Mathematical Foundations of Data Sciences

Gabriel Peyré
CNRS & DMA
École Normale Supérieure
gabriel.peyre@ens.fr
www.gpeyre.com
www.numerical-tours.com

October 7, 2017

Chapter 5

Deep Learning

Before detailing deep architecture and their use, we start this chapter by presenting two essential computational tools that are used to train these model: stochastic optimization methods and automatic differentiation. In practice, they work hand-in-hand to be able to learn painlessly complicated non-linear model on large-scale datasets.

5.1 Stochastic Optimization

We detail stochastic Gradient Descent, with an application to the binary logistic classification problem.

We set the classes indexes to be $\{-1, +1\}$, and remove empty features, normalize X. n is the number of samples, p is the dimensionality of the features,

5.1.1 Batch Gradient Descent (BGD)

We first recall the usual deterministic (batch) gradient descent (BGD) on the problem of supervised logistic classification.

We refer to the dedicated section on logistic classification for background and more details about the derivations of the energy and its gradient.

Logistic classification aims at solving the following convex program

$$\min_w E(w) \overset{\text{def.}}{=} \frac{1}{n} \sum_{i=1}^{n} L(\langle x_i,\, w\rangle, y_i)$$

where the logistic loss reads

$$L(s, y) \overset{\text{def.}}{=} \log(1 + \exp(-sy))$$

We can define energy E and its gradient ∇E and use a vanilla gradient descent

$$w_{\ell+1} = w_\ell - \tau_\ell \nabla E(w_\ell).$$

5.1.2 Stochastic Gradient Descent (SGD)

As any empirical risk minimization procedure, the logistic classification minimization problem can be written as

$$\min_w E(w) = \frac{1}{n} \sum_i E_i(w) \quad \text{where} \quad E_i(w) = L(\langle x_i,\, w\rangle, y_i).$$

For very large n (which could in theory even be infinite, in which case the sum needs to be replaced by an expectation or equivalenty an integral), computing ∇E is prohebitive. It is possible instead to use a

stochastic gradient descent (SGD) scheme

$$w_{\ell+1} = w_\ell - \tau_\ell \nabla E_{i(\ell)}(w_\ell)$$

where, for each iteration index ℓ, $i(\ell)$ is drawn uniformly at random in $\{1, \ldots, n\}$. Note that here

$$\nabla E_i(w) = x_i \nabla L(\langle x_i, w\rangle, y_i) \quad \text{where} \quad \nabla L(u, v) = v \odot \theta(-u)$$

Note that each step of a batch gradient descent has complexity $O(np)$, while a step of SGD only has complexity $O(p)$. SGD is thus advantageous when n is very large, and one cannot afford to do several passes through the data. In some situation, SGD can provide accurate results even with $\ell \ll n$, exploiting redundancy between the samples.

A crucial question is the choice of step size schedule τ_ℓ. It must tends to 0 in order to cancel the noise induced on the gradient by the stochastic sampling. But it should not go too fast to zero in order for the method to keep converging.

A typical schedule that ensures both properties is to have asymptically $\tau_\ell \sim \ell^{-1}$ for $\ell \to +\infty$. We thus propose to use

$$\tau_\ell \stackrel{\text{def.}}{=} \frac{\tau_0}{1 + \ell/\ell_0}$$

where ℓ_0 indicates roughly the number of iterations serving as a "warmup" phase.

One can prove the following convergence result

$$\mathbb{E}(E(w_\ell)) - E(w^\star) = O\left(\frac{1}{\sqrt{\ell}}\right),$$

where $\mathbb{E}$ indicates an expectation with respect to the i.i.d. sampling performed at each iteration.

We can perform the Stochastic gradient descent and display the evolution of the energy $E(w_\ell)$. One can overlay on top (black dashed curve) the convergence of the batch gradient descent, with a carefull scaling of the number of iteration to account for the fact that the complexity of a batch iteration is n times larger.

5.1.3 Stochastic Gradient Descent with Averaging (SGA)

Stochastic gradient descent is slow because of the fast decay of τ_ℓ toward zero.

To improve somehow the convergence speed, it is possible to average the past iterate, i.e. run a "classical" SGD on auxiliary variables $(\tilde{w}_\ell)_\ell$

$$\tilde{w}_{\ell+1} = \tilde{w}_\ell - \tau_\ell \nabla E_{i(\ell)}(\tilde{w}_\ell)$$

and output as estimated weight vector the average

$$w_\ell \stackrel{\text{def.}}{=} \frac{1}{\ell} \sum_{k=1}^{\ell} \tilde{w}_\ell.$$

This defines the Stochastic Gradient Descent with Averaging (SGA) algorithm.

