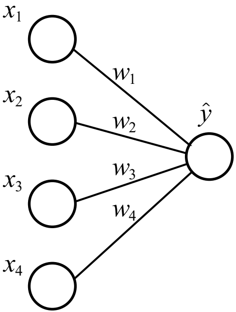
Backpropagation

Foundations of Data Analysis

April 19, 2022

Single Layer Model



Prediction:

$$\begin{aligned}\hat{y} &= \phi(Wx) \\ &= \phi(w_1x_1 + w_2x_2 + w_3x_3 + w_4x_4)\end{aligned}$$

Minimize loss between prediction, $\hat{y}$, and true value, y:

$$\mathcal{L}(y, \hat{y})$$

This represents many different models we've seen!

Loss Functions for Regression

Dependent variable data: $y \in \mathbb{R}$
Model predictions: $\hat{y} \in \mathbb{R}$

Mean squared error (MSE): (Linear regression)

$$\mathcal{L}(y, \hat{y}) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

Mean absolute error (MAE):

$$\mathcal{L}(y, \hat{y}) = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

Loss Functions for Classification

Binary labels: $y \in \{-1, +1\}$
Continuous score predictions: $\hat{y} \in \mathbb{R}$

Zero-One loss: (Perceptron)

$$\mathcal{L}(y, \hat{y}) = \begin{cases} 1 & \text{if } y\hat{y} \leq 0 \\ 0 & \text{if } y\hat{y} > 0 \end{cases}$$

Hinge loss: (Support Vector Machines)

$$\mathcal{L}(y, \hat{y}) = \frac{1}{n} \sum_{i=1}^{n} \max(0, 1 - \hat{y}_i y_i)$$

Loss Functions for Classification

Binary labels: $y \in \{0, 1\}$
Probability predictions: $\hat{y} \in [0, 1]$ predicts $p(y = 1 \mid x)$

Cross entropy: (Logistic regression)

$$\mathcal{L}(y, \hat{y}) = \frac{1}{n} - \sum_{i=1}^{n} (y_i \ln \hat{y}_i + (1 - y_i) \ln(1 - \hat{y}_i))$$

Chain Rule

Given two differentiable functions,

$$f : \mathbb{R} \to \mathbb{R}, \quad g : \mathbb{R} \to \mathbb{R},$$

the derivative of their composition is:

$$\frac{d}{dx}\left[f(g(x))\right] = f'(g(x))g'(x)$$

Multivariate Mappings

Given a multivariate mapping,

$$g : \mathbb{R}^p \to \mathbb{R}^q,$$

we can write it as q multivariate functions:

$$g(x_1, \ldots, x_p) = (g_1(x_1, \ldots, x_p), \ldots, g_q(x_1, \ldots, x_p)).$$

Jacobian Matrix

Partial derivatives: $D_j g_i = \frac{\partial g_i}{\partial x_j}$

The **Jacobian matrix** is the $q \times p$ matrix of partial derivatives:

$$Dg = \begin{pmatrix} D_1 g_1 & D_2 g_1 & \cdots & D_p g_1 \\ D_1 g_2 & D_2 g_2 & \cdots & D_p g_2 \\ \vdots & \vdots & & \vdots \\ D_1 g_q & D_2 g_q & \cdots & D_p g_q \end{pmatrix}$$

Multivariate Chain Rule

Given two multivariate mappings,

$$f : \mathbb{R}^q \to \mathbb{R}^r, \quad g : \mathbb{R}^p \to \mathbb{R}^q$$

the Jacobian matrix of their composition is:

$$D\left[f(g(x))\right] = Df(g(x))Dg(x).$$

Note: This is a matrix multiplication on the right.

Gradient Chain Rule

Given a multivariate function,

$$f : \mathbb{R}^q \to \mathbb{R},$$

The Jacobian matrix is the same thing as the transposed gradient:

$$Df(x) = \begin{pmatrix} \frac{\partial f}{\partial x_1} & \cdots & \frac{\partial f}{\partial x_q} \end{pmatrix} = \nabla f(x)^T$$

Gradient Chain Rule

Given two multivariate mappings,

$$f : \mathbb{R}^q \to \mathbb{R}, \quad g : \mathbb{R}^p \to \mathbb{R}^q,$$

the gradient is the transpose of the Jacobian chain rule equation:

$$\nabla \left[f(g(x)) \right] = \left[Df(g(x)) Dg(x) \right]^T = Dg(x)^T \nabla f(g(x)).$$

