confusion with iteration counts and so

$$\vec{x} = \begin{bmatrix} x[1] \\ \vdots \\ x_n \end{bmatrix} \qquad \vec{y} = \begin{bmatrix} y_1 \\ \vdots \\ y_k \end{bmatrix}. \tag{2}$$

Then, the linearization (first-order expansion) becomes

$$\vec{f}(\vec{x}, \vec{y}) \approx \vec{f}(\vec{x}_0, \vec{y}_0) + \frac{\partial \vec{f}}{\partial \vec{x}}(\vec{x}_0, \vec{y}_0) \cdot (\vec{x} - \vec{x}_0) + \frac{\partial \vec{f}}{\partial \vec{y}}(\vec{x}_0, \vec{y}_0) \cdot (\vec{y} - \vec{y}_0)\,. \tag{3}$$

(a) Now, suppose we wanted to approximate the cost for each data point

$$c_i(\vec{w}) = c(\vec{x}_i^\top \vec{w}, \ell_i) \tag{4}$$

where

$$\vec{w} = \begin{bmatrix} w_1 \\ \vdots \\ w_n \end{bmatrix} \tag{5}$$

in the neighborhood of a weight vector $\vec{w}_\star$. Our goal is to write out the first-order expression for approximating the cost function $c_i(\vec{w}_\star + \delta\vec{w})$. This should be something in vector/matrix form like you have seen for the approximation of nonlinear systems by linear systems. We don't want to take any second derivatives just yet — only first derivatives. We have outlined a skeleton for the derivation with some parts missing. Follow the guidelines in each sub-section.

(i) Comparing to eq. (3), we know that $c_i(\vec{w}_\star + \delta\vec{w}) \approx c_i(\vec{w}_\star) + \frac{\partial c_i}{\partial \vec{w}}(\vec{w}_\star) \cdot \delta\vec{w}$. **Write out the vector form of** $\frac{\partial c_i}{\partial \vec{w}}(\vec{w}_\star)$.

(ii) **Write out the partial derivative of** $c_i(\vec{w})$ **with respect to** w_g**, the** g^{th} **component of** $\vec{w}$**, i.e., find** $\frac{\partial c_i(\vec{w})}{\partial w_g}$**.**

You should leave the answer in terms of the expression $c'(p, \ell) = \frac{\mathrm{d}}{\mathrm{d}p} c(p, \ell)$. This "prime" notation can be useful since by itself c is a scalar valued function of two scalar arguments and so this c' is also a scalar valued function of two scalar arguments.

(HINT: Use the linearity of derivatives and sums to compute the partial derivatives with respect to each of the w_g terms. Don't forget the chain rule and the fact that $\vec{x}_i^\top \vec{w} = \sum_{j=1}^n x_{ij} w_j = x_{ig} w_g + \sum_{j \neq g} x_{ij} w_j$.)

(iii) With what you had above, **can you fill in the missing part to express the row vector $\frac{\partial}{\partial \vec{w}} c_i(\vec{w})$?** What should $c'(\cdot, \cdot)$ be?

$$\frac{\partial}{\partial \vec{w}} c_i(\vec{w}) = c'(\vec{x}_i^\top \vec{w}, \ell_i)\_\_\_\_\_ \tag{6}$$

(b) Now, we want a better approximation that includes second derivatives. For a general function, we would look for

$$f(\vec{x}_0 + \delta\vec{x}) \approx f(\vec{x}_0) + \frac{\partial f}{\partial \vec{x}}(\vec{x}_0)\delta\vec{x} + \frac{1}{2}\delta\vec{x}^\top \left(\frac{\partial^2 f}{\partial \vec{x}^2}(\vec{x}_0)\right)\delta\vec{x} \tag{7}$$

where $\frac{\partial f}{\partial \vec{x}}(\vec{x}_0)$ is an appropriate row vector and, as you've seen in the note, $\frac{\partial^2 f}{\partial \vec{x}^2}(\vec{x}_0)$ is called the Hessian and represents the second derivatives.

(i) Comparing to eq. (7), we know that

$$c_i(\vec{w}_\star + \delta\vec{w}) \approx c_i(\vec{w}_\star) + \frac{\partial c_i}{\partial \vec{w}}(\vec{w_\star}) \cdot \delta\vec{w} + \frac{1}{2}\delta\vec{w}^\top \left(\frac{\partial^2 c_i}{\partial \vec{w}^2}(\vec{w_\star})\right)\delta\vec{w} \tag{8}$$

Write out the matrix form of $\frac{\partial^2 c_i}{\partial \vec{w}^2}(\vec{w_\star})$ in terms of the second partial derivatives $\frac{\partial^2 c_i}{\partial w_g \partial w_h}(\vec{w}_\star)$.

(ii) **Take the second derivatives of the cost $c_i(\vec{w})$, i.e. solve for $\frac{\partial^2 c_i(\vec{w})}{\partial w_g \partial w_h}$.**

You should leave the answer in terms of $c''(p, \ell) = \frac{\mathrm{d}^2}{\mathrm{d}p^2} c(p, \ell)$. This c'' is also a familiar scalar second derivative with respect to the scalar first argument of the cost function.

(HINT: You should use the answer to part (a) and just take another derivative. Once again, use the linearity of derivatives and sums to compute the partial derivatives with respect to each of the w_h terms. This will give you $\frac{\partial^2}{\partial w_g \partial w_h}$. Don't forget the chain rule and again use the fact that $\vec{x}_i^\top \vec{w} = \sum_{j=1}^n x_{ij} w_j = x_{ih} w_h + \sum_{j \neq h} x_{ij} w_j$.)

