Lecture 8: Minibatching and Decreasing Step Sizes

CS4787/5777 — Principles of Large-Scale Machine Learning Systems

In [1]:
```
import numpy
import scipy
import matplotlib
from matplotlib import pyplot
import time

matplotlib.rcParams.update({'font.size': 18})
```

Where we left off: SGD for Strongly Convex Objectives

We want to get a sense of how SGD will perform in the "easy-to-analyze" case of strongly convex objectives, so that this will give us more intuition about how SGD behaves at scale. We start with:

$$\mathbf{E}\left[f(w_{t+1})\right] \le \mathbf{E}\left[f(w_t)\right] - \frac{\alpha}{2}\mathbf{E}\left[\left\|\nabla f(w_t)\right\|^2\right] + \frac{\alpha^2\sigma^2 L}{2B},$$

where B is the minibatch size, σ^2 is a bound on the example gradient variance, L is the smoothness constant, and α is the step size.

As we did when we analyzed gradient descent, we apply the Polyak–Lojasiewicz condition,

$$\left\|\nabla f(x)\right\|^2 \ge 2\mu\left(f(x) - f^*\right).$$

This gives us

$$\mathbf{E}\left[f(w_{t+1})\right] \le \mathbf{E}\left[f(w_t)\right] - \mu\alpha\mathbf{E}\left[f(w_t) - f^*\right] + \frac{\alpha^2\sigma^2 L}{2B}.$$

Subtracting f^* from both sides,

$$\mathbf{E}\left[f(w_{t+1}) - f^*\right] \le \mathbf{E}\left[f(w_t) - f^*\right] - \mu\alpha\mathbf{E}\left[f(w_t) - f^*\right] + \frac{\alpha^2\sigma^2 L}{2B}.$$

And gathering terms

$$\mathbf{E}\left[f(w_{t+1}) - f^*\right] \le (1 - \mu\alpha)\mathbf{E}\left[f(w_t) - f^*\right] + \frac{\alpha^2\sigma^2 L}{2B}.$$

So, we have this recurrence relation

$$\mathbf{E}\left[f(w_{t+1}) - f^*\right] \leq (1 - \mu\alpha)\mathbf{E}\left[f(w_t) - f^*\right] + \frac{\alpha^2\sigma^2 L}{2B}$$

and it's a bit complicated to see what's going on. To make the writing a bit more terse, let ρ_t denote the expected objective gap at timestep t, i.e. $\rho_t = \mathbf{E}\left[f(w_t) - f^*\right]$. Then,

$$\rho_{t+1} \leq (1 - \mu\alpha)\rho_t + \frac{\alpha^2\sigma^2 L}{2B}.$$

This is an example of a linear recurrence (albeit one with an inequality). To "solve" this sort of maths problem, we first find the fixed point: the ρ where

$$\rho \leq (1 - \mu\alpha)\rho + \frac{\alpha^2\sigma^2 L}{2B}.$$

It's pretty easy to see that the solution here is

$$\rho = \frac{1}{\mu\alpha} \cdot \frac{\alpha^2\sigma^2 L}{2B} = \frac{\alpha\sigma^2 L}{2B\mu}.$$

Subtracting this fixed point from both sides of our inequality gives us

$$\rho_{t+1} - \frac{\alpha\sigma^2 L}{2B\mu} \leq (1 - \mu\alpha)\rho_t + \frac{\alpha^2\sigma^2 L}{2B} - \frac{\alpha\sigma^2 L}{2B\mu}.$$

This simplifies to

$$\rho_{t+1} - \frac{\alpha\sigma^2 L}{2B\mu} \leq (1 - \mu\alpha)\left(\rho_t - \frac{\alpha\sigma^2 L}{2B\mu}\right).$$

So we're left with

$$\rho_{t+1} - \frac{\alpha\sigma^2 L}{2B\mu} \leq (1 - \mu\alpha)\left(\rho_t - \frac{\alpha\sigma^2 L}{2B\mu}\right).$$

And now we recognize the same geometric decay that we saw in the analysis of gradient descent! (That is, if $x_{t+1} \leq c x_t$ for $c > 0$, then $x_K \leq c^K x_0$.) Applying the same reasoning gives us

$$\rho_K - \frac{\alpha\sigma^2 L}{2B\mu} \leq (1 - \mu\alpha)^K\left(\rho_0 - \frac{\alpha\sigma^2 L}{2B\mu}\right).$$

