AI505

Optimization

Optimization in Machine Learning

Marco Chiarandini

Department of Mathematics & Computer Science
University of Southern Denmark

Introduction

Large-scale machine learning represents a distinctive setting in which traditional nonlinear optimization techniques typically falter

- How do optimization problems arise in machine learning applications and what makes them challenging?
- What have been the most successful optimization methods for large-scale machine learning and why?
- What recent advances have been made in the design of algorithms and what are open questions in this research area?

Two Case Studies

- Logistic regression or support vector machines
  convex optimization problems

- deep neural networks
  highly nonlinear and nonconvex problems

Text Classification via Convex Optimization

Task: determining whether a text document is one that discusses politics.

- set of examples $\{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_n, y_n)\}$, where for each $i \in \{1, \ldots, n\}$
  $\boldsymbol{x}_i$ represents the features of a text document (e.g., the words it includes)
  y_i is a label indicating whether the document belongs ($y_i = 1$) or not ($y_i = -1$) to a particular class.

- h prediction function

- measure performance: count how often the program prediction $h(\boldsymbol{x}_i)$ differs from the correct prediction y_i.

- minimize **empirical risk misclassification**

$$R_n(h) = \frac{1}{n} \sum_{i=1}^{n} \mathbb{I}[h(\boldsymbol{x}_i) \neq y_i], \qquad \text{where} \quad \mathbb{I}[A] = \begin{cases} 1 & \text{if } A \text{ is true,} \\ 0 & \text{otherwise} \end{cases}$$

Text Classification via Convex Optimization

Choosing between prediction functions belonging to a given class by comparing them using cross-validation procedures that involve splitting the examples into three disjoint subsets:

- a training set, optimizing the choice of h by minimizing R_n
- a validation set, generalized performance of each of these remaining candidates is then estimated using the validation set, the best performing of which is chosen as the selected function.
- a testing set, only used to estimate the generalized performance of this selected function

Formalization

- feature vector $\boldsymbol{x} \in \mathbb{R}^d$ whose components are associated with a prescribed set of vocabulary words; $\|\boldsymbol{x}\| = 1$
- $h(\boldsymbol{x}; \boldsymbol{w}, \tau) = \boldsymbol{w}^T \boldsymbol{x} - \tau$, $\boldsymbol{w} \in \mathbb{R}^d$ and $\tau \in \mathbb{R}^d$
- $\text{sign}(h(\boldsymbol{x}; \boldsymbol{w}, \tau)$ discountinuous
- continuous approximation through a loss function that measures a cost for predicting h when the true label is y;
  e.g., one may choose a **log-loss function** of the form

$$L(h, y) = \log(1 + \exp -hy).$$

$$\min_{(\boldsymbol{w}, \tau) \in \mathbb{R}^d \times \mathbb{R}} L(h(\boldsymbol{x}_i; \boldsymbol{w}, \tau), y_i) + \lambda \|\boldsymbol{w}\|_2^2$$

  solve for various λ and choose on the validation set

Deep Neural Networks

Deep Neural Networks: represent hypotheses as computation graphs with tunable weights and compute the gradient of the loss function with respect to those weights in order to fit the training data.

https://playground.tensorflow.org/

Perceptual Tasks via Deep Neural Networks

- Prediction function h whose value is computed by applying successive transformations to a given input vector $x_i \in \mathbb{R}^{d_0}$.
- These transformations are made in layers. A canonical fully connected layer performs the computation

$$\boldsymbol{x}_i^{(j)} = s(W_j \boldsymbol{x}_i^{(j-1)} + b_j) \in \mathbb{R}^{d_j}$$