Note that it is possible to avoid explicitly storing all the iterates by simply updating a running average as follow

$$w_{\ell+1} = \frac{1}{\ell}\tilde{w}_\ell + \frac{\ell - 1}{\ell} w_\ell.$$

In this case, a typical choice of decay is rather of the form

$$\tau_\ell \stackrel{\text{def.}}{=} \frac{\tau_0}{1 + \sqrt{\ell/\ell_0}}.$$

Notice that the step size now goes much slower to 0, at rate $\ell^{-1/2}$.

Typically, because the averaging stabilizes the iterates, the choice of (ℓ_0, τ_0) is less important than for SGD.

Bach proves that for logistic classification, it leads to a faster convergence (the constant involved are smaller) than SGD, since on contrast to SGD, SGA is adaptive to the local strong convexity of E.

We can display the evolution of the energy $E(w_\ell)$.

5.1.4 Stochastic Averaged Gradient Descent (SAG)

For problem size n where the dataset (of size $n \times p$) can fully fit into memory, it is possible to further improve the SGA method by bookeeping the previous gradient. This gives rise to the Stochastic Averaged Gradient Descent (SAG) algorithm.

We stored all the previously computed gradient in $(G^i)_{i=1}^n$, which necessitate $n \times p$ memory. The iterates are defined by using a proxy g for the batch gradient, which is progressively enhanced during the iterates.

The algorithm reads

$$h \leftarrow \nabla E_{i(\ell)}(\tilde{w}_\ell),$$

$$g \leftarrow g - G^{i(\ell)} + h,$$

$$G^{i(\ell)} \leftarrow h,$$

$$w_{\ell+1} = w_\ell - \tau g.$$

Note that in contrast to SGD and SGA, this method uses a fixed step size τ. Similarely to the BGD, in order to ensure convergence, the step size τ should be of the order of $1/L$ where L is the Lipschitz constant of E.

This algorithm improves over SGA and BGD since it has a convergence rate of $O(1/\ell$. Furthermore, in the presence of strong convexity (for instance when X is injective for logistic classification), it has a linear convergence rate, i.e.

$$\mathbb{E}(E(w_\ell)) - E(w^\star) = O\left(\rho^\ell\right),$$

for some $0 < \rho < 1$.

Note that this improvement over SGD and SGA is made possible only because SAG explicitly use the fact that n is finite (while SGD and SGA can be extended to infinite n and more general minimization of expectations).

We display the evolution of the energy $E(w_\ell)$.

5.2 Automatic Differentiation

5.3 Deep Discriminative Models

5.4 Deep Generative Models

5.4.1 Density Fitting

Fitting and MLE

Generative Models

5.4.2 Auto-encoders

5.4.3 GANs

Bibliography

[1] Amir Beck. *Introduction to Nonlinear Optimization: Theory, Algorithms, and Applications with MATLAB*. SIAM, 2014.

[2] Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, and Jonathan Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine Learning*, 3(1):1–122, 2011.

[3] Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[4] E. J. Candès. The restricted isometry property and its implications for compressed sensing. *Compte Rendus de l'Académie des Sciences*, Serie I(346):589–592, 2006.

[5] A. Chambolle. An algorithm for total variation minimization and applications. *J. Math. Imaging Vis.*, 20:89–97, 2004.

[6] Antonin Chambolle, Vicent Caselles, Daniel Cremers, Matteo Novaga, and Thomas Pock. An introduction to total variation for image analysis. *Theoretical foundations and numerical methods for sparse recovery*, 9(263-340):227, 2010.

[7] Antonin Chambolle and Thomas Pock. An introduction to continuous optimization for imaging. *Acta Numerica*, 25:161–319, 2016.

[8] Philippe G Ciarlet. Introduction à l'analyse numérique matricielle et à l'optimisation. 1982.

[9] Simon Foucart and Holger Rauhut. *A mathematical introduction to compressive sensing*, volume 1. Birkhäuser Basel, 2013.

[10] Stephane Mallat. *A wavelet tour of signal processing: the sparse way*. Academic press, 2008.

[11] Neal Parikh, Stephen Boyd, et al. Proximal algorithms. *Foundations and Trends® in Optimization*, 1(3):127–239, 2014.

[12] Otmar Scherzer, Markus Grasmair, Harald Grossauer, Markus Haltmeier, Frank Lenzen, and L Sirovich. *Variational methods in imaging*. Springer, 2009.