The Stochastic gradient algorithm

Geir Storvik

February 8, 2020

1 Introduction

This note discusses stochastic gradient algorithms. Although having existed for many years (Robbins and Monro, 1951, reprinted 1985), the algorithm has recived renewed attention due to its importance in fitting deep neural networks. A thourough discussion of the algorithm is given in Bottou et al. (2018) while a broader discussion on stochastic optimization methods in general is given in Spall (2005).

2 The main algorithm

Suppose our aim is to minimize some function $F(\boldsymbol{\theta})$ with respect to $\boldsymbol{\theta}$. Typically, our function of interest will be some *empirical risk*:

$$F(\boldsymbol{\theta}) = \frac{1}{n}\sum_{i=1}^{n} f_i(\boldsymbol{\theta})$$

with many possible options for $f_i(\boldsymbol{\theta})$, eg

$$f_i(\boldsymbol{\theta}) = \begin{cases} (\hat{y}_i - y_i)^2 & \text{Least squares;} \\ I(\hat{y}_i \neq y_i) & \text{Classification error;} \\ -\log f(y_i;\boldsymbol{\theta}) & \text{log-likelihood.} \end{cases}$$

Note that $\hat{y}_i$ will depend on $\boldsymbol{\theta}$ as well as several covariates, although this has not been explicitly included in the notation above. Also some penalized versions of the empirical risk is possible:

$$F(\boldsymbol{\theta}) = \frac{1}{n}\sum_{i=1}^{n} f_i(\boldsymbol{\theta}) + J(\boldsymbol{\theta})$$

Although typically we will consideder some function of empircal risk, we will see that also *expected risk* is relevant to consider:

$$F(\boldsymbol{\theta}) = E[f(\boldsymbol{\theta};\boldsymbol{\varepsilon})].$$

Algorithm 1 Stochastic Gradient algorithm

1: **for** $t = 1, 2, ...$ **do**
2: Simulate the stochastic gradient $g(\boldsymbol{\theta}^t; \boldsymbol{\xi}^t)$
3: Choose a stepsize α^t
4: Update the new value by $\boldsymbol{\theta}^{t+1} \leftarrow \boldsymbol{\theta}^t - \alpha_t \boldsymbol{M}_t^{-1} \hat{g}(\boldsymbol{\theta}^t)$
5: **end for**

Here $\boldsymbol{\varepsilon}$ is some random vector used for generating an observation.

If $F(\cdot)$ is nice and smooth, a necessary requirement for the minimum point $\boldsymbol{\theta}^*$ is then that

$$\boldsymbol{g}(\boldsymbol{\theta}^*) = \frac{\partial}{\partial \boldsymbol{\theta}} F(\boldsymbol{\theta}^*) = \mathbf{0} \tag{1}$$

The ordinary gradient ascient methods is based on the iterative scheme

$$\boldsymbol{\theta}^{t+1} = \boldsymbol{\theta}^t - \boldsymbol{M}_t^{-1} \boldsymbol{g}(\boldsymbol{\theta}^t)$$

where $\boldsymbol{M}_t$ is some positive definite matrix, serving as a scaling matrix. The simplest choise would be $\boldsymbol{M}_t = \boldsymbol{I}$. A main problem in many complex settings is that this gradient might be difficult to compute. The *stochastic gradient* algorithm replaces the gradient by an *estimate* instead:

$$\boldsymbol{\theta}^{t+1} = \boldsymbol{\theta}^t - \alpha_t \boldsymbol{M}_t^{-1} \boldsymbol{Z}(\boldsymbol{\theta}^t; \boldsymbol{\xi}^t) \tag{2}$$

where we describe the estimate of the gradient by $Z(\boldsymbol{\theta}^t; \boldsymbol{\xi}^t)$, emphasizing the stochastic nature through the random vector $\boldsymbol{\xi}^t$. A class of possibilities are given by

$$Z(\boldsymbol{\theta}^t; \boldsymbol{\xi}^t) = \left[\frac{1}{n_t} \sum_{i \in \mathcal{S}_t} \nabla f_i(\boldsymbol{\theta}^t)] \right] \tag{3}$$

where $\mathcal{S}_t \subset \{1, ..., n\}$ and $n_t = |\mathcal{S}_t|$ gives the number of observations to base the estimate of the gradient on. Low values of n_t gives high speed in computation but large variance, typically requiring many iterations. The full procedure is given in Algorithm 1. For maximization, we change sign on the last term in step 4. Note that we now obtain a stochastic sequence $\{\boldsymbol{\theta}^t\}$.

3 Example

Consider a logistic regression setting where

$$\begin{aligned} Y_i &\sim \text{Binomial}(1, p(x_i)), \quad i = 1, ..., n \\ p(x) &= \frac{\exp(\theta_0 + \theta_1 x_i)}{1 + \exp(\theta_0 + \theta_1 x_i)} \end{aligned}$$