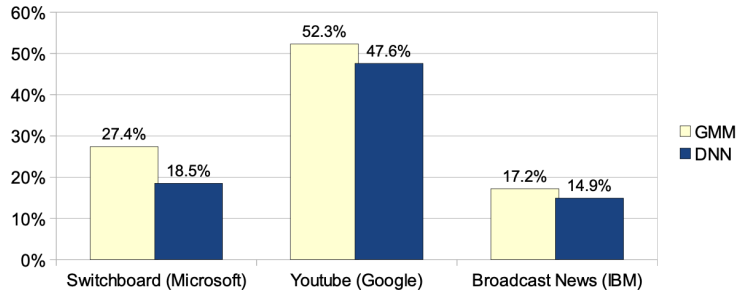
- where $\boldsymbol{x}_i^{(0)} = \boldsymbol{x}_i$, the matrix $W_j \in \mathbb{R}^{d_j \times d_{j-1}}$ and vector $b_j \in \mathbb{R}^{d_j}$ contain the jth layer parameters, and s is a component-wise nonlinear activation function
- $s(x) = 1/(1 + \exp(-x))$ and the hinge function $s(x) = \max(0, x)$ (often called a rectified linear unit (ReLU) in this context)
- $\boldsymbol{x}_i^{(J)}$ leads to the prediction function value $h(\boldsymbol{x}_i; \boldsymbol{w})$, $\boldsymbol{w} = \{(W_1, b_1), \ldots, (W_J, b_J)\}$.
- leads to highliy non-linear and non-convex:

$$\min_{\boldsymbol{w} \in \mathbb{R}^d} \frac{1}{N} \sum_{i=1}^{n} L(h(\boldsymbol{x}_i; \boldsymbol{w}), y_i)$$

- The gradient with respect to $\boldsymbol{w}$ is made of simple expressions that can be computed by passing information back through the network from the output units.
- the gradient computations for any feedforward computation graph have the same structure as the underlying computation graph.
- gradients can be computed by the chain rule and the algorithmic method of automatic differentiation
- back-propagation in deep learning is simply an application of **reverse mode differentiation**, which applies the chain rule "from the outside in"

Speech recognition

- A contemporary fully connected neural network for speech recognition typically has five to seven layers. This amounts to tens of millions of parameters to be optimized,
- the training may require up to thousands of hours of speech data (representing hundreds of millions of training examples) and weeks of computation on a supercomputer



convolutional neural networks

Convolutional neural networks (CNNs) have proved to be very effective for computer vision and signal processing tasks

ImageNet Large Scale Visual Recognition Competition (ILSVRC) with five convolutional layers and three fully connected layers

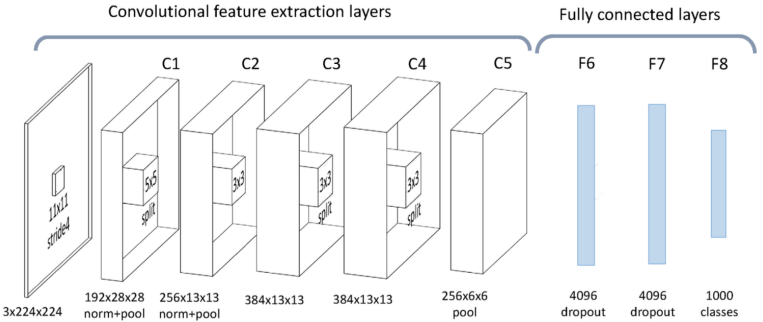


Image Recognition

- input $\boldsymbol{x}_i^{(j-1)}$ is intepreted as a multichannel image of 224×224 pixels.
- convolutional layers, wherein the parameter matrix W_j is a circulant matrix
- product $W_j \boldsymbol{x}_i^{(j-1)}$ computes the convolution of the image by a trainable filter
- activation functions are piecewise linear functions and can perform more complex operations that may be interpreted as image rectification, contrast normalization, or subsampling.
- output scores represent the odds that the image belongs to each of 1,000 categories.
- 60 million parameters
- training on a few million labeled images takes a few days on a dual GPU workstation.

Fundamentals

- Joint probability distribution function $P(x, y)$ that simultaneously represents the distribution $P(x)$ of inputs as well as the conditional probability $P(y \mid x)$ of the label y being appropriate for an input x.