We can simplify this a little by dropping the subtracted term on the right,

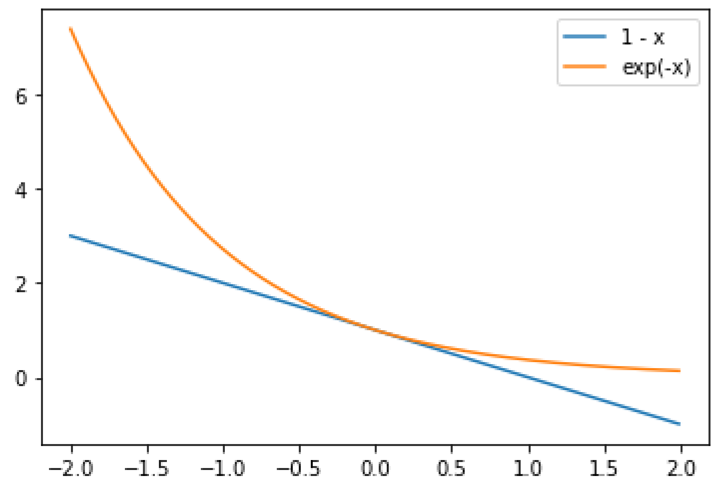
$$\rho_K - \frac{\alpha \sigma^2 L}{2 B \mu} \le \exp(-\mu \alpha K) \cdot \rho_0.$$

Now moving the subtracted term from the left to the right and substituting in our defintion of ρ,

$$\mathbf{E}\left[f(w_K) - f^* \right] \le (1 - \mu \alpha)^K (f(w_0) - f^*) + \frac{\alpha \sigma^2 L}{2 B \mu}.$$

We can simplify this a little more by leveraging the fact that $1 - \mu \alpha \le \exp(-\mu \alpha)$ so $(1 - \mu \alpha)^K \le \exp(-\mu \alpha)^K = \exp(-\mu \alpha K)$.

In [2]:
```
x = numpy.arange(-2,2,0.01)
pyplot.plot(x, 1-x, label="1 - x")
pyplot.plot(x, numpy.exp(-x), label="exp(-x)")
pyplot.legend()
pyplot.show()
```



This gives us

$$\mathbf{E}\left[f(w_K) - f^* \right] \le \exp(-\mu \alpha K) \cdot (f(w_0) - f^*) + \frac{\alpha \sigma^2 L}{2 B \mu}.$$

What can we learn from this expression?

Some takeaways:

$$\mathbf{E}\left[f(w_K) - f^* \right] \le \exp(-\mu \alpha K) \cdot (f(w_0) - f^*) + \frac{\alpha \sigma^2 L}{2 B \mu}.$$

- With gradient descent, if we wanted to get a solution of a desired level of accuracy we could just keep running until we observed a gradient small enough to satisfy our desires.

- With SGD using a fixed step size, this won't necessarily happen.

One way to achieve a desired level of error...

...is to choose the hyperparameters α and K as a function of the error. To achieve a guarantee that the expected loss gap will be no greater than ϵ from our formula

$$\mathbf{E}\left[f(w_K) - f^*\right] \le \exp(-\mu\alpha K) \cdot (f(w_0) - f^*) + \frac{\alpha\sigma^2 L}{2B\mu},$$

it suffices to choose α and K such that

$$\frac{\epsilon}{2} \ge \exp(-\mu\alpha K) \cdot (f(w_0) - f^*) \;\text{ and }\; \frac{\epsilon}{2} \ge \frac{\alpha\sigma^2 L}{2B\mu}.$$

Now solving for α and K, and remembering that we already required $\alpha \le 1/L$, gives us

$$\alpha = \min\left(\frac{\epsilon B\mu}{\sigma^2 L}, \frac{1}{L}\right) \;\text{ and }\; K = \frac{1}{\alpha\mu} \cdot \log\left(\frac{2(f(w_0) - f^*)}{\epsilon}\right).$$

In terms of the condition number $\kappa = L/\mu$, this gives us

$$K = \max\left(\frac{\sigma^2}{\epsilon B\mu}, 1\right) \cdot \kappa \cdot \log\left(\frac{2(f(w_0) - f^*)}{\epsilon}\right)$$

In comparison, gradient descent had

$$K \ge \kappa \cdot \log\left(\frac{f(w_0) - f^*}{\epsilon}\right).$$

What can we conclude from this? Here's one thing that we can get: the asymptotic runtime used by these algorithms. **For each of strongly convex GD/SGD, write a big-$\mathcal{O}$ expression for the total amount of compute that would be done by the algorithm to achieve error ϵ. Give your result in terms of ϵ, κ (for strongly-convex), n, and σ^2, treating all other expressions (such as $f(w_0) - f^*$) as constant.**

GD: $n\kappa \log\left(\frac{1}{\epsilon}\right)$

SGD: $\max(\frac{\sigma^2}{\epsilon B\mu}, 1) \cdot \kappa \cdot \log\left(\frac{1}{\epsilon}\right)$

When might SGD be better than gradient descent and vice versa?

- For sufficiently small ϵ, minibatching with batch size B decreases the number of iterations needed to get to objective gap ϵ by a factor of B.

