MLCC 2022
Stochastic gradient descent and friends

Lorenzo Rosasco
MaLGa, DIBRIS, UNIGE-MIT-IIT

About this class

- We introduce a SGD (Stochastic Gradient Descent) the workhorse in ML
- We start gently with some background on gradient methods

ERM

Remember[1]

$$\min_{w \in \mathbb{R}^d} \hat{\mathcal{E}}_\lambda(w) = \hat{\mathcal{E}}(w) + \lambda \left\| w \right\|^2, \quad \lambda \geq 0,$$

with

$$\hat{\mathcal{E}}(w) = \frac{1}{n} \sum_{i=1}^{n} \ell(y_i, w^\top x_i)$$

This is an optimization problem we need to solve!

[1] We simplify the notation here $\mathcal{E}(w) = \mathcal{E}(f_w)$.

Optimality conditions

Recall that $\widehat{w}$ is a minimizer iff

$$\nabla_w \hat{\mathcal{E}}_\lambda(\hat{w}) = 0$$

For least squares $(y - w^\top x)^2$: quadratic error+quadratic norm
$\Rightarrow$ linear equations.

Optimality conditions

Recall that $\widehat{w}$ is a minimizer iff

$$\nabla_w \hat{\mathcal{E}}_\lambda(\hat{w}) = 0$$

For for logistic regression $\log(1 + e^{-yw^x})$, we get non linear equations...

$$\nabla \hat{\mathcal{E}}(w) = \frac{1}{n} \sum_{i=1}^{n} \frac{-y_i x_i e^{-y_i x_i^T w}}{1 + e^{-y_i x_i^T w}} + 2\lambda w = 0.$$

Solving ERM

$$\min_{w \in \mathbb{R}^d} \hat{\mathcal{E}}_\lambda(w)$$

How do we solve it then??

(Spoiler SGD)

Optimization

Let $F : \mathbb{R}^d \to \mathbb{R}$ find

$$\min_{w \in \mathbb{R}^d} F(w).$$

Life is good if F smooth, but especially if F is convex!

We assume both conditions are true for now!

Gradient based optimization

Before introducing SGD we take a detour and discuss

- Newton method
- Gradient descent

Newton method

$F : \mathbb{R} \to \mathbb{R}$ smooth (twice differentiable), convex.

$$w_{t+1} = w_t - \gamma_t F'(w_t), \qquad \gamma_t = [F''(w_t)]^{-1}$$

A picture might help here.

Newton method (cont.)

- The iterative process takes steps in the opposite direction of the gradient
- The steps-size depends on how "curved" is the function - hence the second derivative

Newton's is a second order method - it uses both first and second derivatives.

Newton method in multiple dimensions

$F : \mathbb{R}^d \to \mathbb{R}$ smooth (twice differentiable), convex.

$$w_{t+1} = w_t - \gamma_t \nabla F(w_t) \qquad \gamma_t = [H_F(w_t)]^{-1}$$

- First derivatives become gradients $F' \mapsto \nabla F$
- Second derivatives become Hessians $F'' \mapsto H_F$

The problem with Newton method

$F : \mathbb{R}^d \to \mathbb{R}$ smooth (twice differentiable), convex.

$$w_{t+1} = w_t - \gamma_t \nabla F(w_t) \qquad \gamma_t = [H_F(w_t)]^{-1}$$

Computing and inverting the an Hessian matrix at each step is not feasible/efficient.

From Newton method to gradient descent

$F : \mathbb{R}^d \to \mathbb{R}$ smooth (twice differentiable), convex.

$$w_{t+1} = w_t - \gamma_t \nabla F(w_t) \qquad (\gamma_t)_t \quad \text{a "suitable" step-size sequence}$$

Gradient descent is a first order method - it uses only first derivatives.

How do we choose the steps-size?

Back to logistic regression with GD

Recalling

$$\nabla \hat{\mathcal{E}}(w) = \frac{1}{n} \sum_{i=1}^{n} \frac{-y_i x_i e^{-y_i x_i^T w}}{1 + e^{-y_i x_i^T w}} + 2\lambda w$$

The GD iteration becomes

$$\widehat{w}_{t+1} = \widehat{w}_t - \gamma \left(\frac{1}{n} \sum_{i=1}^{n} \frac{-y_i x_i e^{-y_i x_i^T \widehat{w}_t}}{1 + e^{-y_i x_i^T \widehat{w}_t}} + 2\lambda \widehat{w}_t \right)$$

GD the goods and the bads

$$\widehat{w}_{t+1} = \widehat{w}_t - \gamma \left(\frac{1}{n} \sum_{i=1}^{n} \frac{-y_i x_i e^{-y_i x_i^T \widehat{w}_t}}{1 + e^{-y_i x_i^T \widehat{w}_t}} + 2\lambda \widehat{w}_t \right)$$

- Possibly slow but requires only vector multiplications, hence easily parallelizable.
- But at each iteration all the data needs be stored and processed.

