Neural Nets, Deep-Learning and applications (NLP)

A. Allauzen

Université Paris-Sud / LIMSI-CNRS

08/01/2016

Outline

1. Known issues and solutions
   - Regularization and Dropout
   - The vanishing gradient issue

Outline

1. Known issues and solutions
   - Regularization and Dropout
   - The vanishing gradient issue

Outline

1. Known issues and solutions
   - Regularization and Dropout
   - The vanishing gradient issue

Regularization l^2 or gaussian prior or weight decay

The basic way :

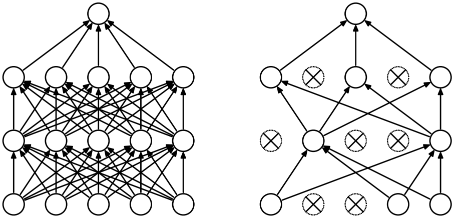
$$\mathcal{L}(\boldsymbol{\theta}) = \sum_{i=1}^{N} l(\boldsymbol{\theta}, \boldsymbol{x}_{(i)}, c_{(i)}) + \frac{\lambda}{2}||\boldsymbol{\theta}||^2$$

- The second term is the regularization term.
- Each parameter has a gaussian prior : $\mathcal{N}(0, 1/\lambda)$.
- λ is a hyperparameter.
- The update has the form :

$$\boldsymbol{\theta} = (1 + \eta_t \lambda)\boldsymbol{\theta} - \eta_t \nabla_{\boldsymbol{\theta}}$$

Dropout

A new regularization scheme (Srivastava and Salakhutdinov2014)



(a) Standard Neural Net (b) After applying dropout.

- For each training example : randomly turn-off the neurons of hidden units (with $p = 0.5$)
- At test time, use each neuron scaled down by p

- Dropout serves to separate effects from strongly correlated features and
- prevents co-adaptation between units
- It can be seen as averaging different models that share parameters.
- It acts as a powerful regularization scheme.

Dropout - implementation

The layer should keep :

- $\boldsymbol{W}^{(l)}$: the parameters
- $f^{(l)}$: its activation function
- $\boldsymbol{x}^{(l)}$: its input
- $\boldsymbol{a}^{(l)}$: its pre-activation associated to the input
- $\boldsymbol{\delta}^{(l)}$: for the update and the back-propagation to the layer $l-1$
- $\boldsymbol{m}^{(l)}$: the dropout mask, to be applied on $\boldsymbol{x}^{(l)}$

Forward pass

For $l = 1$ to $(L-1)$

- Compute $\boldsymbol{y}^{(l)} = f^{(l)}(\boldsymbol{W}^{(l)}\boldsymbol{x}^{(l)})$
- $\boldsymbol{x}^{(l+1)} = \boldsymbol{y}^{(l)} = \boldsymbol{y}^{(l)} \circ \boldsymbol{m}^{(l)}$

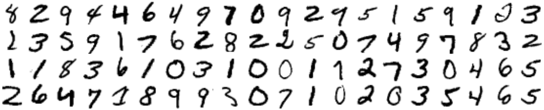
$\boldsymbol{y}^{(L)} = f^{(L)}(\boldsymbol{W}^{(L)}\boldsymbol{x}^{(L)})$

Outline

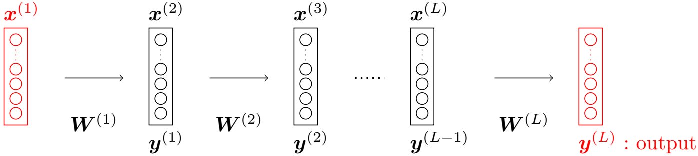
1. Known issues and solutions
   - Regularization and Dropout
   - The vanishing gradient issue

Experimental observations (MNIST task) - 1

The MNIST database



Comparison of different depth for feed-forward architecture



- Hidden layers have a sigmoid activation function.
- The output layer is a softmax.

Experimental observations (MNIST task) - 2

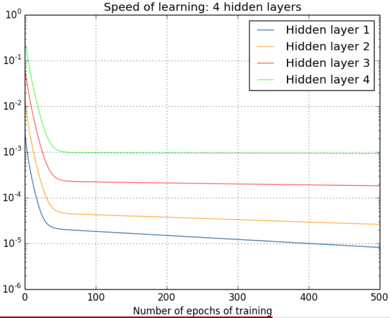
Varying the depth

- Without hidden layer : $\approx 88\%$ accuracy
- 1 hidden layer (30) : $\approx 96.5\%$ accuracy
- 2 hidden layer (30) : $\approx 96.9\%$ accuracy
- 3 hidden layer (30) : $\approx 96.5\%$ accuracy
- 4 hidden layer (30) : $\approx 96.5\%$ accuracy

Experimental observations (MNIST task) - 2

Varying the depth

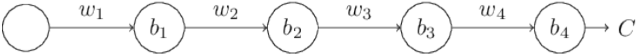
- Without hidden layer : ≈ 88% accuracy
- 1 hidden layer (30) : ≈ 96.5% accuracy
- 2 hidden layer (30) : ≈ 96.9% accuracy
- 3 hidden layer (30) : ≈ 96.5% accuracy
- 4 hidden layer (30) : ≈ 96.5% accuracy



(From `http://neuralnetworksanddeeplearning.com/chap5.html`)

Intuitive explanation

Let consider the simplest deep neural network, with just a single neuron in each layer.



w_i, b_i are resp. the weight and bias of neuron i and C some cost function.