- **But doing minibatching with batch size B increases the amount of compute we need to do by a factor of B!**
- Can also increase the batch size over time...eventually doing gradient descent.

So is there any point to using minibatching?

...

Diminishing Step Size Schemes

Another thing that our analysis of SGD motivates us to do is to think of ways we could recover the asymptotic-convergence-to-the-global-optimum that gradient descent enjoys. Since we saw that constant-step-size SGD has convergence that is limited by a "noise ball" with size proportional to the step size, a natural way to try to fix this is to **decrease the step size over time**.

Let's suppose that we run SGD with a step size that changes over time, and let α_t denote the step size used at time t. Then the same analysis we used for constant-step-size SGD (on strongly convex objectives) will give us (where $\rho_t = \mathbf{E}\left[f(w_t) - f^*\right]$ is the expected objective gap at timestep t)

$$\rho_{t+1} \leq (1 - \mu\alpha_t)\rho_t + \frac{\alpha_t^2 \sigma^2 L}{2}.$$

One way to try to "derive" what a good diminishing step size scheme will look like is to minimize this expression with respect to α_t. Differentiating with respect to α_t to minimize gives

$$0 = -\mu\rho_t + \alpha_t \sigma^2 L,$$

which simplifies to

$$\alpha_t = \frac{\mu\rho_t}{\sigma^2 L}.$$

Substituting this back into our inequality

$$\rho_{t+1} \leq \left(1 - \mu\frac{\mu\rho_t}{\sigma^2 L}\right)\rho_t + \frac{\sigma^2 L}{2} \cdot \left(\frac{\mu\rho_t}{\sigma^2 L}\right)^2 = \rho_t - \frac{\mu^2}{2\sigma^2 L}\rho_t^2.$$

So we have

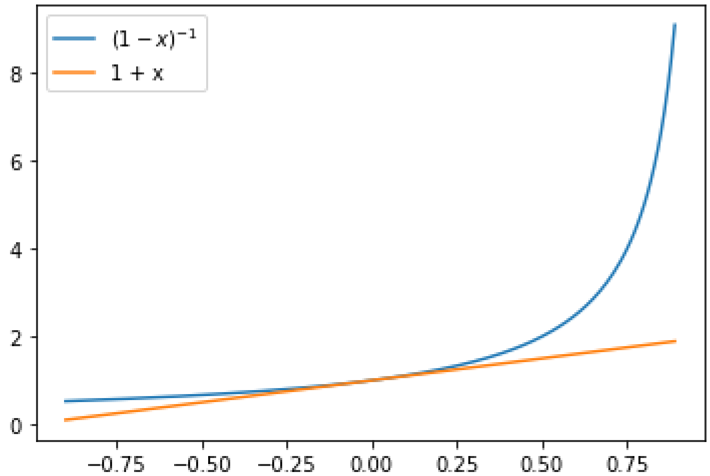
$$\rho_{t+1} \leq \rho_t - \frac{\mu^2}{2\sigma^2 L}\rho_t^2.$$

Inverting both sides gives us

$$\frac{1}{\rho_{t+1}} \geq \left(\rho_t - \frac{\mu^2}{2\sigma^2 L} \rho_t^2 \right)^{-1} = \frac{1}{\rho_t} \cdot \left(1 - \frac{\mu^2}{2\sigma^2 L} \rho_t \right)^{-1}.$$

Noticing that $(1-x)^{-1} \geq 1 + x$

In [3]:
```
x = numpy.arange(-0.9,0.9,0.01)
pyplot.plot(x, 1/(1-x), label="$(1 - x)^{-1}$")
pyplot.plot(x, 1+x, label="1 + x")
pyplot.legend()
pyplot.show()
```



we can get

$$\frac{1}{\rho_{t+1}} \geq \frac{1}{\rho_t} \cdot \left(1 + \frac{\mu^2}{2\sigma^2 L} \rho_t \right) = \frac{1}{\rho_t} + \frac{\mu^2}{2\sigma^2 L}.$$

Finally, applying this recursively gives us

$$\frac{1}{\rho_K} \geq \frac{1}{\rho_0} + \frac{\mu^2 K}{2\sigma^2 L} \geq \frac{\mu^2 K}{2\sigma^2 L}.$$

So,

$$\mathbf{E}\left[f(w_K) - f^* \right] = \rho_K \leq \frac{2\sigma^2 L}{\mu^2 K},$$

and this assignment of ρ_K suggests we should set α_t to be

$$\alpha_t = \frac{\mu}{\sigma^2 L} \cdot \frac{2\sigma^2 L}{\mu^2 t} = \frac{2}{\mu t}.$$

This is called a $1/t$ step size scheme, and it's very common in ML and in optimization more generally!

- Use it, or other step size schemes like it, when you believe your loss function is fairly smooth and either strongly convex or "strongly-convex-like" nearby its local minima.

```
In [ ]:
```