PHS597 - Deep Learning

Renan Sauteraud

Mon Feb 3 16:40:22 2020

Contents

1 Unsupervised learning 1

2 Nonnegative matrix factorization (NMF) 1

3 Deep Feedforward network 1

3.1 Basic structure 2

4 Computation for Neural Networks 3

4.1 Issues 3

4.2 Gradient descent 3

4.3 Deep learning optimization 3

1 Unsupervised learning

Dissimilarity metrics

For continuous vars: - Square differences - Absolute differences - Correlation coefficients

For categorical vars: - Symmetric matrix of identity

For ordinal vars: - Rank based measures

2 Nonnegative matrix factorization (NMF)

Used as “soft clustering” method. Finding a lower rank matrix to represent a large matrix.

3 Deep Feedforward network

Also called Multi Layer Perceptron (MLP).

Goal: For some intput X with outcome Y, approximate a function f. $Y = f(X)$

Constructed by composing functions together

$$f(x) = f^{(3)}(f^{(2)}(f^{(1)}(x)))$$

Where $f^{(1)}$ would be the function between the input layer and the first hidden layer.

3.1 Basic structure

X -> $h^{(1)}$ -> Y

Assuming only one hidden layer

$$h^1 = g^1(w^{1T}X + b^1)$$
$$\hat{y} = g^2(w^{2T}h^1 + b^2)$$

- g: Non-linear activation function.
- w: Weights (To be learned)
- Loss function: Used to evaluate difference between observed and predicted outcome.

3.1.1 Choice of activation function

Note: Activation function has to be non-linear. Otherwise, if every function in the network is linear, any number of layers could be reduced to a two layers input-output model.

For the output node, the choice is dictated by the type of outcome. e.g: You can use sigmoid function $\sigma(x) = \frac{exp(x)}{1+exp(x)}$ for binary data or softmax $softmax(Z)(j) = \frac{exp(Z_j)}{\sum_j exp(Z_j)}$ for categorical data.

For **hidden layers**: Rectified linear unit (ReLU)

$$g(z) = max(0, z)$$

Benefits: No vanishing or exploding gradient prolem (only non differentiable at 0). Limitations: Half the function has gradient 0. If the value is <0, the gradient is 0, so the parameters cannot be updated (It's like the neuron was not used). An extension to counter this is to add a non-zero slope to the 0 part of ReLU. Specifically

$$g(z, \alpha) = max(0, z) + \alpha \times min(0, z)$$

- $\alpha = -1$ leads to $g(X) = |X|$ absolute value rectification.
- $\alpha = 0.01$ (or other small values) is leaky ReLU.
- If α is a parameter to be estimated it is called parametric ReLU.

Note: Exploding gradient is when large error gradient accumulate, which leads to large updates to the weights which in turn leads to the model being unstable.

3.1.2 Choice of loss function

Mostly depends on the response:

- Squared error loss for continuous response $L(\hat{y}, y) = \sum_i (\hat{y}_i - y_i)^2$
- -log(likelihood) for binary outcomes $L(\hat{y}, y) == -\sum_i log(\hat{y}_(i) + (1 - y_i)log(1 - \hat{y}_i)$

3.1.3 Backward/Forward propagation

It is not an optimization algorithm. In essence, it is the chain rule for computing derivatives for composition of functions.

Chain rule: Given $y = g(x)$ and $z = f(y) = f(g(x)$ chain rule can be applied to compute the derivatite of z with resoect to x.

e.g: For `z <- y <- x <- w` with $z = f(f(f(w)))$.

Apply the chain rule twice: $\frac{\partial z}{\partial w} = \frac{\partial z}{\partial y}\frac{\partial y}{\partial w} = \frac{\partial z}{\partial y}(y)\left[\frac{\partial y}{\partial x}(y)\frac{\partial x}{\partial w}(w)\right]$

Table 1: Transformations Associated with the Johnson System

Johnson Family	Transformation	Parameter Conditions	X Condition
S_B	$Z = \gamma + \eta ln(\frac{X-\epsilon}{\lambda+\epsilon-X})$	$\eta, \lambda > 0, -\infty < \gamma, \epsilon < \infty$	$\epsilon < X < \epsilon + \lambda$
S_L	$Z = \gamma + \eta ln(X - \epsilon)$	$\eta > 0, -\infty < \gamma, \epsilon < \infty$	$X > \epsilon$
S_U	$Z = \gamma + \eta \sinh^{-1}(\frac{X-\epsilon}{\lambda})$	$\eta, \lambda > 0, -\infty < \gamma, \epsilon < \infty$	$-\infty < X < \infty$

i.e: Calculate derivative of the function at each layer with respect to its predecessor.

Note: If there are multiple path (edges) leading to a node. You compute the derivatives by adding the derivatives from all paths.

Unlike Gradient descent that optimizes parameter values, Forward/Backward prop only calculate derivatives.

3.1.4 Gradient descent algorithm

Goal: Find the minimum of a function $f(x)$ by updating the value of parameters in the direction of the gradient descent.

$$x_{i+1} = x_i + \alpha f'(x_i)$$

Where α is the learning rate. If it is too small, convergence is slow. If it is too big, it may overshoot minimums, also leading to a slow convergence.

4 Computation for Neural Networks

4.1 Issues

- Overflow and underflow (small numbers rounded to 0) are common issues.
- Ill conditioning: When small change in the data can lead to big changes in the parameter estimates. Ususally caused by near-singular matrices. Formally, its definition is $\underset{max}{i,j}|\frac{\lambda_i}{\lambda_j}|$, where λ_i, j are eigenvalues.
- Saddle point: When the derivative is 0 but you are neither at a minimum or maximum. This can be diagnosed by second order derivatives.

Depending on the task, global minimum do not necessarily have to be reached as long as a satisfactory accuracy is reached.

4.2 Gradient descent

4.3 Deep learning optimization

By the CLT, the expectation of the loss over a large enough subset of observation approximate the true loss function. You can thus minimize over a subset of all your data. This is stochastic gradient descent (SGD), or minibatch.

table | c1 | c2 |
a | b | c |
d | e | f |

a	b	c	d