Code 1 R code for logistic regression example.

```
#Simulate data
set.seed(34534)
n = 10000
x = rnorm(n)
theta = c(0.1,2)
p = exp(theta[1]+theta[2]*x)/(1+exp(theta[1]+theta[2]*x))
y = rbinom(n,1,prob=p)

#Initialization
b = c(0,1)          #Initial value
N.it = 1000         #Number of iterations
k = 10              #Number of samples for estimating gradient
#SG-loop
for(it in 1:N.it)
{
  i = sample(1:n,k)
  alpha = 10/it
  p.i = exp(b[1]+b[2]*x[i])/(1+exp(b[1]+b[2]*x[i]))
  g = colMeans(cbind(y[i]-p.i,(y[i]-p.i)*x[i]))
  b = b + alpha*g
}
```

with n large. Our objective function is to maximize

$$\begin{aligned} F(\boldsymbol{\theta}) &= \sum_{i=1}^{n} [y_i \log(p_i) + (1-y_i)\log(1-p_i)] \\ &= \sum_{i=1}^{n} [y_i(\theta_0 + \theta_1 x_i) - \log(1+\exp(\theta_0+\theta_1 x_i))] \end{aligned}$$

Defining

$$f_i(\boldsymbol{\theta}) = y_i(\theta_0 + \theta_1 x_i) - \log(1+\exp(\theta_0+\theta_1 x_i))$$

we have

$$\nabla f_i(\boldsymbol{\theta}) = \begin{pmatrix} y_i - \frac{\exp(\theta_0+\theta_1 x_i)}{1+\exp(\theta_0+\theta_1 x_i)} \\ [y_i - \frac{\exp(\theta_0+\theta_1 x_i)}{1+\exp(\theta_0+\theta_1 x_i)}]x_i \end{pmatrix}$$

Code 1 shows R-code for performing stocahstic gradients on this example. Figure 1 shows the convergence of the stockastic gradient algorithms for different choices of n_t in the gradient estimate (3). We see that for $n_t = 1$ the algorithm is slow to convergence, probably due to that the variance of the gradient estimate is too large. For $n_t = 10$ however the convergence is very fast and clearly beats the standard gradient ascent method.

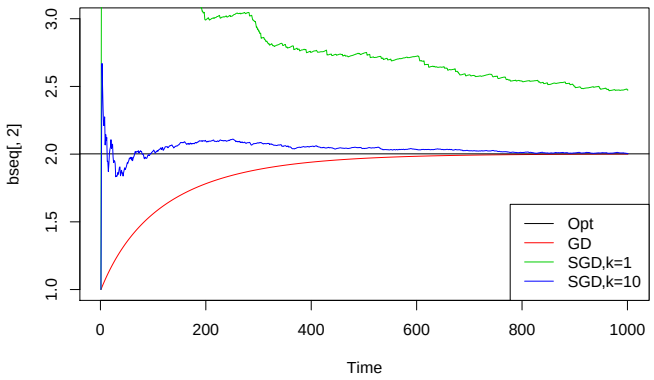

Figure 1: $\{\beta_2\}$ at different iterations for standard gradient ascent and stochastic gradients using $n_t = 1$ and $n_t = 10$. For all curves the x-axis is the seconds needed for obtaining the corresponding β_2 values.

4 Theoretical results

Definition 1. *If* $\lim_{t\to\infty} \boldsymbol{\theta}^t = \boldsymbol{\theta}^*$ in probability*, irrespective of any arbitrary initival value* $\boldsymbol{\theta}^{(0)}$*, we call the procedure* consistent*. Here, convergence in probability means that for any* $\varepsilon > 0$*,*

$$\lim_{t\to\infty} \Pr(|\boldsymbol{\theta}^t - \boldsymbol{\theta}^*| > \varepsilon) = 0$$

We will show that the stochastic gradient algorithm indeed converge in probability (under suitable conditions). We will only give results and proofs in some simplifying settings, following mainly Robbins and Monro (1951) and only when θ is univariate. More general proofs are given in e.g. Bottou et al. (2018).

We will start with the following result:

Lemma 1. *Define*

$$b_t = E[(\theta_t - \theta^*)^2].$$

If $lim_{t\to\infty} b_t = 0$*, then* $\{\theta_t\}$ *is consistent.*

Proof. Defining $p_t(\cdot)$ to be the density of θ_t, we have that

$$\begin{aligned}
\Pr(|\boldsymbol{\theta}^t - \boldsymbol{\theta}^*| > \varepsilon) &= \int_z I[(z-\theta^*)^2 > \varepsilon^2] p_t(z) dz \\
&\leq \int_z \tfrac{(z-\theta^*)^2}{\varepsilon^2} p_t(z) dz \\
&= \frac{1}{\varepsilon^2} \int_z (z-\theta^*)^2 p_t(z) dz = \frac{1}{\varepsilon^2} b_t \to 0
\end{aligned}$$

proving the result. □

The main requirements needed for proving convergence can be divided into requirements on the sequence $\{\alpha_t\}$ and on the function $g(z)$. We will consider the following assumptions on $\{\alpha_t\}$:

$$\alpha_t > 0 \tag{A-1}$$

$$\sum_{t=2}^{\infty} \frac{\alpha_t}{\alpha_1 + \cdots + \alpha_{t-1}} = \infty \tag{A-2}$$

$$\sum_{t=1}^{\infty} \alpha_t^2 < \infty \tag{A-3}$$

Note that (A-2) implies $\sum_{t=1}^{\infty} \alpha_t = \infty$.
Requirements on the function $g(z)$:

$$\exists \delta \geq 0 \text{ such that } g(x) \leq -\delta \text{ for } x < \theta^* \text{ and } g(x) \geq \delta \text{ for } x > \theta^*. \tag{A-4}$$

$$E[Z(\theta;\phi)] = g(\theta) \text{ and } \Pr(|Z(\theta;\phi)| < C) = 1 \tag{A-5}$$

The constraint $|Z(\theta;\xi)| < C$ is included to simplify the proof. More general results are available.

