COS 597G: Toward Theoretical Understanding of Deep Learning **Fall 2018**

Lecture 5: Adaptive Regularization

Lecturer: Sanjeev Arora *Scribe: Ethan Tseng, DiJia(Andy) Su*

This lecture is about adaptive regularization, specifically, Adagrad (Duchi, Hazan, and Singer 2011), the method that initiated this stream of work. We give a clear statement of the underlying design principle, which is slightly hidden under the covers in the usual "regret framework."

Let $f : \mathbb{R}^n \to \mathbb{R}$ be the objective function of the model parameters $x \in \mathbb{R}^n$. Recall that gradient descent (GD) explores the state space by taking small steps along $-(\nabla f(x))$. The analysis often uses a second order Taylor series expansion.

$$f(x + \Delta x) = f(x) + (\nabla f(x))^T \Delta x + \frac{1}{2}(\Delta x)^T (\nabla^2 f(x))(\Delta x)$$

Recall that $\nabla f(x)$ is the gradient vector of f and $\nabla^2 f(x)$ is the Hessian matrix of f.

5.1 Pre-conditioner for gradient descent

The analysis of GD often relies upon properties of the Hessian. The simplest is smoothness.

1. L-smoothness: $\|\nabla f(x) - \nabla f(x')\| \leq L\|x - x'\|$
2. Which is equivalent to a bound on the absolute value of the eigenvalues of Hessian: $-LI \preceq (\nabla^2 f(x)) \preceq LI$.

From simple integration it can be shown that if f is L-smooth then $f(x + \Delta x) \leq f(x) + (\nabla f(x))^T \Delta x + \frac{1}{2} L||\Delta x||^2$.

We had seen the Descent Lemma, which shows GD makes progress if learning rate is set appropriately using the smoothness. Thus even crude 2nd order information can help gradient descent, for instance to tune its learning rate.

Lemma 1 (Descent Lemma)**.** *If $\eta < 1/2L$ then*

$$f(x + \Delta x) - f(x) \leq (\nabla f(x))^T \Delta x + \frac{1}{2} L\|\Delta x\|^2 = -\eta\|\nabla f(x)\|^2 + \frac{L}{2}\eta^2\|\nabla f(x)\|^2 = -\frac{1}{2}\eta\|\nabla f(x)\|^2$$

A more sophisticated use of 2nd order info involves pre-conditioners for the gradient:

$$x^{(t+1)} \leftarrow x^{(t)} - \eta H^{-1} \nabla f(x^{(t)})$$

where H is the pre-conditioner matrix. It can be seen as shaping the gradient; varying the emphasis in different directions. The following examples shows that by using full 2nd order information the pre-conditioner can be set appropriately to greatly speed up optimization.

> **Example:** If $f(x)$ is quadratic then $f(x) = c + a^T x + x^T B x$, where $c \in \mathbb{R}, a \in \mathbb{R}^n, B \in \mathbb{R}^{n \times n}$. Then $\nabla f(x) = a + 2Bx$ and $\nabla^2 f(x) = 2B$.
>
> We will show that we can converge in one step to the optimum solution point using the pre-conditioned gradient version of GD as follows. Let $x^{(i)}$ be any point in $\mathbb{R}^n$. In order to reach the optimum in one step we need
>
> $$\nabla f(x^{(i+1)}) = \vec{0}$$
> $$\Rightarrow\ a + 2Bx^{(i+1)} = \vec{0}$$

From the definition of pre-conditioned gradient descent we have that

$$\begin{aligned} & a + 2B(x^{(i)} - \eta H^{-1}\nabla f(x^{(i)})) = \vec{0} \\ \Rightarrow\ & a + 2B(x^{(i)} - \eta H^{-1}(a + 2Bx^{(i)})) = \vec{0} \\ \Rightarrow\ & a + 2Bx^{(i)} - 2\eta BH^{-1}(a + 2Bx^{(i)}) = \vec{0} \\ \Rightarrow\ & a + 2Bx^{(i)} - 2\eta BH^{-1}a - 4\eta BH^{-1}Bx^{(i)} = \vec{0} \end{aligned}$$

Now let $H = B$. We have that

$$a + 2Bx^{(i)} - 2\eta a - 4\eta Bx^{(i)} = \vec{0}$$

Now let the step size $\eta = \frac{1}{2}$ and we have that $f(x^{(i+1)}) = \vec{0}$ as desired.

