Optimization for Data Science

F. Rinaldi[1]

[1]  Università degli Studi di Padova | Dipartimento Matematica

Padova
2020

Outline

Optimization for Data Science

Limits of Stochastic Gradient

Remark

In expectation, the stochastic gradient is identical to the classic gradient

$$\mathbb{E}[\nabla f_i(x)] = \sum_{j=1}^{m} P(i=j) \cdot \nabla f_j(x) = \sum_{j=1}^{m} \frac{1}{m} \cdot \nabla f_j(x) = \frac{1}{m} \sum_{j=1}^{m} \nabla f_j(x).$$

Notice that we exploit definition of expectation and independence of samples.

- ISSUE: Deviation from the mean value in some of the instances might be very high (large variance).
- This is the reason why we get a sublinear rate when using the SG method.
- Any strategy to overcome this issue?

Estimation of $\Theta = \mathbb{E}[X]$

- An estimator is a rule for calculating an estimate of a given parameter based on observed data.
- It can be *unbiased* in case the expected value gives exactly the parameter, *biased* otherwise.
- Consider random variable Y (highly correlated with X).
- Assume that calculating $\mathbb{E}[Y]$ is easy.

Point Estimator of $\Theta = \mathbb{E}[X]$

- Let us consider the following point estimator with $\gamma \in [0, 1]$:

$$\hat{\Theta}_\gamma = \gamma(X - Y) + \mathbb{E}[Y]. \tag{1}$$

Comments on the Estimate

Expectation and Variance

- The expectation and variance are:

$$\mathbb{E}[\hat{\Theta}_\gamma] = \gamma\mathbb{E}[X] + (1-\gamma)\mathbb{E}[Y];$$

$$var[\hat{\Theta}_\gamma] = \gamma^2\left(var[X] + var[Y] - 2cov[X,Y]\right).$$

- Unbiased estimator when $\gamma = 1$ (i.e., $\mathbb{E}[\hat{\Theta}_\gamma] = \mathbb{E}[X]$).
- Zero variance estimator when $\gamma = 0$ (in this case the bias is big since the estimator is the constant $E[Y]$).

Comments on the Estimate II

- Most important part here is that $var[\hat{\Theta}_\gamma] < var[X]$ when $cov(X, Y)$ is large.
- Estimator has smaller variance than X.
- This is a **general method for reducing variance**.
- Method is included in many hybrid gradient methods recently proposed in the literature (like, e.g., SAG, SAGA, SVRG).
- Before analyzing those techniques, we set

$$\Theta = \mathbb{E}[\nabla f_i(x)]$$

and

$$X = \nabla f_i(x).$$

SAG Approach

SAG

At each iteration, the method picks at random i_k and sets

$$g_k^i = \begin{cases} \nabla f_{i_k}(x_k) & \text{if } i = i_k \\ g_{k-1}^i & \text{otherwise.} \end{cases}$$

Main iteration is then defined the following way:

$$x_{k+1} = x_k - \alpha_k \frac{1}{m} \sum_{i=1}^{m} g_k^i.$$

SAG Approach

- In this case we can rewrite the iterate as

$$x_{k+1} = x_k - \alpha_k \left[\frac{1}{m} (\nabla f_{i_k}(x_k) - g_{k-1}^{i_k}) + \frac{1}{m} \sum_{i=1}^{m} g_{k-1}^{i} \right].$$

- Espression in squared bracket with $\gamma = 1/m$ and $Y = g_{k-1}^{i}$ is same as Equation (1).
- PROs: Method guarantees a linear convergence rate.
- CONs: Cost in terms of memory is very high (i.e., $\mathcal{O}(m)$) due to the fact that we need to store terms g_{k-1}^{i}.

Algorithmic Scheme

Algorithm 1 `SAG method`

1 Choose a point $x_1 \in \mathbb{R}^n$ and set $g_0^i = \nabla f_i(x_1)$, $i = 1, \ldots, m$
2 For $k = 1, \ldots$
3 If x_k satisfies some specific condition, then STOP
4 Pick at random i_k and set

$$g_k^i = \begin{cases} \nabla f_{i_k}(x_k) & \text{if } i = i_k \\ g_{k-1}^i & \text{otherwise.} \end{cases}$$

5 Set $x_{k+1} = x_k - \alpha_k \frac{1}{m} \sum_{i=1}^m g_k^i$, with $\alpha_k > 0$
a suitably chosen stepsize
6 End for

SAGA Approach

SAGA

It is very similar to SAG. The main iteration of this stochastic method is

$$x_{k+1} = x_k - \alpha_k \left[\nabla f_{i_k}(x_k) - g_{k-1}^{i_k} + \frac{1}{m} \sum_{i=1}^{m} g_{k-1}^{i} \right],$$

with i_k randomly chosen index.