Theorem 1. *Assume* (A-1), (A-3), (A-4) *and* (A-5). *Then the sequence*

$$\theta^{t+1} = \theta^t - \alpha_t Z(\theta^t;\phi^t) \tag{4}$$

will converge in probability.

Remark: Note that this result only gives convergence to some value, not necessarily to the optimal value. This will be proved in the following results were also (A-2) will be assumed. In the following we will simplify the notation somewhat in denoting $Z(\theta^t;\xi^t)$ by Z_t.

Proof. We have

$$\begin{aligned}
b_{t+1} &= E[(\theta^{t+1} - \theta^*)^2] \\
&= E[E[(\theta^{t+1} - \theta^*)^2|\theta^t]] \\
&= E[E[(\theta^t - \alpha_t Z_t - \theta^*)^2|\theta^t]] \\
&= E[(\theta^t - \theta^*)^2 + \alpha_t^2 E[Z_t^2|\theta_t] - 2\alpha_t(\theta^t - \theta^*)E[Z_t|\theta^t]] \\
&= b_t + \alpha_t^2 E[Z_t^2] - 2\alpha_t E[(\theta^t - \theta^*)g(\theta^t)].
\end{aligned}$$

Defining

$$d_t = E[(\theta^t - \theta^*)g(\theta^t)], \quad e_t = E[Z_t^2],$$

we get

$$b_{t+1} - b_t = \alpha_t^2 e_t - 2\alpha_t d_t.$$

which again results in that (by summing the equation above over t)

$$b_{t+1} = b_1 + \sum_{s=1}^{t} \alpha_s e_s - 2\sum_{s=1}^{t} \alpha_s d_s. \tag{5}$$

From (A-4) we have that $d_t \geq 0$. Since $b_{t+1} \geq 0$ it follows from (5) that

$$\sum_{s=1}^{t} \alpha_s d_s = \frac{1}{2}\left[b_1 + \sum_{s=1}^{t} \alpha_s^2 e_s - b_{t+1}\right] \leq \frac{1}{2}\left[b_1 + \sum_{s=1}^{\infty} \alpha_s^2 e_s\right].$$

From $|Z(\theta;\xi)| \leq C$ we have

$$\sum_{s=1}^{\infty} \alpha_s^2 e_s \leq C^2 \sum_{s=1}^{\infty} \alpha_s^2 < \infty$$

proving that both terms in (5), and thereby $\{b_t\}$, converges. □

We now have proven that the series of parameter estimates converges, but not that it converges to the right value. We will first prove the following lemma:

Lemma 2. *Assume* (A-1), (A-3), (A-4) *and* (A-5). *Assume* $\{k_t\}$ *is a sequence of nonnegative constants satisfying*

$$k_t b_t \leq d_t, \quad \sum_{t=1}^{\infty} \alpha_t k_t = \infty \tag{6}$$

Then $\lim_{t\to\infty} b_t = 0$.

Proof. We have that

$$\sum_{t=1}^{\infty} \alpha_t k_t b_t \leq \sum_{t=1}^{\infty} \alpha_t d_t < \infty$$

from the proof of Theorem 1. From the second part of (6) there must be an infinite number of b_t's for which $b_t < \varepsilon$ for any value of ε. Since we have already shown that $\lim_{t\to\infty} b_t$ exists, this shows that the limit has to be zero. □

The following lemma utilize Lemma 2 to construct criteria which can be more easily checked.

Lemma 3. *Assume* (A-1), (A-2), (A-3), (A-4) *and* (A-5). *Assume for some constant* $\delta > 0$ *that*

$$\inf_{z \in [\theta - A_t, \theta + A_t]} \left[\frac{g(z)}{z - \theta} \right] \geq \frac{\delta}{A_t} \text{ for } t > N. \tag{7}$$

where

$$A_t = |\theta^1 - \theta| + C(\alpha_t + \cdots + \alpha_{t-1}). \tag{8}$$

Then $\lim_{t \to \infty} b_t = 0$.

Proof. We have that

$$\theta^t = \theta^1 - \sum_{s=1}^{t-1} \alpha_s Z_s$$

so that

$$\begin{aligned} |\theta^t - \theta^*| =& |\theta^1 - \theta^* - \sum_{s=1}^{t-1} \alpha_s Z_s| \\ \leq& |\theta^1 - \theta^*| + \sum_{s=1}^{t-1} \alpha_s |Z_s| \leq |\theta^1 - \theta^*| + \sum_{s=1}^{t-1} \alpha_s C = A_t \end{aligned}$$

where the second inequality is with probability 1. Now define

$$k_t = \inf_{z \in [\theta^* - A_t, \theta^* + A_t]} \left[\frac{g(z)}{z - \theta^*} \right].$$

From (A-4) we have that $k_t \geq 0$. Further, defining $p_t(\cdot)$ to be the density for Z_t, we have

$$\begin{aligned} k_t b_t =& k_t E[(Z_t - \theta^*)^2] = \int_z k_t (z - \theta^*)^2 p_t(z) dz \\ =& \int_{z \in A_t} k_t (z - \theta^*)^2 p_t(z) dz \leq \int_{z \in A_t} \frac{g(z)}{z - \theta^*} (z - \theta^*)^2 p_t(z) dz \\ =& \int_{z \in A_t} g(z)(z - \theta^*) p_t(z) dz = E[g(Z_t)(Z_t - \theta^*)] = d_t \end{aligned}$$

which shows the first requirement in (6).