Thus, we have that we can converge to the optimum solution in one step using pre-conditioned gradient descent and when $f(x)$ is quadratic. There are many other examples which converge very quickly given 2nd order information. Of course, the catch is that in deep learning the Hessian is too expensive to compute, which is where adaptive pre-conditioners (aka adaptive regularization) come in.

5.2 Analysis of GD and SGD for bounded convex f

As a warmup to analysing AdaGrad we recall the standard proof of convergence of GD and SGD. The following is the Gradient Descent algorithm.

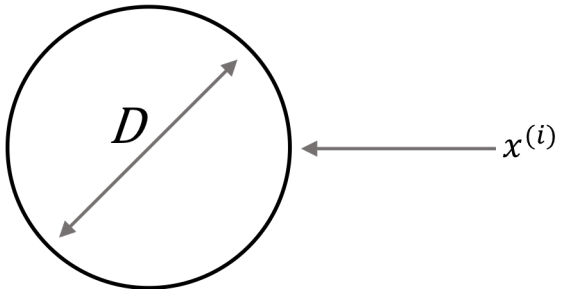
Algorithm 1 GRADIENT DESCENT

Let $\eta = \frac{D}{G\sqrt{T}}$
Repeat for $i = 0$ to T
$x^{(i+1)} \leftarrow x^{(i)} - \eta\nabla f(x^{(i)})$
At the end output $\frac{1}{T}\sum_i x^{(i)}$

We assume the learning rate η is constant throughout the iterations of gradient descent, as set later.

5.2.1 Analysis of Gradient Descent

For the analysis section, we assume that all $x^{(i)}$ reside within $\mathcal{K}$, where $\mathcal{K}$ is some bounded convex ball of diameter D, where $D = \max_{x,y\in\mathcal{K}} \|x - y\|$. This could be ensured by a projection step after each update, where any $x^{(i)}$ that strays outside of the ball is projected back onto the ball, as shown in the following diagram. For ease of notation, we will omit this projection step and assume that the points lie inside the ball.



Let x^* be the optimum point. We want to show that the iterations satisfy $f\left(\frac{1}{T}\sum_{i=1}^{T} x^{(i)}\right) \leq f(x^*) + \frac{DG}{\sqrt{T}}$, and we show this happens when $\eta = \frac{D}{G\sqrt{T}}$, where G is an upperbound on the ℓ_2 norm of the gradient in $\mathcal{K}$. Throughout this proof we will use $\|\cdot\| : \mathbb{R}^n \to \mathbb{R}$ as the Euclidean norm operator.

NB: There is a strong similarity between the proof of GD and of AdaGrad . (In class Sanjeev wrote the proofs side by side on the board.) This is highlighted via the numbering of displayed expressions, so that (5.2.1) in the GD proof is parallel to (5.3.1) in the AdaGrad proof.

Proof of GD convergence: We start with the following inequality relationship:

$$\begin{aligned}
\|x^{(i+1)} - x^*\|^2 &= \|(x^{(i)} - \eta\nabla f(x^{(i)})) - x^*\|^2 && \text{(Using the Algorithm definition)} \\
&= \|(x^{(i)} - x^*) - \eta\nabla f(x^{(i)})\|^2 && \text{(Re-ordering)} \\
&= \|(x^{(i)} - x^*)\|^2 + \eta^2\|\nabla f(x^{(i)})\|^2 - 2\eta\nabla f(x^{(i)}) \cdot (x^{(i)} - x^*) && \text{(Expansion)} \qquad (5.2.1)
\end{aligned}$$

We can then re-organize and write the following:

$$\nabla f(x^{(i)}) \cdot (x^{(i)} - x^*) = \frac{1}{2\eta}\left(\|x^{(i)} - x^*\|^2 - \|x^{(i+1)} - x^*\|^2\right) + \frac{\eta}{2}\|\nabla f(x^{(i)})\|^2 \qquad (5.2.2)$$

Now, convexity of f implies $\nabla f(x) \cdot (y - x) \leq f(y) - f(x)$. A quick proof by picture is shown below.

$$\nabla f(x) \cdot (y - x) \leq f(y) - f(x) \quad \forall x, z$$

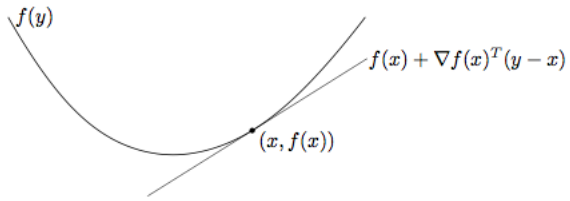


Figure 3: A differentiable convex function lies above the tangent plane $f(x) + \nabla f(x) \cdot (y - x)$

