Machine Learning
Large Scale Learning

Dr. Gerard Pons-Moll

23.01.2019

Large Scale Machine Learning

Optimization problems in machine learning often have the form

$$\min_{w \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^{n} L(y_i, \langle w, x_i \rangle) + \lambda \Omega(w).$$

What does large scale mean ?

- too large to fit into memory
- so large that training on subset yields already good results
- hyperparameter selection is done by optimizing on a validation set

Observation: very accurate solution not required !

Program for today

Large Scale Learning

- Interaction of Learning and Optimization
  How does the game change when one has huge amounts of training data ?
- Stochastic/Cyclic Dual Coordinate Ascent
- Stochastic Gradient Descent

Classical Tradeoff in Learning

Goal: Bayes optimal classifier $f^* = \arg\min_f \mathbb{E}[L(Y, f(X))] =: R(f)$

- optimization over all functions impossible - choose function class $\mathcal{F}$ best function in $\mathcal{F}$:

$$f^*_{\mathcal{F}} = \arg\min_{f \in \mathcal{F}} \mathbb{E}[L(Y, f(X))].$$

- only n i.i.d. samples - replace expectation with empirical average **empirical risk minimization:**

$$f^*_n = \arg\min_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} L(Y_i, f(X_i)) =: R_n[f].$$

Classical (Small-Scale) Tradeoff in Learning:

$$\underbrace{R(f_n) - R(f^*)}_{\textbf{excess risk}} = \underbrace{R(f_n) - R(f^*_{\mathcal{F}})}_{\textbf{estimation error}} + \underbrace{R(f^*_{\mathcal{F}}) - R(f^*)}_{\textbf{approximation error}}$$

Tradeoff in Large-Scale Learning

Key idea: Optimize up to the estimation error (Bottou, Bousquet (2008)).

Learning and Optimization should be seen as joint problem !

Optimization error as a new source of error

- we only estimate function $\tilde{f}$ with:

$$R_n(\tilde{f}) < R_n(f_n) + \rho,$$

that is we get empirical risk minimizer only up to accuracy ρ

Goal: Minimize excess risk of $\tilde{f}$,

$$\underbrace{R(\tilde{f}) - R(f^*)}_{\textbf{excess risk}} = \underbrace{R(\tilde{f}) - R(f_n)}_{\textbf{optimization error}} + \underbrace{R(f_n) - R(f^*_{\mathcal{F}})}_{\textbf{estimation error}} + \underbrace{R(f^*_{\mathcal{F}}) - R(f^*)}_{\textbf{approximation error}}$$

subject to $n < n_{\max}$ (label budget) and $t < t_{\max}$ (time budget)

Analysis

The estimation error behaves typically as $\frac{1}{\sqrt{n}}$ and as $\frac{1}{n}$ (fast rates).

- **small scale:** bounded by $n_{\max}$ - minimize ρ as much as possible
- **large scale:** bounded by $t_{\max}$ - allow larger ρ (on the order of the estimation error) in order to process more samples n.

Recap: Gradient Descent

Optimization problem:

$$\min_{w \in \mathbb{R}^d} \phi(w) := \frac{1}{n} \sum_{i=1}^{n} L(y_i, \langle w, x_i \rangle) + \lambda \Omega(w).$$

General gradient descent: Start with initial point w_0,

$$\text{Sequence:} \quad w_{t+1} = w_t - \alpha_t \, \nabla_w \phi.$$

Stepsize and stopping criteria:

- α_t is the **stepsize** $\rightarrow$ has to be chosen sufficiently small, such that $f(x_{t+1}) < f(x_t)$.
  Find minimum of $g(\alpha)$ (**line search**)

  $$g(\alpha) := f(x_t + \alpha_t \, d_t)$$

  In practice: backtracking line search.
- Several different **stopping criteria** e.g. $\|\nabla f(x_{t+1})\| \leq \epsilon$.