By (A-2), $\sum_{t=1}^{\infty} \alpha_t = \infty$ which implies that for t larger than some T

$$2C(\alpha_1 + \cdots + \alpha_{t-1}) = A_t + C(\alpha_1 + \cdots + \alpha_{t-1}) - |\theta^1 - \theta| \geq A_t.$$

This results in that

$$\begin{aligned} \sum_{t=1}^{\infty} \alpha_t k_t \geq& \sum_{t=\min\{N,T\}}^{\infty} \alpha_t k_t \geq \sum_{t=\min\{N,T\}}^{\infty} \frac{\alpha_t \delta}{A_t} \\ \geq& \sum_{t=\min\{N,T\}}^{\infty} \frac{\alpha_t \delta}{2C(\alpha_t + \cdots + \alpha_{t-1})} = \infty \end{aligned}$$

showing the second requirement in (6). □

We are now ready to state the following result:

Theorem 2. *Assume* (A-1), (A-2), (A-3), (A-4) *and* (A-5). *Assume further* $\delta > 0$ *in* (A-4). *Then* $\lim_{t\to\infty} b_t = 0$.

Proof. We have for any $z \in [\theta^* - A_n, \theta^* + A_n]$

$$\frac{g(z)}{z-\theta^*} \geq \frac{\delta}{|z-\theta^*|} \geq \frac{\delta}{A_t}$$

implying that (7) is fulfilled which by Lemma 3 imply the result. □

An alternative to Theorem 2 is the following:

Theorem 3. *Assume* (A-1), (A-2), (A-3) *and* (A-5). *Assume further*

$$g(z) \text{ is nondecreasing}; \tag{9}$$
$$g(\theta^*) = 0; \tag{10}$$
$$g'(\theta^*) > 0. \tag{11}$$

Then $\lim_{t\to\infty} b_t = 0$.

Proof. Since $g'(\theta^*) = \lim_{x\to\theta^*} \frac{g(x)-g(\theta^*)}{x-\theta^*}$, we have that there exists some function $\varepsilon(t)$ with $\lim_{t\to 0} \varepsilon(t) = 0$ and

$$\frac{g(x)}{x-\theta^*} = g'(\theta^*) + \varepsilon(x-\theta^*)$$

giving

$$\varepsilon(x-\theta^*) = \frac{g(x)}{x-\theta^*} - g'(\theta^*) \geq -\frac{1}{2}g'(\theta^*)$$

for $|x-\theta^*| < \delta$ and δ small enough. Thereby

$$\frac{g(x)}{x-\theta^*} \geq \frac{1}{2}g'(\theta^*), \text{ for } |x-\theta^*| \geq \delta.$$

Hense for $\theta^* + \delta \leq \theta^* + A_t$, since $g(z)$ is nondecreasing

$$\frac{g(x)}{x-\theta^*} \geq \frac{g(x+\delta)}{A_t} \geq \frac{\delta g'(\theta^*)}{2A_t}$$

while for $\theta^* - A_t \leq x \leq \theta^* - \delta$

$$\frac{g(x)}{x-\theta^*} \geq \frac{-g(x)}{\theta^*-x} \geq \frac{-g(\theta-\delta)}{A_t} \geq \frac{\delta g'(\theta^*)}{2A_t}.$$

We may assume without generality that $\delta/A_t \leq 1$ giving

$$\frac{g(x)}{x-\theta^*} \geq \frac{\delta g'(\theta^*)}{2A_t} \text{ for } 0 < |x-\theta^*| \leq A_t$$

showing that (7) is fulfilled. □

4.1 SG for expected risk

As mentioned in the introduction, stochastic gradients can also be used to estimate *expected* risk. Assume now data y_t is sampled sequentially and perform the updating

$$\begin{aligned}\boldsymbol{\theta}^{t+1} =&\boldsymbol{\theta}^t - \alpha_t \boldsymbol{M}_t^{-1} \nabla f_{t+1}(\boldsymbol{\theta}^t)\\ f_t(\boldsymbol{\theta}) =&-\log f(y_t;\boldsymbol{\theta})\end{aligned}$$

Then $\boldsymbol{\theta}^t$ will converge towards the minimum of $F(\boldsymbol{\theta})) = E[f(\boldsymbol{\theta},\varepsilon)]$!

5 Neural network

Consider now a neural network model with one latent layer:

$$z_{im} = h(\boldsymbol{\alpha}_m^T \mathbf{x}_i), \qquad m = 1,...,M \tag{12}$$

$$T_i = \beta_0 + \boldsymbol{\beta}^T \mathbf{z}_i \tag{13}$$

$$y_i = g(T_i) + \varepsilon_i. \tag{14}$$

Note that this is equivalent to

$$y_i = f(\boldsymbol{x}_i) + \varepsilon_i$$

where

$$f(\boldsymbol{x}_i) = g(\beta_0 + \sum_{m=1}^{M} \beta_m h(\sum_{j=1}^{p} \alpha_{mj} x_{ij})),$$

showing that it essentially is a (complex) parametric model. For regression, common choices for the $h(\cdot)$ function are

$$\begin{aligned}h(x) =&\frac{1}{1+\exp(-x)} && \text{sigmoid funksjonen;}\\ g(t) =&t && \text{identity function.}\end{aligned}$$