Using the convexity inequality we obtain the following.

$$f(x^{(i)}) - f(x^*) \leq \frac{1}{2\eta}\left(\|x^{(i)} - x^*\|^2 - \|x^{(i+1)} - x^*\|^2\right) + \frac{\eta}{2}\|\nabla f(x^{(i)})\|^2 \qquad (5.2.3)$$

Once we arrive at this form, we can perform telescoping. In telescoping, we will sum the above inequality over indices $i = 1, 2, ..., T$. Doing this allows us to cancel overlapping terms of $x^{(i)}$ and $x^{(i+1)}$:

$$\begin{aligned}
\sum_{i=1}^{T}(f(x^{(i)}) - f(x^*)) &\leq \frac{1}{2\eta}\sum_{i=1}^{T}\left(\|x^{(i)} - x^*\|^2 - \|x^{(i+1)} - x^*\|^2\right) + \frac{\eta}{2}\sum_{i=1}^{T}\|\nabla f(x^{(i)})\|^2 \\
&= \frac{1}{2\eta}\left(\|x^{(0)} - x^*\|^2 - \|x^{(T)} - x^*\|^2\right) + \frac{\eta}{2}\sum_{i=1}^{T}\|\nabla f(x^{(i)})\|^2 \qquad (5.2.4)
\end{aligned}$$

Now, let G = upper bound of $\|\nabla f(x)\|$ for any $x \in \mathcal{K}$. We now rewrite the equation using G and D (previously defined).

$$\sum_{i=1}^{T}(f(x^{(i)}) - f(x^*)) \leq \frac{1}{2\eta}D^2 + \frac{\eta}{2}TG^2 \tag{5.2.5}$$

We divide both sides by T and we have

$$\frac{1}{T}\sum_{i=1}^{T}(f(x^{(i)})) - f(x^*) \leq \frac{D^2}{2\eta T} + \frac{\eta}{2}G^2 \tag{5.2.6}$$

Since f is convex we know that $f(\frac{1}{T}\sum_i x^{(i)}) \leq \frac{1}{T}\sum_i f(x^{(i)})$. After applying this we have

$$f\left(\frac{1}{T}\sum_{i=1}^{T}x^{(i)}\right) - f(x^*) \leq \frac{D^2}{2\eta T} + \frac{\eta}{2}G^2 \tag{5.2.7}$$

Finally, setting that $\eta = \frac{D}{G\sqrt{T}}$ gives us

$$f\left(\frac{1}{T}\sum_{i=1}^{T}x^{(i)}\right) \leq f(x^*) + \frac{DG}{\sqrt{T}} \tag{5.2.8}$$

5.2.2 Analysis of Stochastic Gradient Descent

The algorithm for SGD is the same as GD with the exception that we have an expected gradient oracle. Showing that the expected output of the algorithm converges follows the same format as the previous proof.

Theorem 1. $\mathrm{E}\left[f\left(\frac{1}{T}\sum_{i=1}^{T}x^{(i)}\right)\right] \leq f(x^*) + \frac{2DG}{\sqrt{T}}$, *where D is the diameter as before and G is an upperbound of the norm of any gradient vector ever output by the oracle.*

Proof: Let f_i be the gradient in GD and g_i be the gradient in SGD.

$$\begin{aligned}
f\left(\frac{1}{T}\sum_{i=1}^{T}x^{(i)}\right) - f(x^*) &\leq \frac{1}{T}\sum_i(f(x^{(i)})) - f(x^*)) && \text{(by convexity of } f\text{)}\\
&\leq \frac{1}{T}\sum_i(\nabla f(x^{(i)} \cdot (x^{(i)} - x^*))\\
&= \frac{1}{T}\sum_i \mathrm{E}[g_i \cdot (x^{(i)} - x^*)] && \text{(Since expected gradient is the true gradient)}\\
&= \frac{1}{T}\sum_i \mathrm{E}[f_i(x^{(i)}) - f_i(x^*)] && \text{(Definition of } f_i\text{)}\\
&= \frac{1}{T}\mathrm{E}[\sum_i(f_i(x^{(i)}) - f_i(x^*))]\\
&\leq \frac{2DG}{\sqrt{T}}
\end{aligned}$$