Discussion of gradient descent

- requires pass over full training set in each iteration to compute gradient and to do function evaluations (computational cost $O(nd)$)
- a lot of computation is wasted in the initial iterations

Problems with Large Scale Problems:

- each iteration is very costly (if training data fits into memory) or not affordable (if training data does not fit into memory)
- high accuracy not needed

Stochastic Gradient Descent

Rewrite optimization problem as :

$$\min_{w\in\mathbb{R}^d} \phi(w) := \frac{1}{n}\sum_{i=1}^{n}\Big(L(y_i, \langle w, x_i\rangle) + n\lambda\Omega(w)\Big) := \frac{1}{n}\sum_{i=1}^{n}\phi_i(w).$$

Key idea: don't compute gradient with respect to full problem, but with respect to ϕ_i.

Stochastic gradient descent:

- variants: draw sample (x_i, y_i) with or without replacement
- $w_{k+1} = w_k - \alpha_k \nabla\phi_i(w_k)$

If training data is i.i.d., then for every fixed w and every $i \in \{1, \dots, n\}$,

$$\mathbb{E}[\nabla_w \phi_i(w)] = \nabla_w \mathbb{E}[L(Y, \langle w, X\rangle)] + n\lambda\nabla_w\Omega(w).$$

thus the gradients of ϕ_i are unbiased estimators of the true gradient of the objective.

Learning as Stochastic Optimization

Stochastic Optimization: $\min_{w \in \mathbb{R}^d} \mathbb{E}_X[f(w, X)]$. Let X_k be an i.i.d. sample from the probability measure of X, then

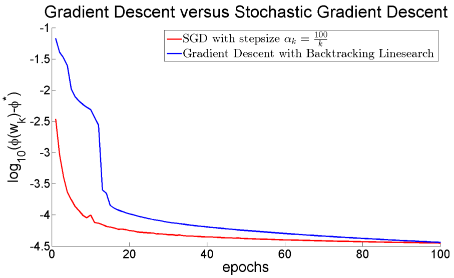
$$w_{k+1} = w_k - \alpha^k \nabla f(w_k, X_k).$$

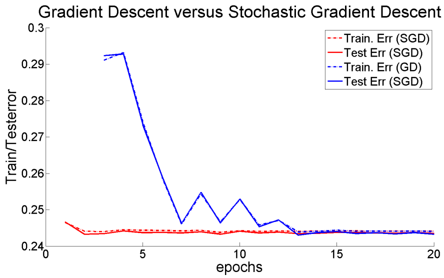
Learning Problem: $\min_{w \in \mathbb{R}^d} \mathbb{E}_{(X,Y)}[L(Y, \langle X, w \rangle)]$.

Two ways to see stochastic gradient desent for learning:

- stochastic optimization with respect to the empirical measure, that is optimization of the empirical loss $\frac{1}{n}\sum_{i=1}^n L(Y_i, \langle X_i, w \rangle)$
- the first epoch (until we have seen all samples once) can be seen as stochastic optimization of the expected loss $\mathbb{E}[L(Y, \langle X, w \rangle)]$.

Gradient Descent versus SGD





Comparison of Gradient Descent (GD) and Stochastic Gradient Descent (SGD), Problem: $n \approx 480000$ and $d = 55$, we fit logistic loss without regularizer.

Extensions

- **SGD with momentum:** keep track of direction of previous gradients

$$v_k = \eta_k v_{k-1} - \alpha_k \nabla \phi_i(w_k), \quad w_{k+1} = w_k + v_k.$$

  related: Stochastic Average Gradient Descent (SAG) (2012), requires $O(nd)$ memory.
- **SGD with averaging:** typical, average iterates $w^* = \frac{1}{K}\sum_{k=1}^{K} w_k$.
  $\Longrightarrow$ improved convergence rates compared to vanilla SGD
- **proximal methods:** $w_{k+1} = \arg\min_{w \in \mathbb{R}^d} \phi_i(w) + \frac{1}{2\alpha_k} \|w - w_k\|_2^2$.
- a lot of current research in adaptive techniques: AdaGrad, RMSProp, Adam,...