For classification, the model is slightly modified to

$$z_{im} = h(\boldsymbol{\alpha}_m^T \mathbf{x}_i), \qquad m = 1,...,M \tag{15}$$

$$T_{i,k} = \beta_{0,k} + \boldsymbol{\beta}_k^T \mathbf{z}_i \tag{16}$$

$$\Pr(Y_i = k) = p_{i,k} = g_k(\boldsymbol{T}_i) \tag{17}$$

where typically the softmax function is used:

$$g_k(\boldsymbol{T}) = \frac{\exp(T_k)}{\sum_{l=1}^{K} \exp(T_l)}.$$

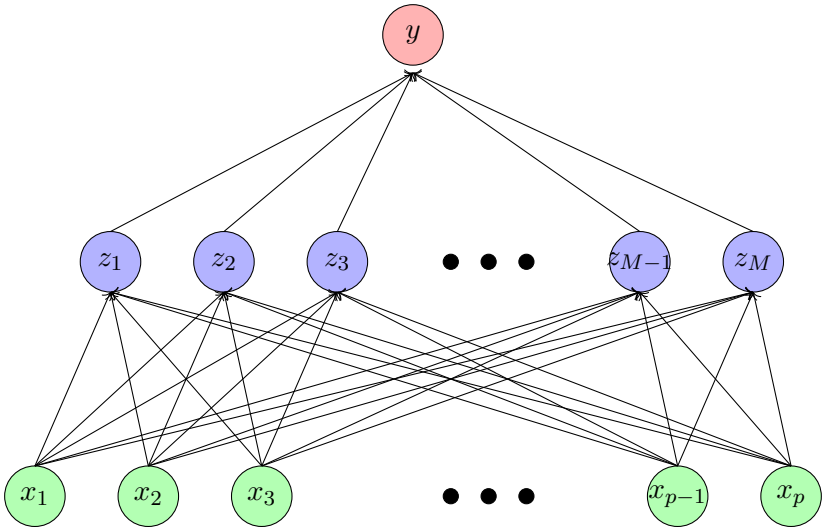

Figure 2: Visualisation of neural network with one hidden layer.

5.1 Estimation of parameters

In general there will be *many* parameters to be estimated. For the regression setting these are

$$\boldsymbol{\theta} = \{\alpha_{mj}, m = 1, ..., M, j = 1, .., p, \beta_m, m = 0, ..., M\}$$

A possible estimation criterion in this case is to minimize

$$Q(\boldsymbol{\theta}) = \sum_{i=1}^{n} \left(y_i - g\left(\beta_0 + \sum_{m=1}^{M} \beta_m h(\sum_{j=1}^{p} \alpha_{mj} x_{ij}) \right) \right)^2 + \lambda_1 \sum_{m=1}^{M} \sum_{j=1}^{p} \alpha_{mj}^2 + \lambda_2 \sum_{m=1}^{M} \beta_m^2$$

where the first term is a fit to data while the two other terms are penalty terms included to avoid overfitting. If a gradient descent algorithm was to be used, derivatives, involving all data points, would have to be eval-

uated. Assuming $h(\cdot)$ and $g(\cdot)$ are smooth and differentiable, we have

$$\begin{aligned}
\frac{\partial Q(\boldsymbol{\theta})}{\beta_{m^*}} =& -2\sum_{i=1}^{n}(y_i - f(\mathbf{x}_i))\frac{\partial f(\mathbf{x}_i)}{\partial \beta_{m^*}} + 2\lambda_2\beta_{m^*} \\
\frac{\partial f(\mathbf{x}_i)}{\partial \beta_{m^*}} =& g'(\beta_0 + \sum_{m=1}^{M}\beta_m h(\sum_{j=1}^{p}\alpha_{mj}x_{ij}))h(\sum_{j=1}^{p}\alpha_{m^*j}x_{ij}) \\
=& g'(T_i)z_{im^*} \\
\frac{\partial Q(\boldsymbol{\theta})}{\partial \alpha_{m^*j^*}} =& -2\sum_{i=1}^{n}(y_i - f(\mathbf{x}_i))\frac{\partial f(\mathbf{x}_i)}{\partial \alpha_{m^*j^*}} + 2\lambda_1\alpha_{m^*j^*} \\
\frac{\partial f(\mathbf{x}_i)}{\partial \alpha_{m^*j^*}} =& g'(\beta_0 + \sum_{m=1}^{M}\beta_m h(\sum_{j=1}^{p}\alpha_{mj}x_{ij}))\beta_{m^*}h'(\sum_{j=1}^{p}\alpha_{m^*j}x_{ij})x_{ij^*} \\
=& g'(T_i)\beta_{m^*}h'(\boldsymbol{\alpha}_{m^*}^T\mathbf{x}_i)x_{ij^*}.
\end{aligned}$$

Note that calculations of these quantities (for a given set of parameter values) requires a *forward* loop for calculation of latent variables and a *backward* loop for calculation of derivatives. The calculation of the derivatives with this procedure is called the *back-propagation* algorithm.

A gradient descent algorithm will interatively update the parameters through

$$\beta_m^{t+1} = \beta_m^t - \gamma_t \left.\frac{\partial Q(\boldsymbol{\theta})}{\beta_m}\right|_{\boldsymbol{\theta}=\boldsymbol{\theta}^t} \tag{18}$$

$$\alpha_{mj}^{t+1} = \alpha_{mj}^t - \gamma_t \left.\frac{\partial Q(\boldsymbol{\theta})}{\alpha_{mj}}\right|_{\boldsymbol{\theta}=\boldsymbol{\theta}^t} \tag{19}$$

until convergence to a (local) optimum is reached.