Compute the gradient of C $w.r.t$ the bias b_1

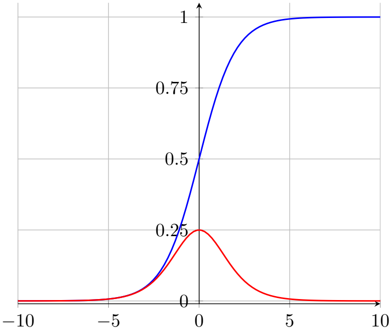
$$\frac{\partial C}{\partial b_1} = \frac{\partial C}{\partial y_4} \times \frac{\partial y_4}{\partial a_4} \times \frac{\partial a_4}{\partial y_3} \times \frac{\partial y_3}{\partial a_3} \times \frac{\partial a_3}{\partial y_2} \times \frac{\partial y_2}{\partial a_2} \times \frac{\partial a_2}{\partial y_1} \times \frac{\partial y_1}{\partial a_1} \times \frac{\partial a_1}{\partial b_1} \tag{1}$$

$$= \frac{\partial C}{\partial y_4} \times \sigma'(a_4) \times w_4 \times \sigma'(a_3) \times w_3 \times \sigma'(a_2) \times w_2 \times \sigma'(a_1) \tag{2}$$

$$\tag{3}$$

Intuitive explanation - 2

The derivative of the activation function : σ'



$$\sigma'(x) = \sigma(x)(1 - \sigma(x))$$

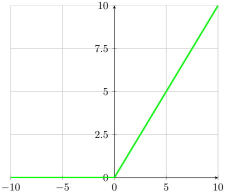
But weights are initialize around 0.

The different layers in our deep network are learning at vastly different speeds :

- when later layers in the network are learning well,
- early layers often get stuck during training, learning almost nothing at all.

Solutions

Change the activation function (Rectified Linear Unit or ReLU)



- Avoid the vanishing gradient
- Some units can "die"

See (Glorot et al.2011) for more details

Do pre-training when it is possible

See (Hinton et al.2006; Bengio et al.2007) :

when you cannot really escape from the initial (random) point, find a good starting point.

More details

See (Hochreiter et al.2001; Glorot and Bengio2010; LeCun et al.2012)

Yoshua Bengio, Pascal Lamblin, Dan Popovici, and Hugo Larochelle.
2007.
Greedy layer-wise training of deep networks.
In B. Schölkopf, J.C. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems 19*, pages 153–160. MIT Press.

Xavier Glorot and Yoshua Bengio.
2010.
Understanding the difficulty of training deep feedforward neural networks.
In *JMLR W&CP : Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics (AISTATS 2010)*, volume 9, pages 249–256, May.

Xavier Glorot, Antoine Bordes, and Yoshua Bengio.
2011.
Deep sparse rectifier neural networks.
In Geoffrey J. Gordon and David B. Dunson, editors, *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics (AISTATS-11)*, volume 15, pages 315–323. Journal of Machine Learning Research - Workshop and Conference Proceedings.

Geoffrey E. Hinton, Simon Osindero, and Yee-Whye Teh.
2006.
A fast learning algorithm for deep belief nets.
Neural Computation, 18(7) :1527–1554, JUL.

S. Hochreiter, Y. Bengio, P. Frasconi, and J. Schmidhuber.
2001.
Gradient flow in recurrent nets : the difficulty of learning long-term dependencies.
In Kremer and Kolen, editors, *A Field Guide to Dynamical Recurrent Neural Networks*. IEEE Press.

Yann LeCun, Léon Bottou, Genevieve Orr, and Klaus-Robert Müller.
2012.
Efficient backprop.
In Grégoire Montavon, GenevièveB. Orr, and Klaus-Robert Müller, editors, *Neural Networks : Tricks of the Trade*, volume 7700 of *Lecture Notes in Computer Science*, pages 9–48. Springer Berlin Heidelberg.

Nitish Srivastava and Ruslan Salakhutdinov.
2014.
Multimodal learning with deep boltzmann machines.
Journal of Machine Learning Research, 15 :2949–2980.