5.3 Adagrad: Adaptive Pre-conditioner

In practice the choice of pre-conditioner is unclear given only gradient information, and various heuristic methods have been proposed for designing pre-conditioners. AdaGrad (Adaptive subGradient method) is designed to yield a provable guarantee. Starting with a trivial pre-conditioner, it iteratively updates it, such that at the end one can guarantee that the total descent achieved is within factor $2D$, D the diameter of the optimization region, with what could be achieved with the best fixed pre-conditioner in hindsight (which is unknown at the start, of course). This guarantee sounds too good to be true and of course it is. There is an important qualification in this statement which we will clarify later. Note also that the provable guarantees are only for convex f; the nonconvex case remains open.

AdaGrad starts with pre-conditioner I (i.e., no modification to gradient) and at the t-th iteration uses:

$$G_t = \left(\sum_{i=1}^{t} \nabla f(x_i) \nabla f(x_i)^T \right)^{\frac{1}{2}}.$$

From this, we have the following update rule for AdaGrad:

$$x_{t+1} = x_t - \eta G_t^{-1} \nabla f(x_t)$$

To analyse this algorithm we will show the convergence of AdaGrad in the same manner as we did for GD. That is, we will show that $f\left(\frac{1}{T}\sum_i x_i\right) \leq f(x^*) + \frac{2D}{T}\mathbf{Tr}(G_T)$. This time, we will be using matrix norms instead of Euclidean norms.

Proof of AdaGrad Convergence: Let $\mathcal{K}$ be a bounded convex set that we are optimizing over. Let $\{x_t\}$ be the sequence of values achieved by the AdaGrad algorithm. Let $D = \max_{u \in \mathcal{K}} \|u - x_1\|$

Define the matrix norm operator $\|\cdot\|_A^2 : \mathbb{R}^n \to \mathbb{R}$ as

$$\|x\|_A^2 \triangleq x^T A x$$

We now proceed to prove convergence for AdaGrad in the same style as the proof of GD. Refer to the equation numbers to see the parallels between the two proofs. Again, we assume that every x_i resides within $\mathcal{K}$ (otherwise we just project x_i onto $\mathcal{K}$).

$$\begin{aligned} \|x_{i+1} - x^*\|_{G_i}^2 &= \left\|x_i - \eta G_i^{-1} f(x_i) - x^*\right\|_{G_i}^2 \\ &= \left\|(x_i - x^*) - \eta G_i^{-1} \nabla f(x_i)\right\|_{G_i}^2 \\ &= \|x_i - x^*\|_{G_i}^2 + \eta^2 \|\nabla f(x_i)\|_{G_i^{-1}}^2 - 2\eta \nabla f(x_i) \cdot (x_i - x^*) \end{aligned} \tag{5.3.1}$$

By rearranging terms we have that

$$\nabla f(x_i) \cdot (x_i - x^*) = \frac{1}{2\eta}\left(\|x_i - x^*\|_{G_i}^2 - \|x_{i+1} - x^*\|_{G_i}^2\right) + \frac{\eta}{2}\|\nabla f(x_i)\|_{G_i^{-1}}^2 \tag{5.3.2}$$

By convexity we know that $(\nabla f(x))^T (y - x) \leq f(y) - f(x)$ so we have that

$$f(x_i) - f(x^*) \leq \frac{1}{2\eta}\left(\|x_i - x^*\|_{G_i}^2 - \|x_{i+1} - x^*\|_{G_i}^2\right) + \frac{\eta}{2}\|\nabla f(x_i)\|_{G_i^{-1}}^2 \tag{5.3.3}$$

We now sum over $i = 1, 2, ..., T$.

$$\begin{aligned}\sum_{i=1}^{T}(f(x_i) - f(x^*)) &\leq \frac{1}{2\eta}\sum_{i=1}^{T}\left(\|x_i - x^*\|_{G_i}^2 - \|x_{i+1} - x^*\|_{G_i}^2\right) + \frac{\eta}{2}\sum_{i=1}^{T}\|\nabla f(x_i)\|_{G_i^{-1}}^2 \\ &= \frac{1}{2\eta}\|x_1 - x^*\|_{G_0}^2 + \frac{1}{2\eta}\sum_{i=1}^{T}\left(\|x_i - x^*\|_{G_i}^2 - \|x_i - x^*\|_{G_{i-1}}^2\right) \\ &\quad - \frac{1}{2\eta}\|x_{T+1} - x^*\|_{G_T}^2 + \frac{\eta}{2}\sum_{i=1}^{T}\|\nabla f(x_i)\|_{G_i^{-1}}^2\end{aligned} \tag{5.3.4}$$