The tuning parameter γ_t usually called the *learning-rate*. It is typically constant, but can be a sequence converging slowly to zero. One can can increase convergence speed by using second derivatives.

5.2 Stochastic gradient and neural nets

Q and their derivatives require a sum of n terms. One can use a stochastic version by sampling randomly a *subset* of $\{1, ..., n\}$ which in the machine learning literature is called *mini-batching*. In addition to make the calculations much faster, LeCun et al. (2012) state that it also often gives *better solutions* and in a setting where observations are ordered in time also can be used to *track changes*.

In general, optimization will be difficult due to the large number of parameters. Some simplification can be achieved by utilizing some of the structure involved. Assume a regression setting with $g(t) = t$. For a given set of α_{mj}'s, the estimation of the β_m's corresponds to a standard linear

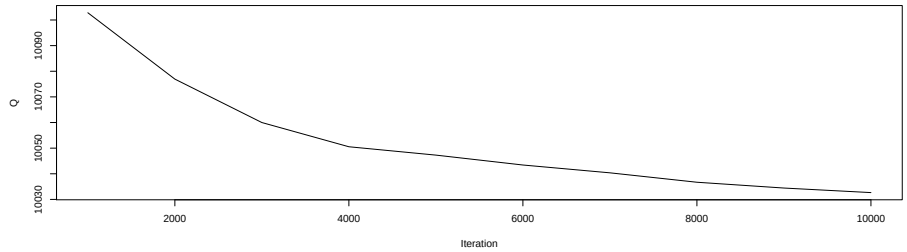

Figure 3: Q at different iterations for the stochastic gradients using $n_t = 5$ based on simulated data with $n = 10\,000$ observations. See the script `Stoch_grad_NN.R` for details.

regression fit with the z_{im}'s as explanatory variables (which are given when the α_{mj}'s are given). This is an example of dimension reduction where the main problem then is reduced to finding optimal α_{mj}'s.

On the course web-page there is a script, `Stoch_grad_NN.R`, which performs fitting to a neural network on simulated data where the algorithm is hard-coded. Figure 3 show that the objective function Q nicely decreases as iterations evolves (only every 1000 iterations are shown).

There exists many routines available for fitting neural network models. In R there are `neuralnetwork` within the `ANN2` package, mlp within the `RSNNS` package, `nnet` within the `nnet` package. Typically there are different trickes applied within the implementations. One such trick is the momentum method in which case the gradient is dynamically updated by assuming the gradient is not changing much from one iteration to another:

$$\begin{aligned} v^{t+1} &= \gamma v^t + \alpha \nabla F(\boldsymbol{\theta}^t) \\ \boldsymbol{\theta}^{t+1} &= \boldsymbol{\theta}^t - v^{t+1} \end{aligned}$$

Another trick is adaptive learning rates

$$\boldsymbol{\theta}^{t+1} = \boldsymbol{\theta}^t - \frac{\alpha}{\sqrt{||\nabla F(\boldsymbol{\theta}^t)||^2 + \varepsilon}} \nabla F(\boldsymbol{\theta}^t)$$

See LeCun et al. (2012) for further details on this.

The script `ANN2_zip.R` at the course web-page shows the use of the ANN2 package on the ZIP code data (images of handwritten digits). This is a classification task and the simple neural network model described above was in this case extended to include 5 layers with 256, 15, 15, 15 and 10 nodes on the different layers. The fitting of the model was somewhat time-consuming. However, the confussion table below shows that the predictions on a test set was quite successfull with an overall

error rate of 8.2%.

		Prediced class									
		0	1	2	3	4	5	6	7	8	9
True class	0	348	0	1	1	1	0	5	1	1	1
	1	0	253	1	1	2	0	4	0	3	0
	2	3	0	169	7	4	2	5	3	5	0
	3	1	0	6	145	1	10	0	0	2	1
	4	2	1	6	1	180	1	3	1	2	3
	5	4	0	0	7	1	141	1	1	4	1
	6	2	0	2	0	2	4	158	0	2	0
	7	0	0	1	1	5	0	0	134	0	6
	8	6	0	5	6	2	1	1	1	142	2
	9	0	0	0	0	2	0	0	2	1	172

6 SG and dependent data

Consider now spatial data colleced at different geographical sites $\boldsymbol{s}_1, ..., \boldsymbol{s}_n$. We will assume a model where the vector $\boldsymbol{Y} = (Y(\boldsymbol{s}_1), ..., Y(\boldsymbol{s}_m))^T$ is multivariate Gaussian with

$$\begin{aligned} E[Y(\boldsymbol{s}_i)] =& \mu \\ \text{Var}[Y(\boldsymbol{s}_i)] =& \sigma^2 + \tau^2 \\ \text{Cov}[Y(\boldsymbol{s}_i), Y(\boldsymbol{s}_j)] =& \sigma^2 r(||\boldsymbol{s}_i - \boldsymbol{s}_j||; \boldsymbol{\phi}) \end{aligned}$$

where $r(d; \boldsymbol{\phi})$ is a function that typically decrease with distance d (it do have some restrictions in that the final covariance matrix must be positive semi-definite). The extra term τ^2 is usually describing observation noise which is assumed to be independent. We can write this in matrix form as

$$\boldsymbol{Y} \sim N(\boldsymbol{\mu}, \boldsymbol{\Sigma})$$

where $\boldsymbol{\mu} = \mu \mathbf{1}$, $\boldsymbol{\Sigma} = \sigma^2 \boldsymbol{R} + \boldsymbol{\tau}^2 \boldsymbol{I}$ and $R_{ij} = r(||\boldsymbol{s}_i - \boldsymbol{s}_j||; \boldsymbol{\phi})$. The vector $\mathbf{1}$ has length n and consist of only ones, the matrix $\boldsymbol{I}$ is the identity.

