Linear Classifiers (Part 3)

CS114B Lab 4

Kenneth Lai

February 10, 2023

Gradients in Logistic Regression

- $L(\hat{y}, y) = -[y \log \hat{y} + (1 - y) \log(1 - \hat{y})]$

Gradients in Logistic Regression

- $L(\hat{y}, y) = -[y \log \hat{y} + (1-y)\log(1-\hat{y})]$
- ...
- (calculus–see supplement slides)
- ...
- $\dfrac{\partial L}{\partial w_j} = (\hat{y} - y)x_j$
- $\dfrac{\partial L}{\partial b} = \hat{y} - y$
- $\nabla L = (\hat{y} - y)\mathbf{x}$

Gradients in Perceptrons

- $L(\hat{y}, y) = (\hat{y} - y)z$

Gradients in Perceptrons

- $L(\hat{y}, y) = (\hat{y} - y)z$
- ...
- (calculus–see supplement slides)
- ...
- $\dfrac{\partial L}{\partial w_j} = (\hat{y} - y)x_j$
- $\dfrac{\partial L}{\partial b} = \hat{y} - y$
- $\nabla L = (\hat{y} - y)\mathbf{x}$

Gradients in Perceptrons

- $L(\hat{y}, y) = (\hat{y} - y)z$
- ...
- (calculus–see supplement slides)
- ...
- $\dfrac{\partial L}{\partial w_j} = (\hat{y} - y)x_j$
- $\dfrac{\partial L}{\partial b} = \hat{y} - y$
- $\nabla L = (\hat{y} - y)\mathbf{x}$
- Does this look familiar?

Perceptron Learning Algorithm

- If $\hat{y} = y$, then $\hat{y} - y = 0$
  - Do nothing

Perceptron Learning Algorithm

- If $\hat{y} = y$, then $\hat{y} - y = 0$
  - Do nothing
- If $\hat{y} = 0$ and $y = 1$, then $\hat{y} - y = -1$
  - $\nabla L = -\mathbf{x}$
  - $\theta_{t+1} = \theta_t + \eta \mathbf{x}$
  - Increment weights

Perceptron Learning Algorithm

- If $\hat{y} = y$, then $\hat{y} - y = 0$
  - Do nothing
- If $\hat{y} = 0$ and $y = 1$, then $\hat{y} - y = -1$
  - $\nabla L = -\mathbf{x}$
  - $\theta_{t+1} = \theta_t + \eta \mathbf{x}$
  - Increment weights
- If $\hat{y} = 1$ and $y = 0$, then $\hat{y} - y = 1$
  - $\nabla L = \mathbf{x}$
  - $\theta_{t+1} = \theta_t - \eta \mathbf{x}$
  - Decrement weights

Gradients in Multinomial Logistic Regression

- Cross-entropy loss $L(\hat{\mathbf{y}}, \mathbf{y}) = -\sum_{k=1}^{p} y_k \log \hat{y}_k$

Gradients in Multinomial Logistic Regression

- Cross-entropy loss $L(\hat{\mathbf{y}}, \mathbf{y}) = -\sum_{k=1}^{p} y_k \log \hat{y}_k$
- Gradient ∇L becomes a matrix, where
  - $\dfrac{\partial L}{\partial W_{jk}} = (\hat{y}_k - y_k) x_j$
  - $\dfrac{\partial L}{\partial b_k} = \hat{y}_k - y_k$

Gradients in Multinomial Logistic Regression

- Cross-entropy loss $L(\hat{\mathbf{y}}, \mathbf{y}) = -\sum_{k=1}^{p} y_k \log \hat{y}_k$
- Gradient ∇L becomes a matrix, where
  - $\dfrac{\partial L}{\partial W_{jk}} = (\hat{y}_k - y_k) x_j$
  - $\dfrac{\partial L}{\partial b_k} = \hat{y}_k - y_k$
- $\nabla L = \mathbf{x} \otimes (\hat{\mathbf{y}} - \mathbf{y})$, where
  - $\otimes$ denotes the outer product

Multi-class Perceptron Learning Algorithm

- Multi-class perceptron loss $L(\hat{y}, y) = z_{\hat{y}} - z_y$

Multi-class Perceptron Learning Algorithm

- Multi-class perceptron loss $L(\hat{y}, y) = z_{\hat{y}} - z_y$
- If $\hat{y} = y$, then do nothing
- Else:

Multi-class Perceptron Learning Algorithm

- Multi-class perceptron loss $L(\hat{y}, y) = z_{\hat{y}} - z_y$
- If $\hat{y} = y$, then do nothing
- Else:
  - For the correct class y, $\dfrac{\partial L}{\partial z_y} = -1$
    - $(\nabla L)_y = -\mathbf{x}$
    - $(\theta_y)_{t+1} = (\theta_y)_t + \eta \mathbf{x}$
    - Increment weights