(iii) The expression in part (ii) is for the $(g,h)^{\text{th}}$ component of the second derivative. $\frac{1}{2}$ times this times δw_g times δw_h would give us that component's contribution to the second-derivative term in the approximation, and we have to sum this up over all g and h to get the total contribution of the second-derivative term in the approximation. Now, we want to group terms to restructure this into matrix-vector form by utilizing the outer-product form of matrix multiplication. **What should the space in the following expression be filled with?**

$$\frac{\partial^2}{\partial \vec{w}^2} c_i(\vec{w}) = c''(\vec{x}_i^\top \vec{w}, \ell_i)\_\_\_\_\_ \tag{9}$$

What should $c''(\cdot,\cdot)$ be?

(c) Now we have successfully expressed the second order approximation of $c_i(\vec{w}_\star + \delta\vec{w})$. Since we eventually want to minimize the total cost $c_{\text{total}}(\vec{w}) = \sum_{i=1}^m c_i(\vec{w})$, **can you write out the second order approximation of $c_{\text{total}}(\vec{w}_\star + \delta\vec{w})$ using results from (a) and (b)?**

(d) In this part we explore solving this problem using "Newton's method," which will feel superficially different from the iterative least squares formulation above which is pursued in the homework. Recall that for a differentiable scalar-valued function $f(\vec{w})$, we can set $\frac{\partial f}{\partial \vec{w}}(\vec{w_\star}) = \vec{0}^\top$ to find a candidate extremum (in analogy with what you know about finding maxima and minima in single variable calculus — here there are many different variables and we want the function to be locally maximum/minimum with respect to each of them). This is a system of nonlinear equations of the type solved by Newton's method via local linearization. Consider a linearization of the derivative of our cost function, $\frac{\partial f}{\partial \vec{w}}^\top$ at the point $\vec{w}$. **Use a linearization for $\frac{\partial f}{\partial \vec{w}}(\vec{w})^\top$ and the fact that $\frac{\partial f}{\partial \vec{w}}(\vec{w_\star}) = \vec{0}^\top$ to derive an update using Newton's method.**

(This is like the Inverse Kinematics problem you did on the last homework.)

2. Using Automatic Differentiation using pytorch

In this part we introduce basic ideas of auto-differentiation with PyTorch.

(a) The choice of the loss function effects the decision boundary that we recover. Let's consider different examples of cost functions

- squared error: $c^+_{\text{sq}}(p) = (p-1)^2,\ c^-_{\text{sq}}(p) = (p+1)^2$;
- exponential: $c^+_{\text{exp}}(p) = \mathrm{e}^{-p},\ c^-_{\text{exp}}(p) = \mathrm{e}^{p}$;
- and logistic: $c^+_{\text{logistic}}(p) = \ln\left(1+\mathrm{e}^{-p}\right),\ c^-_{\text{logistic}}(p) = \ln(1+\mathrm{e}^{p})$.

Plot the different cost functions.

(b) In the previous question, we derived a second order linearization of the total cost function

$$c_i(\vec{w}_\star + \delta\vec{w}) \approx c_i(\vec{w}_\star) + \frac{\partial c_i}{\partial \vec{w}}(\vec{w}_\star)\delta\vec{w} + \frac{1}{2}\delta\vec{w}^\top\left(\frac{\partial^2 c_i}{\partial \vec{w}^2}(\vec{w}_\star)\right)\delta\vec{w} \tag{10}$$

The above expression requires us to compute the derivative and hessian of $c(\vec{w})$. Consider the following optimzation problem

$$\min_{\vec{w}} \|\vec{w}\|_2^2$$

where $w \in \mathbb{R}^2$. This is akin to finding a $\vec{w}$ which minimizes the cost function $c(\vec{w}) = \|\vec{w}\|_2^2$. To solve this problem, we consider two approaches

- with derivative : $\vec{w}_{t+1} = \vec{w}_t - \frac{\partial c}{\partial \vec{w}}(\vec{w}_\star)^\top$
- with hessian : $\vec{w}_{t+1} = \vec{w}_t - \left\{\frac{\partial^2 c}{\partial \vec{w}^2}(\vec{w}_\star)\right\}^{-1}\frac{\partial c_i}{\partial \vec{w}}(\vec{w}_\star)$

Let's visualize the loss surface, the weights during multiple iterations of update.

(c) Let's revisit the classification problem. Consider the following dataset, where points have two labels, *red* and *blue*. We want to recover a decision boundary $\vec{w}$, such that for $\vec{w}^\top\vec{x} \geq 0$ for red points, $\vec{w}^\top\vec{x} < 0$ for blue points. **Visualize the datasets and true decision boundary.**

(d) **Compare the training loss for different cost functions across multiple iterations.**

(e) In this problem we learn $\vec{w} \in \mathbb{R}^3$. **Visualize the value of the cost function at different points in the space of weights, and track the weights across multiple iterations.**

(f) **Compare the following decision boundaries as you change the number of iterations**

- logistic cost with gradient update
- logistic cost with Hessian update
- exponential cost with Hessian update
- true classifier boundary

Contributors:

- Kourosh Hakhamaneshi.
- Kuan-Yun Lee.
- Nathan Lambert.
- Sidney Buchbinder.
- Gaoyue Zhou.
- Anant Sahai.
- Kumar Krishna Agrawal.