The observation set $\boldsymbol{Y}$ is a realisation of a process defined continuously in a space $\mathcal{S}$. The log-likelihood with $\boldsymbol{\theta} = (\boldsymbol{\mu}, \sigma^2, \tau^2, \boldsymbol{\phi})$ is given by

$$l(\boldsymbol{\theta}) = -\frac{n}{2} \log 2\pi - \frac{1}{2} \log(|\boldsymbol{\Sigma}|) - \frac{1}{2} (\boldsymbol{y} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\boldsymbol{y} - \boldsymbol{\mu})$$

In general, the computational burden is $O(n^3)$, which can be problematic for large n.

6.1 ML and Kullback-Leibler divergence

The use of stochastic gradients is more complex in this case due to the dependence between the observations. Obtaining an unbiased estimate of the gradient through sub-sampling then become more problematic. A possible approach in this case is then to go one step backwards and rather than using maximum likelihood as criterion for estimation, the criterion now will be to select a parametric model $f_{\boldsymbol{\theta}}(\boldsymbol{y})$ which is as close as possible to the true distribution $g(\boldsymbol{y})$. We then need to define what we mean about close, which we will do through *Kullback-Leibler divergence*:

$$\begin{aligned} KL(f_{\boldsymbol{\theta}}, g) &= \int \log\left(\frac{g(y)}{f_{\boldsymbol{\theta}}(y)}\right) g(y)dy \\ &= \int \log(g(y))g(y)dy - \int \log(f_{\boldsymbol{\theta}}(y))g(y)dy \geq 0 \end{aligned}$$

Minimizing the Kullback-Leibler divergence then becomes equivalent to maximizing $\int \log(f_{\boldsymbol{\theta}}(y))g(y)dy$. A problem however is that $g(y)$ is unknown.

For IID data on can *approximate* $g(y)$ by the empirical distribution $\hat{g}(y) : \Pr(Y = y_i) = \frac{1}{n}$. In that case this results in maximizing

$$\sum_{i=1}^{n} \frac{1}{n} \log(f_{\boldsymbol{\theta}}(y_i)) = \frac{1}{n}\ell(\boldsymbol{\theta})$$

which corresponds to maximizing the log-likelihood. This gives an alternative motivation for the maximum likelihood approach.

For spatial data we have (now also including the geographical positions as part of the observations)

$$\begin{aligned} KL(f_{\boldsymbol{\theta}}, g) &= \int\int \log\left(\frac{g(\boldsymbol{y}|\boldsymbol{s})}{f_{\boldsymbol{\theta}}(\boldsymbol{y}|\boldsymbol{s})}\right) g(\boldsymbol{y}|\boldsymbol{s})g(\boldsymbol{s})d\boldsymbol{y}d\boldsymbol{s} \\ &= \int\int \log(g(\boldsymbol{y}|\boldsymbol{s}))g(\boldsymbol{y}|\boldsymbol{s})g(\boldsymbol{s})d\boldsymbol{y}d\boldsymbol{s} - \\ &\quad \int\int \log(f_{\boldsymbol{\theta}}(\boldsymbol{y}|\boldsymbol{s}))g(\boldsymbol{y}|\boldsymbol{s})g(\boldsymbol{s})d\boldsymbol{y}d\boldsymbol{s} \end{aligned}$$

It is not obvious how to approximate $g(\boldsymbol{y}, \boldsymbol{s}) = g(\boldsymbol{y}|\boldsymbol{s})g(\boldsymbol{s})$ in this case.

We have in this case *one* set of observations $\boldsymbol{y}$. We can approximate $g(\boldsymbol{y}, \boldsymbol{s})$ by giving probability 1 to this observation set. This leads to the maximum (log-)likelihood approach but has the computational burden mentioned earlier. It also has a problem in a poor description of g, leading to that ML estimate may not behave well!

Liang et al. (2013) proposed the following idea: Approximate KL by

$$\widehat{KL}(f_{\boldsymbol{\theta}}, g) = C - \frac{1}{\binom{n}{m}} \sum_{k=1}^{\binom{n}{m}} \log(f_{\boldsymbol{\theta}}(\boldsymbol{y}_k|\boldsymbol{s}_k))$$

where $(\boldsymbol{y}_k, \boldsymbol{s}_k)$ is a subset of $(\boldsymbol{y}, \boldsymbol{s})$ of size $m < n$. Then find $\boldsymbol{\theta}$ as the solution of

$$\frac{\partial}{\partial \boldsymbol{\theta}} \widehat{KL}(f_{\boldsymbol{\theta}}, g) = C - \frac{1}{\binom{n}{m}} \sum_{k=1}^{\binom{n}{m}} H(\boldsymbol{\theta}, \boldsymbol{y}_k, \boldsymbol{s}_k)$$

where

$$H(\boldsymbol{\theta}, \boldsymbol{y}_k, \boldsymbol{s}_k) = \frac{\partial}{\partial \boldsymbol{\theta}} \log(f_{\boldsymbol{\theta}}(\boldsymbol{y}_k | \boldsymbol{s}_k))$$

by the *stochastic gradient algorithm*, which in this case means to approximate the large sum by a (small) set of subsets $(\boldsymbol{y}_k, \boldsymbol{s}_k)$.