- Espression in squared bracket with $\gamma = 1$ and $Y = g_{k-1}^{i}$ is same as Equation (1).
- PROs: Method guarantees a linear convergence rate with better constants than SAG.
- CONs: Cost in terms of memory is very high (i.e., $\mathcal{O}(m)$) due to the fact that we need to store terms g_{k-1}^{i}.

SVRG Approach

SVRG

At each epoch s the method performs l steps of the form

$$x_{k+1} = x_k - \alpha_k \left[\nabla f_{i_k}(x_k) - \nabla f_{i_k}(\tilde{x}) + \frac{1}{m} \sum_{i=1}^{m} \nabla f_i(\tilde{x}) \right],$$

with i_k randomly chosen index and $\tilde{x}$ the last iterate coming from the previous epoch $s-1$.

- Espression in squared bracket with $\gamma = 1$ and $Y = \nabla f_i(\tilde{x})$ is same as Equation (1).

Comments

PROs

- Method guarantees a linear convergence rate.
- Does not require gradient storage (i.e., big savings in terms of memory).
- Theoretical analysis is easy and very intuitive.

Logistic Regression

- We consider a binary classification problem here.
- It is possible to use various classes of *learning machines* for constructing an approximation f of a unknown functional dependency.
- Training set:

$$T = \{(x^i,\ y^i) \mid x^i \in X,\ y^i \in Y \ \text{ and } i = 1, \ldots, m\}\ .$$

- Based on the input feature vectors and certain type of function we adopt, the output can be predicted based on some specific prediction rules.
- For binary classification problem, prediction rules have form:

$$y = \begin{cases} 1 & \text{if } f(x; w) \geq 0 \\ -1 & \text{if } f(x; w) < 0. \end{cases}$$

Binary Classification

- An error term $E^i(w)$ related to a given sample (x^i, y^i) can be used to denote whether the output is successfully predicted or not:

$$E^i(w) = \begin{cases} 1 & \text{if} \quad y^i \cdot f(x^i; w) < 0 \\ 0 & \text{if} \quad y^i \cdot f(x^i; w) \geq 0. \end{cases}$$

- We define the *0-1 loss function*:

$$\ell(z) = \begin{cases} 1 & \text{if } z < 0 \\ 0 & \text{if } z \geq 0. \end{cases}$$

- Error term in our case is:

$$E^i(w) = \ell(y^i \cdot f(x^i; w)).$$

Optimization Problem

Goal

Find the best parameter w that minimizes the total error in the training set. Thus the optimization problem related to the classification model becomes

$$\min_{w \in \mathbb{R}^n} \sum_{i=1}^{m} \ell(y^i \cdot f(x^i; w))$$

which is usually called **0-1 loss minimization problem**.

- This is a non-convex optimization problem, which is very hard to solve!

How to Get an Easy Problem

Idea

Instead of solving the 0-1 loss minimization problem, use some convex function and solve problems that give convex upper bounds of the 0-1 loss function.

We list here 4 convex loss functions that are commonly used in the literature:

- hinge loss function: $\ell(v) = \max\{1 - v, 0\}$;
- exponential loss function $\ell(v) = \exp(-v)$;
- modified least square loss function $\ell(v) = \max\{1 - v, 0\}^2$;
- logistic loss function $\ell(v) = \log(1 + \exp(-v))$.

Logistic Regression and Friends

Logistic Regression

When we consider the logistic loss, we end up solving a *logistic regression* problem. The problem is then described as follows:

$$\min_{w\in\mathbb{R}^n} \sum_{i=1}^{m} \log\left(1+\exp\left(-y^i w^\top x^i\right)\right),$$

where we use a linear function as classifier, i.e. $f(x;w) = w^\top x$.

Regularized Logistic Regression

Sometimes, a regularized version of the problem is considered in practice:

$$\min_{w\in\mathbb{R}^n} h(w) = \sum_{i=1}^{m} \log\left(1+\exp\left(-y^i w^\top x^i\right)\right) + \frac{\lambda}{2}\|w\|^2,$$

with $\lambda > 0$ regularization parameter.

Why Regularization?

Goal

Prevent that our model picks up "peculiarities", "noise" coming from the given data.

- Usually considered to improve the *generalization performance*, i.e., the performance on new, unseen data.
- We can think of regularization as a penalty against complexity.
- Increasing the regularization parameter penalizes "large" weight coefficients.
- Again, we don't want the model to memorize the training dataset, we want a model that generalizes well to new, unseen data.
- We can think of regularization as adding (or increasing the) bias if our model suffers from (high) variance (i.e., it overfits the training data).
- Too much bias will result in underfitting!

