Chapter 3

Classification and Automatic Differentiation

3.1 Functions vs conditional probability

Function:

$$x \to y$$
$$y = f(x; \theta)$$

Completely deterministic (1 action to 1 possibility)

Conditional probability
1 to many possibilities

- Logits:
  - Magnitudes of unnormalized have no meaning
- Softmax: logits $\to$ probability vector

$$\textsc{Softmax}(\mathbf{z}) = \frac{1}{\sum_{j=1}^{K} \exp(z_j)} \begin{bmatrix} \exp(z_1) \\ \vdots \\ \exp(z_k) \end{bmatrix} = \begin{bmatrix} \textsc{Softmax}(\mathbf{z})_1 \\ \vdots \\ \textsc{Softmax}(\mathbf{z})_k \end{bmatrix} = \begin{bmatrix} p_1 \\ \vdots \\ p_k \end{bmatrix} \tag{3.1}$$

3.2 The Story So Far

You know

- how to create a probability vector $\mathbf{p}$ from (raw) logits $\mathbf{z}$
- how to draw labels from y from it
- But where does $\mathbf{z}$ come from?

3.3 Multiclass classification

- A classifier

$$\chi \in \mathbf{x} \to f(\mathbf{x};\theta) \in \mathbb{R}^K$$

- A linear classifier

$$\mathbf{W} \in \mathbb{R}^{d\times K}, \theta = \mathbf{W}$$
$$f(\mathbf{x};\theta) = \mathbf{x}\mathbf{W}$$

  where d is the input dimension (e.g. # of pixels), K is the # of classes, and $\mathbf{x}$ is a row vector.

  – For MNIST digit recognition $\approx 92\%$ test accuracy using a “good” linear model.

- What does the output vector mean?

3.4 Conditional Probability

Whenever you use cross-entropy to train a linear model/deep model learning, you are assuming that your model is reflective of real world probability.

3.5 Negative Log-Likelihood

$$\min_\theta \sum_{i=1}^{m} -\log\left(\operatorname{softmax}\left(f\left(\mathbf{x}^{(i)};\theta\right)\right)_{y^{(i)}}\right) \tag{3.2}$$

3.6 The Cross Entropy Loss

$$-\log\left(\operatorname{softmax}\left(\mathbf{z}^{(i)}\right)_{y^{(i)}}\right) =: L\left(\mathbf{z}^{(i)}, y^{(i)}\right) \tag{3.3}$$

where $\mathbf{z}^{(i)}$ is the prediction and $y^{(i)}$ is the label.

- In regression, first arg and second arg are both floats.
- In classification, first arg is a vector of floats and second arg is a (discrete) integer

3.7 How to Calculate the Derivative

- W.grad means $\frac{d}{dW}$
- W.shape should be equal to W.grad.shape

$$\frac{\partial}{\partial \mathbf{W}} \mathbf{x}\mathbf{W} \Rightarrow x^T$$

x kind of stands as a constant, but you need to take the transpose for the shapes to make sense.

$$\frac{\partial}{\partial \mathbf{W}} \left(L\left(\mathbf{x}\mathbf{W}, y\right)\right) = \mathbf{x}^T \frac{\partial L}{\partial \mathbf{z}} \left(\mathbf{x}\mathbf{W}, y\right)$$

3.7.1 Derivative of the Softmax

Case 1: $j \neq y$

Numerator does not depend on z_j

$$\begin{aligned} \frac{\partial}{\partial z_j} \left(\frac{\exp(z_j)}{\sum_{l=1}^{K} \exp(z_l)} \right) &= -\frac{\exp(z_y)}{\left(\dots\right)^2} \quad = \dots \\ &= -p_y \times p_j \end{aligned}$$

Case 2: $j = y$

$$\begin{aligned} \frac{\partial}{\partial z_y} \left(\frac{\exp(z_j)}{\sum_{l=1}^{K} \exp(z_l)} \right) &= \frac{\partial}{\partial z_y} \left(\frac{1}{\sum_{l=1}^{K} \exp(z_l - z_y)} \right) \\ &= \left(\frac{1}{\sum_{l=1}^{K} \exp(z_l - z_y)} \right)^2 \frac{\partial}{\partial z_y} \sum_{l=1}^{K} \exp(z_l - z_y) \\ &= \dots \\ &= p_y(1 - p_y) \end{aligned}$$

3.7.2 Derivative

$$\begin{aligned} \frac{\partial L}{\partial \mathbf{z}} \left(\mathbf{x}\mathbf{W}, y\right) &= \frac{\partial}{\partial \mathbf{z}} \left(-\log\left(\text{softmax}(\mathbf{z})_y\right)\right) \\ \frac{\partial}{\partial z_j} \left(-\log\left(\text{softmax}(\mathbf{z})_y\right)\right) &= -\frac{1}{\text{softmax}(\mathbf{z}_y)} \times \frac{\partial}{\partial z_j} (\text{softmax}(\mathbf{z})_y) \\ \frac{\partial}{\partial z_j} (\text{softmax}(\mathbf{z})_y) &= \frac{\partial}{\partial z_j} \left(\frac{\exp(z_j)}{\sum_{l=1}^{K} \exp(z_l)} \right) \end{aligned}$$

$$\frac{\partial L}{\partial \mathbf{z}}(\mathbf{z}, y) = \mathbf{p} - \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \leftarrow \ y\text{-th position} \tag{3.4}$$

If $p ==$ one hot at y, then derivative is zero for that particular datapoint (gradient contribution is zero). The further away you are from being one hot at y, the "larger" the gradient.

3.8 Automatic Differentiation

What is in an *automatic differentiation* (AD) library?

- Tensors with "grad" attached (i.e., numbers in multidim arrays)
- Operators (e.g., addition, matrix multiplication, logarithm, ...)
- Computation graph to keep track of the operations on tensors

3.9 Running Example

$$f(x, y) = \log\left(1 + e^{-(x+y)}\right)(x + y)$$

3.9.1 Computing $f(x, y)$

- Easy to compute $f(x, y)\dots$

3.9.2 Computational Graph

- Leaves have no incoming edges
- Intermediate `ag.Scalars` have blue discs
- Constants are gray plates

3.9.3 Ingredients for Algorithmic Computation

-

3.9.4 Quantities

3.9.5 Goal: (Partial) Derivatives

- Not just computing $f(x, y)$, but also:

$$\frac{\partial}{\partial x} f(x, y), \frac{\partial}{\partial y} f(x, y)$$

3.9.6 The `ag.Scalar` Class

`ag.Scalar` class – value plus additional information:

grad – keeps track of the gradient

inputs – tracks the inputs leading to a value

op – label for the operation that produced the quantities (for visualization)

_backward – function for propagating the gradient to the inputs

label – label for drawing the computational graph (for visualization)

Pink bullet points are the most mathematically important. Rest are just for readability. If you lose the final ag.Scalar object, then you basically lost the graph.

3.9.7 Key Property

For assigning internal quantities $z_{l+1}, z_{l+2}, \ldots$:

3.9.8 Chain Rule for $\frac{\partial f}{\partial z_7}$

By the chain rule:

$$\begin{aligned}\frac{\partial}{\partial z_7} f(x,y) &= \frac{\partial}{\partial z_8} f(x,y) \times \frac{\partial z_8}{\partial z_7} \\ &= 1 \times \frac{\partial z_8}{\partial z_7} \\ &= \frac{\partial(z_7 \times z_3)}{\partial z_7}\end{aligned}$$