Deep Learning in NLP

Alexey Sorokin
Vasily Konovalov
Darya Moroz

Spring 2020

Content

1. What can be approximated by NN?
2. How to define the error function?
3. How to optimize the error function?
4. How to calculate the gradients?

Why do need nonlinearity?

Why do need nonlinearity?

If the activation functions of all the hidden units in a network are taken to be linear, then for any such network we can always find an equivalent network without hidden units.

Why do need nonlinearity?

If the activation functions of all the hidden units in a network are taken to be linear, then for any such network we can always find an equivalent network without hidden units.

This follows from the fact that the composition of successive linear transformations is itself a linear transformation.

Universal Approximators

The class of deep neural networks is a universal approximator if and only if the activation function is not polynomial. [a]

---

[a] Leshno, Moshe; Lin, Vladimir Ya.; Pinkus, Allan; Schocken, Shimon (January 1993). "Multilayer feedforward networks with a nonpolynomial activation function can approximate any function".

Universal Approximators

The class of deep neural networks is a universal approximator if and only if the activation function is not polynomial. [a]

[a] Leshno, Moshe; Lin, Vladimir Ya.; Pinkus, Allan; Schocken, Shimon (January 1993). "Multilayer feedforward networks with a nonpolynomial activation function can approximate any function".

The ReLU networks with width $n+1$ is sufficient to approximate any continuous function of n-dimensional input variables. [a]

[a] Hanin, B. (2018). Approximating Continuous Functions by ReLU Nets of Minimal Width. arXiv preprint arXiv:1710.11278.

Regression: Error function and optimization

$$RSS(w) = \sum_{i=1}^{N} (y_i - x_i^T w)^2$$

How to find an optimal w^*?

- RSS is a convex function
- $w^* = (X^T X)_{-1} X^T y$ in case when X is not singular
- and you can apply gradient descent as well

Regression: Error function and optimization

$$RSS(w) = \sum_{i=1}^{N} (y_i - x_i^T w)^2$$

How to find an optimal w^*?

- RSS is a convex function
- $w^* = (X^T X)_{-1} X^T y$ in case when X is not singular
- and you can apply gradient descent as well 🙂

But this is often not the case!

Error function: Classification and optimization

Accuracy as an error function for classification

Error function: Classification and optimization

Accuracy as an error function for classification
Constant piece-wise function

Error function: Classification and optimization

Accuracy as an error function for classification
Constant piece-wise function
Gradient either equals 0 or doesn't exists

Error function: Classification and optimization

Accuracy as an error function for classification
Constant piece-wise function
Gradient either equals 0 or doesn't exists
The same applies to ranking

Cross-entropy as an error function

$$KL(P \parallel Q) = \sum_{i=1}^{n} p(x_i) \log \frac{p(x_i)}{q(x_i)}$$

Properties

- $KL(P \parallel Q) \neq KL(Q \parallel P)$
- $KL(P \parallel Q) \geq 0$
- $KL(P \parallel Q) = 0$ when $\log \frac{P}{Q} = 0$

[1]

[1] https://medium.com/activating-robotic-minds/demystifying-kl-divergence-7ebe4317ee68

Cross-entropy as an error function

$$KL(P \parallel Q) = H(P) + H(P, Q)$$

$$KL(P \parallel Q) = H(P, Q)$$

Cross-entropy as an error function

$$KL(P \parallel Q) = H(P) + H(P, Q)$$

$$KL(P \parallel Q) = H(P, Q)$$

Case: Binary classification

$$L(\theta) = -\frac{1}{N}\sum_{i=1}^{N}(y_i \log \hat{y}_i(\theta)) + (1 - y_i)\log(1 - \hat{y}_i(\theta)))$$

Gradient Descent

Apparently there is no analytical solutions for these error functions

[2] https://ml-cheatsheet.readthedocs.io/en/latest/gradient_descent.html

Gradient Descent

Apparently there is no analytical solutions for these error functions
But if we can evaluate the function at the point we can apply numerical optimization

[2] `https://ml-cheatsheet.readthedocs.io/en/latest/gradient_descent.html`

Gradient Descent

Apparently there is no analytical solutions for these error functions
But if we can evaluate the function at the point we can apply numerical optimization
Gradient descent is an optimization algorithm used to minimize some function by iteratively moving in the direction of steepest descent as defined by the negative of the gradient. [2]

[2] https://ml-cheatsheet.readthedocs.io/en/latest/gradient_descent.html

Gradient Descent

$$E(\theta) = \sum_{(x,y) \in D} E(f(x, \theta), y)$$

Gradient Descent

$$E(\theta) = \sum_{(x,y)\in D} E(f(x,\theta), y)$$

$$\begin{aligned}\theta_t &= \theta_{t-1} - \eta \nabla E(\theta_{t-1}) \\ &= \theta_{t-1} - \eta \sum_{(x,y)\in D} \nabla E(f(x,\theta_{t-1}), y)\end{aligned}$$

Stochastic Gradient Descent

$$\theta_{t-1} = \theta_{t-1} - \eta \nabla E(f(x, \theta_{t-1}), y)$$

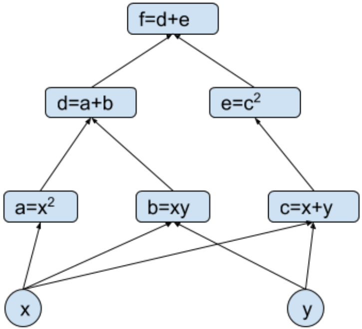
Computational Graph

Computational Graph - a graph of "simple"functions that compose "complex"function.

Computational Graph

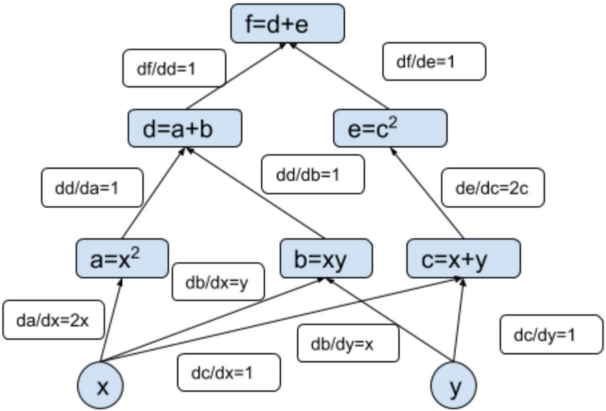
$$f = x^2 + xy + (x + y)^2$$

Credit: Николенко, Кадурин, Архангельская: Глубокое обучение. Погружение в мир нейронных сетей



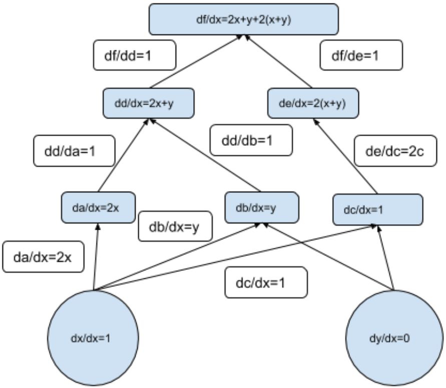
Computational Graph

$$f = x^2 + xy + (x + y)^2$$



Computational Graph

$$f = x^2 + xy + (x + y)^2$$



Computational Graph

$$f = x^2 + xy + (x+y)^2$$