Deep Learning and Temporal Data Processing

0 - Gradient Descent

Andrea Palazzi

July 10th, 2017

University of Modena and Reggio Emilia

Agenda

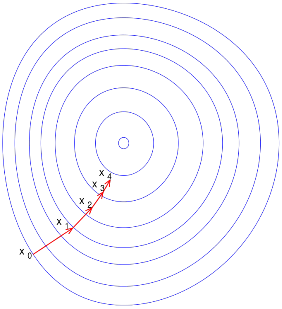
Gradient Descent

Credits

References

Gradient Descent

Gradient descent is an iterative optimization algorithm for finding the minimum of a function. How? Take step proportional to the negative of the gradient of the function at the current point.



Gradient descent on a series of level sets

Gradient Descent Update

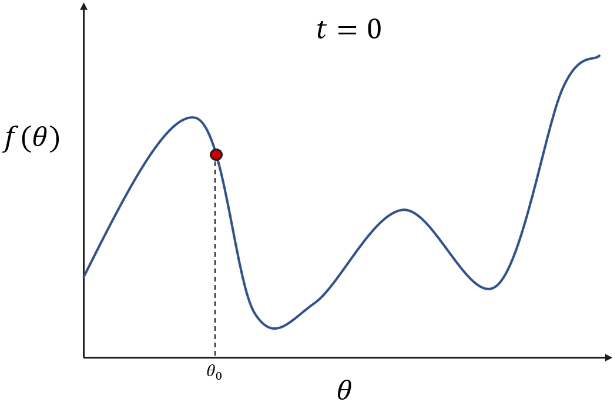
If we consider a function $f(\boldsymbol{\theta})$, the **gradient descent update** can be expressed as:

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} f(\boldsymbol{\theta}) \tag{1}$$

for each parameter θ_j.

The size of the step is controlled by **learning rate** α.

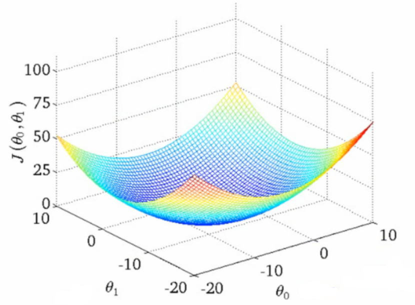
Gradient Descent for 1-d function $f(\theta)$.



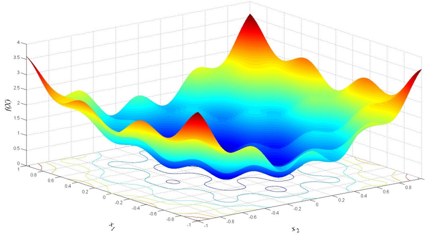
$$\frac{df}{d\theta} < 0$$

$$\theta_{new} = \theta_{old} - \alpha \frac{df}{d\theta}$$

Turns out that if the function is **convex** gradient descent will converge to the **global minimum**. For **non-convex** functions, it may converge to **local minima**.



Convex Function



Non-Convex Function

Gradient Descent

Gradient descent is often used in machine learning to **minimize a cost function**, usually also called *objective* or *loss* function and denoted $L(\cdot)$ or $J(\cdot)$.

The cost function depends on the model's parameters and is a proxy to evaluate model's performance. Generally speaking, in this framework minimizing the cost equals to maximizing the effectiveness of the model.

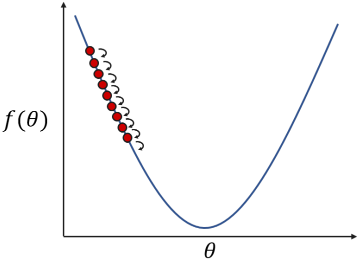
Stochastic Gradient Descent

In principle, to perform a single update step you should run through all your training examples. This is known as **batch gradient descent**.

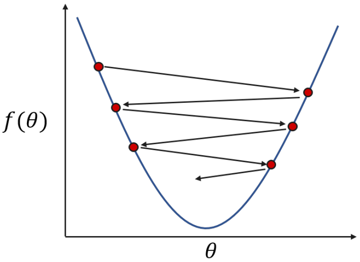
A different strategy is the one of **minibatch stochastic gradient descent**. In this case, only a small subset of the training dataset is considered at each update step.

In the extreme case in which only a random example of the training set is considered to perform the update step, we talk of **stochastic gradient descent**.

Choosing the the right **learning rate** α is essential to correctly proceed towards the minimum. A step *too small* could lead to an extremely *slow* convergence. If the step is *too big* the optimizer could *overshoot* the minimum or even *diverge*.



Learning Rate too small



Learning Rate too big

In practice, it's quite rare to see the procedure described above (so called **vanilla SGD**) used for optimization in the real-world.

Conversely, a number of cutting-edge optimizers [2, 1, 3] are commonly used. However, these advanced optimization techniques are out of the scope of this short overview.

Credits

These slides heavily borrow from a number of awesome sources. I'm really grateful to all the people who take the time to share their knowledge on this subject with others.

In particular:

- Stanford CS231n Convolutional Neural Networks for Visual Recognition
  http://cs231n.stanford.edu/
- Deep Learning Book (GoodFellow, Bengio, Courville)
  http://www.deeplearningbook.org/
- Convolution arithmetic animations
  https://github.com/vdumoulin/conv_arithmetic

- Andrej Karphathy personal blog
  http://karpathy.github.io/
- WildML blog on AI, DL and NLP
  http://www.wildml.com/
- Michael Nielsen Deep Learning online book
  http://neuralnetworksanddeeplearning.com/

References

References i

[1] J. Duchi, E. Hazan, and Y. Singer.
Adaptive subgradient methods for online learning and stochastic optimization.
Journal of Machine Learning Research, 12(Jul):2121–2159, 2011.

[2] D. Kingma and J. Ba.
Adam: A method for stochastic optimization.
arXiv preprint arXiv:1412.6980, 2014.

[3] M. D. Zeiler.
Adadelta: an adaptive learning rate method.
arXiv preprint arXiv:1212.5701, 2012.