Because $G_0 = \mathbf{0}$ we have that $\|x_1 - x^*\|_{G_0}^2 = 0$. By Lemma 5.12 in OCO book we have that $\sum_{i=1}^{T}\|\nabla f(x_i)\|_{G_i^{-1}}^2 \leq 2\mathbf{Tr}(G_T)$. By Lemma 5.13 in OCO book hwe have that $\sum_{i=1}^{T}\left(\|x_i - x^*\|_{G_i}^2 - \|x_i - x^*\|_{G_{i-1}}^2\right) \leq D^2\mathbf{Tr}(G_T)$. Thus, we have that

$$\sum_{i=1}^{T}(f(x_i) - f(x^*)) \leq \frac{1}{2\eta}D^2\mathbf{Tr}(G_T) + \frac{\eta}{2}\left(2\mathbf{Tr}(G_T)\right) \tag{5.3.5}$$

Dividing both sides by T we have

$$\frac{1}{T}\sum_{i=1}^{T}(f(x_i) - f(x^*)) \leq \frac{D^2\mathbf{Tr}(G_T)}{2\eta T} + \frac{\eta\mathbf{Tr}(G_T)}{T} \tag{5.3.6}$$

By convexity we have that $f(\frac{1}{T}\sum_i x_i) \leq \frac{1}{T}\sum_i f(x_i)$ so we have that

$$f\left(\frac{1}{T}\sum_{i=1}^{T}x_i\right) - f(x^*) \leq \frac{D^2\mathbf{Tr}(G_T)}{2\eta T} + \frac{\eta\mathbf{Tr}(G_T)}{T} \tag{5.3.7}$$

If we let $\eta = D$ then we have that

$$f\left(\frac{1}{T}\sum_{i=1}^{T}x_i\right) \leq f(x^*) + \frac{2D}{T}\mathbf{Tr}(G_T) \tag{5.3.8}$$

5.3.1 Competitiveness vs the optimal pre-conditioner:

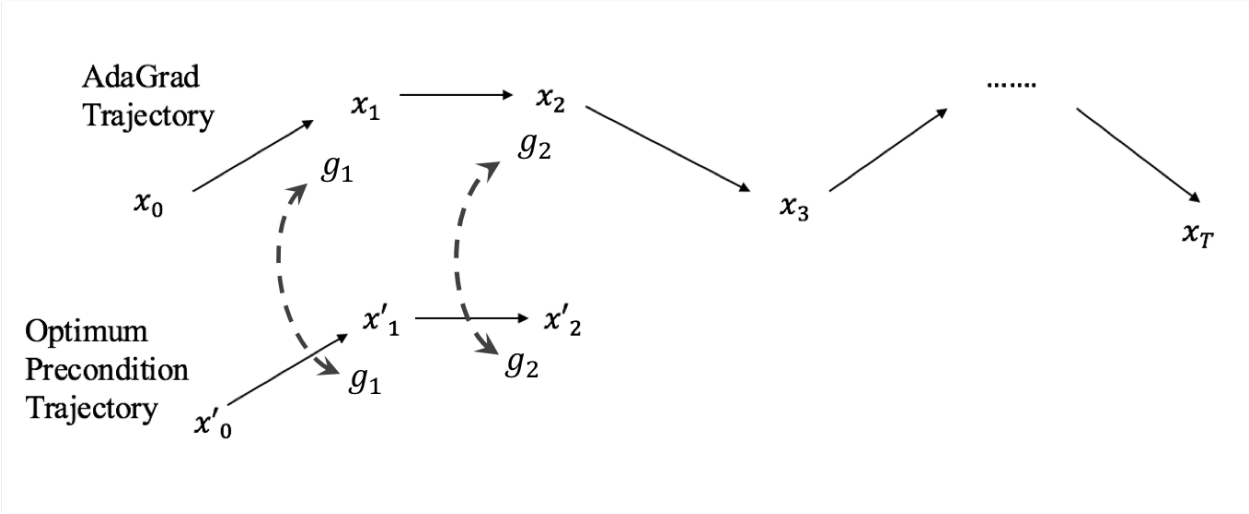
It is often stated that AdaGrad is approximately competitive with the pre-conditioner that's best in hindsight, meaning its performance after T steps tracks that of the optimum preconditioner. We'll try to understand what this means. Note that the statement cannot be true as stated, as illustrated by our earlier example of the quadratic function which can be optimized in one step by the optimal preconditioner, something AdaGrad cannot do.