Multi-class Perceptron Learning Algorithm

- Multi-class perceptron loss $L(\hat{y}, y) = z_{\hat{y}} - z_y$
- If $\hat{y} = y$, then do nothing
- Else:
  - For the correct class y, $\dfrac{\partial L}{\partial z_y} = -1$
    - $(\nabla L)_y = -\mathbf{x}$
    - $(\theta_y)_{t+1} = (\theta_y)_t + \eta \mathbf{x}$
    - Increment weights
  - For the predicted class $\hat{y}$, $\dfrac{\partial L}{\partial z_{\hat{y}}} = 1$
    - $(\nabla L)_{\hat{y}} = \mathbf{x}$
    - $(\theta_{\hat{y}})_{t+1} = (\theta_{\hat{y}})_t - \eta \mathbf{x}$
    - Decrement weights

Multi-class Perceptron Learning Algorithm

- Multi-class perceptron loss $L(\hat{y}, y) = z_{\hat{y}} - z_y$
- If $\hat{y} = y$, then do nothing
- Else:
  - For the correct class y, $\dfrac{\partial L}{\partial z_y} = -1$
    - $(\nabla L)_y = -\mathbf{x}$
    - $(\theta_y)_{t+1} = (\theta_y)_t + \eta \mathbf{x}$
    - Increment weights
  - For the predicted class $\hat{y}$, $\dfrac{\partial L}{\partial z_{\hat{y}}} = 1$
    - $(\nabla L)_{\hat{y}} = \mathbf{x}$
    - $(\theta_{\hat{y}})_{t+1} = (\theta_{\hat{y}})_t - \eta \mathbf{x}$
    - Decrement weights
  - For other classes, do nothing

(Mini-)Batch Training

- What is a time step?
  - Stochastic gradient descent: update θ after every training example
    - Can result in very choppy movements

(Mini-)Batch Training

- What is a time step?
  - Stochastic gradient descent: update θ after every training example
    - Can result in very choppy movements
  - Batch gradient descent: update θ after processing the entire training set

(Mini-)Batch Training

- What is a time step?
  - Stochastic gradient descent: update θ after every training example
    - Can result in very choppy movements
  - Batch gradient descent: update θ after processing the entire training set
  - Minibatch gradient descent: update θ after m training examples
    - Gradient $=$ average of individual gradients

(Mini-)Batch Training

- Let $\mathbf{x}$ consist of the feature vectors $\mathbf{x}^{(i)}$ for each document i in the (mini-)batch of size m, stacked on top of each other

(Mini-)Batch Training

- Let $\mathbf{x}$ consist of the feature vectors $\mathbf{x}^{(i)}$ for each document i in the (mini-)batch of size m, stacked on top of each other
- $\mathbf{x} = \begin{bmatrix} x_1^{(1)} & \dots & x_n^{(1)} & 1 \\ \vdots & \ddots & \vdots & \vdots \\ x_1^{(m)} & \dots & x_n^{(m)} & 1 \end{bmatrix}$
- $\mathbf{y} = \begin{bmatrix} y^{(1)} \\ \vdots \\ y^{(m)} \end{bmatrix}$

(Mini-)Batch Training

- $\nabla L = \frac{1}{m}\left(\mathbf{x}^T(\hat{\mathbf{y}} - \mathbf{y})\right)$
  - What is $\mathbf{x}^T(\hat{\mathbf{y}} - \mathbf{y})$?

(Mini-)Batch Training

$$
\begin{bmatrix} x_1^{(1)} & \dots & x_1^{(m)} \\ \vdots & \ddots & \vdots \\ x_n^{(1)} & \dots & x_n^{(m)} \\ 1 & 1 & 1 \end{bmatrix}
\begin{bmatrix} \hat{y}^{(1)} - y^{(1)} \\ \vdots \\ \hat{y}^{(m)} - y^{(m)} \end{bmatrix}
= \begin{bmatrix} \sum_{i=1}^{m} (\hat{y}^{(i)} - y^{(i)}) x_1^{(i)} \\ \vdots \\ \sum_{i=1}^{m} (\hat{y}^{(i)} - y^{(i)}) x_n^{(i)} \\ \sum_{i=1}^{m} (\hat{y}^{(i)} - y^{(i)}) \end{bmatrix}
$$

$$
= \begin{bmatrix} \sum_{i=1}^{m} \left(\frac{\partial L}{\partial w_1} \right)^{(i)} \\ \vdots \\ \sum_{i=1}^{m} \left(\frac{\partial L}{\partial w_n} \right)^{(i)} \\ \sum_{i=1}^{m} \left(\frac{\partial L}{\partial b} \right)^{(i)} \end{bmatrix}
$$

$$
= \sum_{i=1}^{m} (\nabla L)^{(i)}
$$

(Mini-)Batch Training

- $\nabla L = \frac{1}{m}\left(\mathbf{x}^T(\hat{\mathbf{y}} - \mathbf{y})\right)$
  - What is $\mathbf{x}^T(\hat{\mathbf{y}} - \mathbf{y})$?
    - It computes the sum of the gradients for each document i in the mini-batch!
    - Then to get the average gradient, we just divide by m