Problem Analysis

- We start by defining the function

$$g(z) = \frac{1}{1+e^{-z}}.$$

- Notice that

$$1 - g(z) = \frac{e^{-z}}{1+e^{-z}}.$$

and

$$\frac{\partial g(z)}{\partial z} = g(z)(1-g(z)).$$

- The j-th component of the gradient for the function $h(w)$ is

$$\frac{\partial h(w)}{\partial w_j} = \sum_{i=1}^{m} \frac{-y^i x_j^i e^{-y^i w^\top x^i}}{1+e^{-y^i w^\top x^i}} + \lambda w_j = -\sum_{i=1}^{m} y^i x_j^i (1 - g(y^i w^\top x^i)) + \lambda w_j.$$

- Hence, the full gradient is

$$\nabla h(w) = \sum_{i=1}^{m} \frac{-y^i x^i e^{-y^i w^\top x^i}}{1+e^{-y^i w^\top x^i}} + \lambda w = -\sum_{i=1}^{m} y^i x^i (1 - g(y^i w^\top x^i)) + \lambda w.$$

Problem Analysis II

- The generic element of the Hessian matrix related to the function $h(w)$ is

$$\frac{\partial^2 h(w)}{\partial w_j w_k} = \sum_{i=1}^{m} (y^i)^2 x_j^i x_k^i g(y^i w^\top x^i)(1 - g(y^i w^\top x^i)), \text{ with } j \neq k,$$

and

$$\frac{\partial^2 h(w)}{\partial^2 w_j} = \sum_{i=1}^{m} (y^i)^2 (x_j^i)^2 g(y^i w^\top x^i)(1 - g(y^i w^\top x^i)) + \lambda.$$

- We have

$$\nabla^2 h(w) = \sum_{i=1}^{m} (y^i)^2 x^i g(y^i w^\top x^i)(1-g(y^i w^\top x^i))(x^i)^\top + \lambda I = X^\top P X + \lambda I,$$

where P is the diagonal matrix with element

$$P_{ii} = g(y^i w^\top x^i)(1 - g(y^i w^\top x^i)), \; i = 1, \ldots, m$$

and $X^\top = [x^1 \ldots x^m]$.

Remarks

- We note that $0 < P_{ii} \leq \delta$ for all i, where $\delta = \max_i P_{ii}$.
- It is easy to see that the function $h(w)$ is strongly convex.
- It can be shown that L is less or equal than the biggest eigenvalue of the Hessian calculated for any w.
- We have

$$L \leq \delta\beta + \lambda,$$

where β is the maximum eigenvalue of the matrix $X^\top X$.

Boosting (AdaBoost)

- Idea: creating a highly accurate predictor by combining many relatively weak and inaccurate predictors.
- AdaBoost algorithm of Freund and Schapire (1997) first practical boosting algorithm.
- AdaBoost is the most widely used and studied method, with applications in numerous fields.

Problem Definition

Boosting Problem

Given l classifiers and a set of m datapoints

$$TS = \{(x^i, y^i),\ x^i \in R^n,\ y^i \in \{-1, 1\},\ i = 1, \ldots, m\}$$

improve quality of prediction over TS by **linear combination** of the available classifiers.

- Each classifier j gives value $a_{ij} \in [-1, 1]$ for a specific sample i.
- Build a classification matrix $A \in \mathbb{R}^{m \times l}$ having the form $A = [a^1 \ldots a^l]$.
- Consider a loss function $L(\mathbf{c}, \mathbf{y})$ that measures the cost of making some prediction $\mathbf{c} \in \mathbb{R}^m$ for the given classification $\mathbf{y} \in \mathbb{R}^m$ (notice that $\mathbf{y}^\top = [y^1 \ldots y^m]$).
- Ideal loss function is the 0-1 loss:

$$L(\mathbf{c}, \mathbf{y}) = \frac{1}{m} \sum_{i=1}^{m} \ell(c^i y^i).$$

- Ideal loss replaced with a smooth convex approximation of the original one

$$L(\mathbf{c}, \mathbf{y}) = \frac{1}{m} \sum_{i=1}^{m} e^{-\rho c^i y^i},$$

with $\rho > 0$ tunable parameter.

Boosting (AdaBoost)

Goal

Finding the best possible classification $\mathbf{c}$ by means of a linear combination of the given predictors,

$$\mathbf{c} = Aw,$$

i.e. $L(Aw, \mathbf{y})$ needs to be as small as possible.

- We then write the problem as follows:

$$\min_{w \in R^l} \sum_{i=1}^{m} \frac{e^{-\rho(Aw)^i y^i}}{m}. \tag{2}$$

Comments

- It is easy to see that the function we deal with is convex.
- Some form of regularization can also be used in this context.
- Possible to solve the problem by using one of the algorithms we described.
- Possible to prove that the original Adaboost method is nothing but coordinate descent applied to Problem (2):

$$\min_{w \in R^l} \sum_{i=1}^{m} \frac{e^{-\rho(Aw)^i y^i}}{m}.$$