Instead, the competitiveness is with respect to the following setting: the optimal algorithm is provided with the same sequence of gradients as AdaGrad is. Thus the optimal algorithm has the updates

Now lets put this result in context. We compare AdaGrad to the optimal algorithm with a fixed pre-conditioner, but the optimal algorithm is restricted to use the same gradient at each step as AdaGrad. If AdaGrad moves at step i as $x_{i+1} \leftarrow x_i - \eta G_i^{-1}\nabla f(x_i)$ then the optimal algorithm moves as

$$x'_{i+1} \leftarrow x'_i - \eta H^{-1}\nabla f(x_i)$$

where H is the best pre-conditioner in hindsight. Note that both have to use the gradient $\nabla f(x_i)$. See the picture below.



The justification for this setup is that adaptive preconditioners try to learn a good preconditioner from gradients, and this match-up is testing AdaGrad's competitiveness with the best.

Comparing Optimal Algorithm to AdaGrad: The question is what is the optimal preconditioner H. This involves redoing the AdaGrad analysis, specifically, getting to expression (5.3.4), setting $G_t = H$ for all t, and then solving for best H. This can be done and Corollary 5.10 in the OCO book shows that H is related to G_T, and the distance of the objective function to the optimum value after T steps is upper bounded by $\frac{1}{T}\mathbf{Tr}(G_T)$.

$$f\left(\frac{1}{T}\sum_{i=1}^{T} x'_i\right) - f(x^*) \leq \frac{1}{T}\mathbf{Tr}(G_T)$$

From the aforementioned proof, we have that by using an adaptive pre-conditioner in AdaGrad, the distance of the objective function to the optimum value after T steps is upper bounded by $\frac{2D}{T}\mathbf{Tr}(G_T)$.

$$f\left(\frac{1}{T}\sum_{i=1}^{T} x_i\right) - f(x^*) \leq \frac{2D}{T}\mathbf{Tr}(G_T)$$

This shows that the upper bound on distance to optimality for AdaGrad at step T is within a factor of $2D$ of the upper bound on the distance to optimality for the best fixed pre-conditioner algorithm. Remarkably, AdaGrad accomplishes this without knowing what the best pre-conditioner is in hindsight.

Remark: Actually the expression in (5.3.4) is an upperbound on the true distance so AdaGrad is within factor $2D$ of the best upperbound. Thus the approximation ratio may be worse with respect to true distance to optimum.

5.4 Closing Remarks: Diagonal pre-conditioners and adaptive regularization

Currently the pre-conditioner matrix G_t is too big to compute in deep learning. A proposed solution to this is instead to just keep the diagonal entries of $\sum \nabla f \nabla f^T$.

$$\widehat{G}_t = \text{diag}\left(\left(\sum_{i=1}^{t}(\nabla f(x_i))_j^2\right)^{\frac{1}{2}}\Bigg|_{j\in[n]}\right)$$

When we use $\widehat{G}_t$ as the pre-conditioner it looks something like the following:

$$\widehat{G}_t^{-1}\nabla f(x_t) = \begin{pmatrix}\eta_1 & & & \\ & \eta_2 & & \\ & & \ddots & \\ & & & \eta_n\end{pmatrix}\nabla f(x_t)$$

Essentially, $\widehat{G}_t$ acts as an adjustable learning rate. There currently isn't any good theory for this setting.

Another point to note is that $(\widehat{G}_t)_j^{-1} = \left(\sum_{i=1}^{t}(\nabla f(x_i))_j^2\right)^{-\frac{1}{2}}$ and essentially acts to slow down coordinates that are changing too fast.

The above idea inspired discovery of other algorithms such as RMSProp and Adam which also use similar adjustable learning rate approaches. Whereas AdaGrad weighs history equally when determining the pre-conditioner, Adam weighs recent history with higher weight.

Recently, it was found that there are examples of convex optimization settings where Adam does not converge (i.e. there are bugs in the original proof of convergence).

Task: Understand Adam better in convex and non-convex settings.

5.5 References

1. S. Arora. Advanced Algorithm Design Lecture 14, 2016.
2. E. Hazan. Introduction to Online Convex Optimization. 2017.
`http://ocobook.cs.princeton.edu/OCObook.pdf`