From GD to SGD

For a general (smooth loss function) GD is

$$\widehat{w}_{t+1} = \widehat{w}_t - \gamma \left(\frac{1}{n} \sum_{i=1}^{n} \nabla \ell(y_i, x_i \widehat{w}_t) + 2\lambda \widehat{w}_t \right)$$

SGD corresponds to

$$\widehat{w}_{t+1} = \widehat{w}_t - \gamma_t \left(\nabla \ell(y_{i_t}, x_{i_t} \widehat{w}_t) + 2\lambda \widehat{w}_t \right)$$

Only one data point is processed at each iteration!

SGD

$$\widehat{w}_{t+1} = \widehat{w}_t - \gamma_t \left(\nabla \ell(y_{i_t}, x_{i_t}^\top \widehat{w}_t) + 2\lambda \widehat{w}_t \right)$$

Here i_t corresponds to some selection criterion for the data at each iteration, typically uniformly at random.

Note that the step-size cannot be chosen constant! An ad hoc analysis is needed.

Why SGD?

One reason is the small time/memory cost per iteration.

Another reason is the possibility of dealing with streaming data

$$\widehat{w}_{t+1} = \widehat{w}_t - \gamma_t \left(\nabla \ell(y_t, x_t^\top \widehat{w}_t) + 2\lambda \widehat{w}_t \right)$$

SGD Flavors

SGD is hardly ever used in the basic form

$$\widehat{w}_{t+1} = \widehat{w}_t - \gamma_t \left(\nabla \ell(y_{i_t}, x_{i_t}^\top \widehat{w}_t) + 2\lambda \widehat{w}_t \right).$$

Possible variations:

- data selection
- mini- batching
- acceleration
- averaging

Data sampling

$$\widehat{w}_{t+1} = \widehat{w}_t + \gamma_t \left(\nabla \ell(y_{i_t}, x_{i_t}^\top \widehat{w}_t) + 2\lambda \widehat{w}_t \right).$$

- Sample data uniformly at random.
- Visit data in a fixed prescribed order.
- Visit data sequentially, reshuffling after each pass.

Mini-batching

$$\widehat{w}_{t+1} = \widehat{w}_t + \gamma_t \left(\frac{1}{b} \sum_{j=(b(t-1)+1)}^{bt} \nabla \ell(y_j, x_j^\top \widehat{w}_t) + 2\lambda \widehat{w}_t \right)$$

- Uses b points at each iteration for a more accurate gradient estimate.
- Allows efficient memory use.
- First step towards using distributed resources.

Acceleration: momentum

$$
\begin{aligned}
\widehat{w}_{t+1} &= \hat{v}_t + \gamma_t \left(\nabla \ell(y_{i_t}, x_{i_t}^\top \widehat{v}_t) + 2\lambda \widehat{v}_t \right) \\
\hat{v}_t &= \widehat{w}_t + \beta_t(\widehat{w}_t - \widehat{w}_{t-1})
\end{aligned}
$$

- The momentum is $\widehat{w}_t - \widehat{w}_{t-1}$.
- Two previous iterations are used, with potential increased speed.
- Above, the momentum is used as proposed by Nesterov.

(Polyak) averaging

$$\overline{w}_t = \frac{1}{t} \sum_{j=s}^{t} a_j \widehat{w}_j$$

- $s = a_t = 1$ uniform averaging.
- $s > 1$ tail averaging.
- $a_t \neq 1$ weighted (Cesàro) averages.

Further remarks

- Further constraints can be imposed on the iteration
- Further acceleration are possible with smarter gradient estimates
- SGD can be extended to non smooth but convex functions keeping all guarantees!
- SGD can be extended to non convex functions loosing most guarantees!

Summing up

- First order methods dominate modern learning

- SGD is the method of choice

- SGD is always used with a number of tweaks!

Next class

... beyond linear models!