Indian Institute of Technology Kanpur
CS777 Topics in Learning Theory

Scribe: Amur Ghose
Instructor: Purushottam Kar
Date: March 28, 2018

LECTURE
21

Alternating Minimization Part 3

1 Introduction

Over the past two lectures of Alt-min techniques, we have observed that non-convex functions can be "convexified" via a variety of techniques. This lecture continues on our theme of looking at ad-hoc heuristics along this angle and propping them up on rigorous ground. The generic problem - and the approach - we had been considering could be summarized as :

- For an argmin problem of $f(x)$ with $x \in \mathcal{X}$, create a "proxy" function $Prox$, which in turn creates a function g mapping from $\mathcal{X}$ to $\mathbb{R}$.
- For all timesteps t, create $g^{t+1} \equiv Prox(x^t)$.
- Assign x^{t+1} as the argmin of $g^{t+1}(x)$ over $x \in \mathcal{X}$.

These areas will briefly touch on the varied topics of Pseudo-linear programming, Iteratively reweighed least squares (IRLS), EM methods and Difference of convex programming (DC programming). Before we venture into these problems, we define an important concept of sub-level sets.

2 Sub-level sets

Let f map from $\mathcal{X}$ to $\mathbb{R}$. The α-sub-level set generated by f is denoted $S_f(\alpha)$, and

$$S_f(\alpha) \equiv x \in \mathcal{X} : f(x) \leq \alpha$$

Note that the concept of a **level set** and contours are defined mathematically quite analogously to their common use in geography and are intuitively similar. Trivially, we can observe that :

- A linear function defines a half-space as its sub-level set (yet, observe that the converse does not hold)
- A convex function generates a convex sub-level set (but the converse yet again fails to hold !)

In fact, the observation that the converse does **not** hold for convex sub-level spaces allows us to segue into a discussion that briefly motivates itself thus : We **know** that linear functions and convex functions are very easily optimized. Does it then follow that functions which generate sub-level spaces similar to these functions are similarly easy to optimize, despite not being convex or linear themselves ? We now segue ourselves on to the topic of linear and pseudo-linear programs off this pivot.

3 Linear Programming

Linear programming is an extensively well-studied and analyzed topic and it is beyond the scope of this lecture, or even this course, to cover it in a depth to do it justice. A canonical LP is defined as follows :

$$\max c^T x, Ax \leq b, x \geq 0$$

It should be noted that these are **index-wise inequalities** : for example, the last inequality implies that $x_i \geq 0$ for all i. However, for this course, we define the LP in an unorthodox manner :

$$\min a^T x, x \in S$$

Where S is a set of interest, and hopefully convex. A LP-oracle will return us the $x \in S$ for the LP-problem. Now, it can be seen that the LP, having a linear objective function, defines a half-space as its sub-level set.

Consider the objective function of form :

$$f_o(x) = \frac{a^T x + c}{b^T x + d}$$

It may be verified that the above objective function defines half-spaces as sub-level sets. We define a **pseudo-linear program** as a LP but with its objective function replaced by the ratio of two affine functions as above. For simplicity, we will now consider c, d both to be zero for the remainder of the lecture.

4 Solving the PLP : A Frank Wolfe equivalence

Assume that for this section :

- c, d are set to zero
- $\alpha \leq b^T x \leq \beta$ within $x \in \mathcal{X}$
- The existence of an efficient polytime **LP-oracle**

By a LP-oracle, we mean a blackbox that returns to us the argmax over $x \in \mathcal{X}$ for the LP with objective function $\langle c, x \rangle$ for any c. Further, since we are defining an algorithm, we will enforce polytime. Now, we have that :

$$f(x) = \frac{a^T x}{b^T x} \geq \lambda \Leftrightarrow (a - \lambda b)^T x \geq 0$$

Define $V_\lambda(x) = (a - \lambda b)^T x$ and let $\lambda_t = f(x_t)$ at an iteration timestep t. Then, define the PLP solver algorithm as :

- Generate $\lambda_t = f(x_t)$
- Generate $V_{\lambda_t}(x)$ using λ_t
- Define x_{t+1} as the argmax of $V_{\lambda_t}(x)$ for $x \in \mathcal{X}$

It may be noted that is analogous to the proxy and g functions defined in the introduction. Since $f(x^*)$ is optimal and λ_t are function values, we readily have $\lambda_t \leq f(x^*) \equiv f^*$ for any t.

Further, note that :

$$V_{\lambda_t}(x_t) = 0 \Leftrightarrow V_{\lambda_t}(x_{t+1}) = e_{t+1} \geq 0$$

Now, we will try to relate the values of $f(x_{t+1})$ and $f(x_t)$. Observe that :

$$V_\lambda(x) = c \Leftrightarrow a^T x = \lambda b^T x + c$$

However, we have that $b^T x \leq \beta$, which implies $\lambda b^T x + c \geq \lambda b^T x + \frac{cb^T x}{\beta}$, which yields :

$$V_\lambda(x) = c \Leftrightarrow f(x) \geq (\lambda + \frac{c}{\beta})$$

Since from the other direction $b^T x \geq \alpha$, we also have that

$$V_\lambda(x) = c \Leftrightarrow f(x) \leq (\lambda + \frac{c}{\alpha})$$

Now since $V_{\lambda_t}(x_{t+1}) = e_{t+1}$ we are ready to set a chain of inequalities in motion. Specifically, note that by putting in $e_{t+1} = c$ and $x_{t+1} = x$ in the above, we get :

$$f(x_t + 1) \leq \lambda_t + \frac{e_{t+1}}{\alpha} \Leftrightarrow e_{t+1} \leq (f^* - \lambda_t)\alpha$$