Matrix Derivatives

Think of matrix-vector multiplication as a mapping of a vector x **and** a matrix W:

$$g(W, x) = Wx = \begin{pmatrix} W_{11} & W_{12} & \cdots & W_{1p} \\ W_{21} & W_{22} & \cdots & W_{2p} \\ \vdots & \vdots & & \vdots \\ W_{q1} & W_{q2} & \cdots & W_{qp} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_p \end{pmatrix}$$

Input dimension: p
Output dimension: q
Weight matrix W is $q \times p$

Matrix Derivatives

The kth entry in the output is:

$$g_k(W, x) = W_{k1}x_1 + W_{k2}x_2 + \cdots + W_{kp}x_p$$

Partial derivative wrt W_{ij} is

$$D_{ij}g_k = \frac{\partial g_k}{\partial W_{ij}} = \begin{cases} x_j & \text{if } i = k \\ 0 & \text{if } i \neq k \end{cases}$$

Notice this is a 3D array!

Single-Layer Neural Network

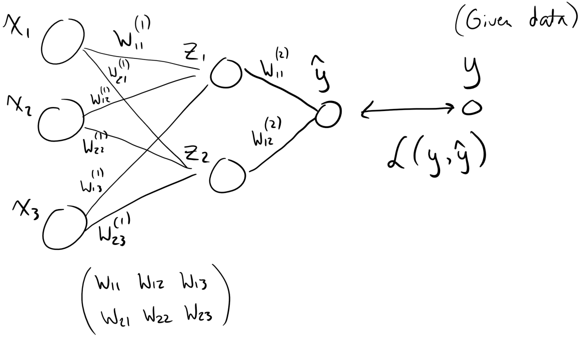
$$z = \phi(Wx) = \phi(g(W, x))$$

Gradient for weights:

$$\frac{\partial z_k}{\partial W_{ij}} = \phi'(g_k(W, x))D_{ij}g_k(W, x)$$

Note: ϕ' is the (univariate) derivative of ϕ

Two-Layer NN



Layer 1: $z = \phi(W^{(1)}x)$
Layer 2: $\hat{y} = \phi(W^{(2)}z)$
Final Loss Function: $\mathcal{L}(y, \hat{y})$

Second-Layer Derivative

Gradient of loss wrt $W^{(2)}$:

$$\frac{\partial \mathcal{L}}{\partial W_{ij}^{(2)}} = \left(\frac{d\mathcal{L}}{d\hat{y}}\right) \left\{ \frac{\partial \hat{y}}{\partial W_{ij}^{(2)}} \right\}$$

$$= \left(\frac{d\mathcal{L}}{d\hat{y}}\right) \left\{ \phi'(g(W^{(2)}, z)) D_{ij} g(W^{(2)}, z) \right\}$$

First-Layer Derivative

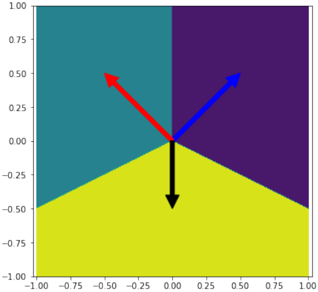
Gradient of loss wrt $W^{(1)}$:

$$
\begin{aligned}
\frac{\partial \mathcal{L}}{\partial W_{ij}^{(1)}} &= \left(\frac{d\mathcal{L}}{d\hat{y}}\right) \{D_z \hat{y}\} \left[\frac{\partial z}{\partial W_{ij}^{(1)}}\right] \\
&= \left(\frac{d\mathcal{L}}{d\hat{y}}\right) \left\{\phi'(g(W^{(2)}, z)) D_z g(W^{(2)}, z)\right\} \times \\
&\qquad \times \left[D_{ij} g(W^{(1)}, x)\right]
\end{aligned}
$$

Softmax

Extends logistic regression to more than 2 classes.

- Class labels: $y = 1, 2, \ldots, K$
- Weight vector for each class: $w_1, \ldots, w_K \in \mathbb{R}^{d+1}$



$$p(y = k \mid x) = \frac{\exp(x^T w_k)}{\sum_{j=1}^{K} \exp(x^T w_j)}$$