- One should seek to find h that yields a small expected risk of misclassification over all possible inputs, i.e., an h that minimizes

  $$R(h) = P[h(\boldsymbol{x}) \neq y] = E[\mathbb{I}[h(\boldsymbol{x}) \neq y]],$$

  which is **variational** since we are optimizing over a set of functions (the h), and is **stochastic** since the objective function involves an expectation.

- without explicit knowledge of P the only tractable option is to construct a surrogate problem that relies solely on the examples $(\boldsymbol{x}_i, y_i)_{i=1}^n$: minimize the empirical risk

Tasks:

- how to choose the parameterized family of prediction functions $\mathcal{H}$ and

- how to determine (and find) the particular prediction function $h \in \mathcal{H}$ that is optimal.

Choice of Prediction Function

1. $\mathcal{H}$ should contain prediction functions that are able to achieve a low empirical risk over the training set, so as to avoid bias or underfitting the data. (rich family of functions or by using a priori knowledge to select a well-targeted family)

2. the gap between expected risk and empirical risk, namely, $R(h) - R_n(h)$, should be small over all $h \in \mathcal{H}$. (increases with rich family of functions)

3. $\mathcal{H}$ should be effciently solvable in the corresponding optimization problem (the richer the family of functions and/or the larger training set the more complex the problem becomes)

Choice of Prediction Function

Uniform laws of large numbers and the Hoeffding inequality gurantee that with probability at least $1 - \eta$

$$\sup_{h \in \mathcal{H}} |R(h) - R_n(h)| \leq \mathcal{O}\left(\sqrt{\frac{1}{2n} \log\left(\frac{2}{\eta}\right) + \frac{d_{\mathcal{H}}}{n} \log\left(\frac{n}{d_{\mathcal{H}}}\right)}\right)$$

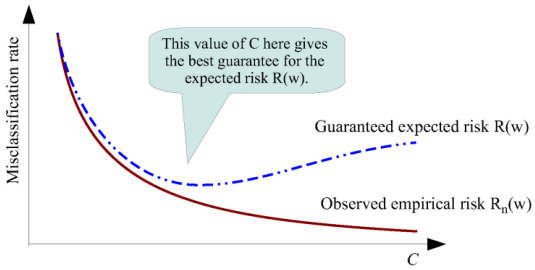
- $d_{\mathcal{H}}$ Vapnik-Chervonenkis (VC) dimension (measure of capacity of separating points)
- not the same as numer of parameters

- for a fixed $d_{\mathcal{H}}$, uniform convergence is obtained by increasing the number of training points n.
- for a fixed n, the gap can widen for larger $d_{\mathcal{H}}$.

In practice it is typically easier to estimate with cross-validation experiments.

Structural Risk Minimization

- Rather than choosing a generic family of prediction functions one chooses a structure, i.e., a collection of nested function families.
- structure can be formed as a collection of subsets of a given family $\mathcal{H}$: given a preference function Ω, choose various values of a hyperparameter C, according to each of which one obtains the subset $\mathcal{H}_C \stackrel{def}{=} \{h \in \mathcal{H} : \omega(h) \leq C\}$. ($C$ is, eg, degree of a polynomial model function, dimension of an inner layer of a DNN)



$$\min R_n(h)$$
$$\text{subject to} \quad \Omega(h) \leq C$$

Regularized empirical risk

$$R_n(h) + \lambda\Omega(h)$$

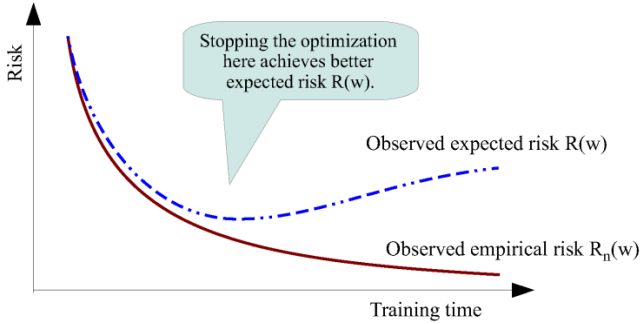
validation set is then used to estimate the expected risk corresponding to each C and to choose one.