Where in the last step we've used that $f^* \geq f(x_{t+1})$. Plug this in to :

$$f(x_t + 1) \geq \lambda_t + \frac{e_{t+1}}{\beta}$$

To get :

$$f(x_t + 1) \geq \lambda_t + \frac{\alpha}{\beta}(f^* - \lambda_t)$$

Now, use the potential function $\Phi_t = f^* - f_t$. Using the above, we obtain that :

$$\Phi_{t+1} \leq \Phi_t - \frac{\alpha}{\beta}\Phi_t = (1 - \frac{\alpha}{\beta})\Phi_t$$

Which suggests that Φ_t varies as $\leq exp(-\frac{\tau}{\kappa})$ where $\kappa = \frac{\beta}{\alpha}$ is the usual condition number. Note that if we consider the objective function $\frac{a^T x}{b^T x}$ then its gradient is $\frac{a - \lambda b}{b^T \lambda}$, which resolves this algorithm as a variant of Frank-Wolfe that moves "entirely" , i.e. sets $\eta = 1$ throughout. Thus, this algorithm **does not find interior solutions** - given a set, it "bounces" around the boundary.

5 Optimizing the F-measure

The F-measure is defined as $\frac{2PR}{P+R}$, where P, R respectively stand for precision and recall. Plugging in the definitions of these quantities - precision being the fraction of true positives among true and false positives and recall being the one of true positives to true positives and false negatives, we simplify it as :

$$\frac{2TP}{P + N + TP - TN}$$

where P, N stand for positive and negative, and T stands for true, i.e. TP maps to true positive. Map a point in $\mathbb{R}^2$ to every classifying weight vector w as $(TP(w), TN(w))$. The sub-level set of the F-measure defines, with $P + N = S$:

$$\frac{2TP}{S + TP - TN} \geq \lambda \Leftrightarrow (2 - \lambda)TP + \lambda TN \geq \lambda S$$

Thus, using the sub-level set argument, we may cast the F-measure maximizing problem as maximizing

$$f_o(w) = (2 - \lambda)TP(w) + \lambda TN(w)$$

Note that the coefficients are non-negative and it resolves to a weighted-regression problem. (λ, after all, is the HM of two quantities in $[0, 1]$ and thus ≤ 1 trivially)

6 Exponential models

A surprisingly large variety of PDFs that occur in linear modeling can be cast in the exponential form :

$$P(y|\theta) = h(y)exp(\eta(\theta)^T \tau(y) - A(\theta))$$

Which becomes **canonical** under the case of $\eta(\theta) = \theta$. However, since we are concerned with modeling $P(y|x, \theta)$ under GLMs, we will modify this somewhat and define the canonical as $\eta(\theta) = \theta^T x$ and leave the x implicit. That is, in our analysis, $\theta^T x$ implicitly replaces θ. So long as the notation is consistent, this does not matter. For more on exponential forms and GLMs, the reader is advised to peruse (link in text) the PML-2017 slides. Rai (2017).

A few examples of this family include :

- The univariate normal distribution of mean μ and std. dev σ : $A = \frac{(\theta^T x)^2}{2\sigma^2}, \tau(y) = \frac{y}{\sigma}, h(y) = \frac{exp(-\frac{y^2}{2\sigma^2})}{\sqrt{2\pi\sigma^2}}$
- Logistic (binomial / logit) : with $\tau(y) = y/2, h(y) = 1$ we recover that $\theta^T x = \frac{\mu}{1-\mu}$, where μ indicates the bias.

6.1 Relating IRLS to the model

Consider solving the NLL problem over an exponential model. The equivalent problem resolves to :

$$\min \sum_{i=1}^{n} [A((\theta^T x^i)) - (\theta^T x^i)\tau(y^i)]$$

Where the minimization is over θ. Viewed as a gradient descent process, this optimization problem resolves (after using the Hessian) to a IRLS problem i.e. a regression problem where at each time-step t, the weights β are re-adjusted. Specifically,

Exercise 21.1. Demonstrate that the Gradient descent process on θ resolves to solving an argmin problem of the form with ${s_i}^t \geq 0$:

$$\sum_{i=1}^{n} {s_i}^t (\theta^T x^i - {z_i}^t)^2$$

This problem is actually better reformulated in matrix form : the above variables z_i resolve as $X\theta^t + (W^t)^{-1}(y - \theta^T x)$, where X is the overall input matrix of all x_i, and W the diagonal matrix formed by taking the diagonal entries from the derivative matrix. This becomes possible since in this case, the Hessian factorizes itself in the form $-X^T W X$. Thus, each z_i term above is a "residual" term multiplied by a matrix. The reader is advised (link in text) to look at the reference Xing et al. (2014), which contains a detailed exposition.

References

Piyush Rai. Probabilistic ML, Lecture 7. Technical report, CSE, IIT Kanpur, 2017.

Eric P. Xing, Alnur Ali, and Yipei Wang. Probabilistic Graphical Models, Lecture 6. Technical report, Carnegie Mellon University, 2014.