6.2 Example

Assume $r(||s_i - s_j||; \boldsymbol{\phi}) = \exp(-(||s_i - s_j||/\phi)$. Then

$$\begin{aligned}
H_\mu(\boldsymbol{\theta}, \boldsymbol{y}_k, \boldsymbol{s}_k) &= \mathbf{1}_m^T \boldsymbol{\Sigma}_k^{-1} (\boldsymbol{y}_k - \mu \mathbf{1}_m) \\
H_{\sigma^2}(\boldsymbol{\theta}, \boldsymbol{y}_k, \boldsymbol{s}_k) &= -\frac{1}{2}\mathrm{tr}(\boldsymbol{\Sigma}_k^{-1}\boldsymbol{R}_k) + \frac{1}{2}(\boldsymbol{y}_k - \mu \mathbf{1}_m)^T \boldsymbol{\Sigma}_k^{-1} \boldsymbol{R}_k \boldsymbol{\Sigma}_k^{-1} (\boldsymbol{y}_k - \mu \mathbf{1}_m) \\
H_{\tau^2}(\boldsymbol{\theta}, \boldsymbol{y}_k, \boldsymbol{s}_k) &= -\frac{1}{2}\mathrm{tr}(\boldsymbol{\Sigma}_k^{-1}) + \frac{1}{2}(\boldsymbol{y}_k - \mu \mathbf{1}_m)^T \boldsymbol{\Sigma}_k^{-2} (\boldsymbol{y}_k - \mu \mathbf{1}_m) \\
H_\phi(\boldsymbol{\theta}, \boldsymbol{y}_k, \boldsymbol{s}_k) &= -\frac{1}{2}\mathrm{tr}(\boldsymbol{\Sigma}_k^{-1}\frac{d\boldsymbol{R}_k}{d\phi}) + \frac{1}{2}(\boldsymbol{y}_k - \mu \mathbf{1}_m)^T \boldsymbol{\Sigma}_k^{-1} \frac{d\boldsymbol{R}_k}{d\phi} \boldsymbol{\Sigma}_k^{-1} (\boldsymbol{y}_k - \mu \mathbf{1}_m) \\
(\boldsymbol{\Sigma}_k)_{i,j} &= \mathsf{cov}(Y(\boldsymbol{s}_{k,i}) - Y(\boldsymbol{s}_{k,j}) = \tau^2 I(i=j) + \sigma^2 \exp(-(||\boldsymbol{s}_{k,i} - \boldsymbol{s}_{k,j}||/\phi) \\
(\boldsymbol{R}_k)_{i,j} &= \exp(-(||\boldsymbol{s}_{k,i} - \boldsymbol{s}_{k,j}||/\phi) \\
\frac{d(\boldsymbol{R}_k)_{i,j}}{d\phi} &= ||\boldsymbol{s}_{k,i} - \boldsymbol{s}_{k,j}||/\phi^2 \cdot \exp(-(||\boldsymbol{s}_{k,i} - \boldsymbol{s}_{k,j}||/\phi)
\end{aligned}$$

The R script `Geostat_SG.R` contains an example of the use of stochastic gradients on this example using simulated data. Subsample sizes of $m = 5, 10$ and 20 was used. Figure 4 shows the log-likelihood values obtained for 10 repetitions of the algorithm. For too small values of m, the variability seems to be too high, while $m = 20$ give quite stable results.

References

L. Bottou, F. E. Curtis, and J. Nocedal. Optimization methods for large-scale machine learning. *SIAM Review*, 60(2):223–311, 2018.

Y. A. LeCun, L. Bottou, G. B. Orr, and K.-R. Müller. Efficient backprop. In *Neural networks: Tricks of the trade*, pages 9–48. Springer, 2012.

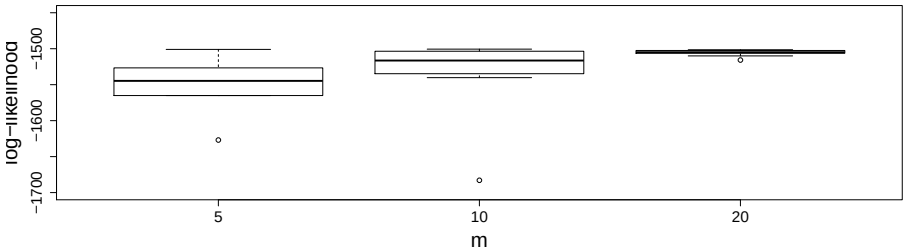

Figure 4: Boxplot of values of ℓ obtained for $m = 5, 10$ and 20 through 10 repetitions of the stochastic gradient algorithm. For $m = 5$ a very small value of -9757.242 obtained at one of the runs have been suppressed.

F. Liang, Y. Cheng, Q. Song, J. Park, and P. Yang. A resampling-based stochastic approximation method for analysis of large geostatistical data. *Journal of the American Statistical Association*, 108(501):325–339, 2013.

H. Robbins and S. Monro. A Stochastic Approximation Method. *Annals Math. Statistics*, 22:400–407, 1951.

J. C. Spall. *Introduction to stochastic search and optimization: estimation, simulation, and control*, volume 65. John Wiley & Sons, 2005.