Structural Risk Minimization

Another approach:

- employ an algorithm for minimizing R_n, but terminate the algorithm early, i.e., before an actual minimizer of R_n is found.
  The hyperparameter is played by the training time allowed

- often essential due to computational budget limitations.



Formal Optimization Problem Statements

- We do not consider a variational optimization problem over $\mathcal{H}$,
- instead we assume that the prediction function h has a fixed form and is parameterized by a real vector $\boldsymbol{w} \in \mathbb{R}^d$
- for some given $h(\cdot;\cdot) : \mathbb{R}^{d_x} \times \mathbb{R}^d \to \mathbb{R}^{d_y}$, we consider the family of prediction functions $\mathcal{H} \stackrel{def}{=} \{h(\cdot;\boldsymbol{w}) : \boldsymbol{w} \in \mathbb{R}^d\}$
- aim to find $h \in \mathcal{H}$ that minimizes a given **loss function** $L : \mathbb{R}^{d_x} \times \mathbb{R}^d \to \mathbb{R}^{d_y}$, $L(h(x;\boldsymbol{w}), y)$
- Ideally, the expected loss is defined over **any** input-output pair. Assuming probability distribution $P(\boldsymbol{x}, y)$ represents the true input-output relationship:

$$R(w) = \int_{\mathbb{R}^{d_x} \times \mathbb{R}^{d_y}} L(h(x;w), y) dP(x,y) = E[L(h(x;w), y)] \qquad \text{Expected Risk}$$

Formal Optimization Problem Statements

- In practice, one seeks the solution of a problem that involves an estimate of the expected risk R.

- In supervised learning, we have access (either all-at-once or incrementally) to a set of $n \in \mathbb{N}$ independently drawn input-output samples $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^n \subseteq \mathbb{R}^{d_x} \times \mathbb{R}^{d_y}$, with which we define the **empirical risk function** $R_n : \mathbb{R}^d \to \mathbb{R}$ by

$$R_n(\boldsymbol{w}) \stackrel{def}{=} \frac{1}{n} \sum_{i=1}^{n} L(h(\boldsymbol{x}_i; \boldsymbol{w}), y_i) \qquad \text{Empirical Risk}$$

(note that before we used misclassification error while now L.

Simplified Notation

Let ξ be a random seed or the realization of a single (or a set of) sample $(\boldsymbol{x}, y)$.
For a given $(\boldsymbol{w}, \xi)$ let $f(\boldsymbol{w}; \xi)$ be the composition of the loss function L and the prediction function h
Then:

$$R(\boldsymbol{w}) = \mathbb{E}_{\xi}[f(\boldsymbol{w}; \xi)] \qquad \text{Expected Risk}$$

Let $\{\xi_{[i]}\}_{i=1}^{n}$ be realizations of ξ corresponding to $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^{n}$ and $f_i(\boldsymbol{w}) \stackrel{def}{=} f(\boldsymbol{w}; \xi_{[i]})$
Then:

$$R_n(\boldsymbol{w}) \stackrel{def}{=} \frac{1}{n} \sum_{i=1}^{n} f_i(\boldsymbol{w}) \qquad \text{Empirical Risk}$$

Stochastic vs Batch Optimization Methods

Reduction to minimizing R_n, with $\boldsymbol{w}_0 \in \mathbb{R}^d$ given (deterministic problem)

Stochastic Approach: Stochastic Gradient (Robbins and Monro, 1951)

$$\boldsymbol{w}_{k+1} \leftarrow \boldsymbol{w}_k - \alpha_k \nabla f_{i_k}(\boldsymbol{w}_k)$$

i_k is chosen randomly from $\{1, \ldots, n\}$, $\alpha_k > 0$.

- very cheap iteration only on one sample.
- $\{\boldsymbol{w}_k\}$ is a stochastic process determined by the random sequence $\{i_k\}$.
- the direction might not always be a descent but if it is a descent direction in **expectation**, then the sequence $\{\boldsymbol{w}_k\}$ can be guided toward a minimizer of R_n.

Batch Approach: batch gradient, steepest descent, full gradient method:

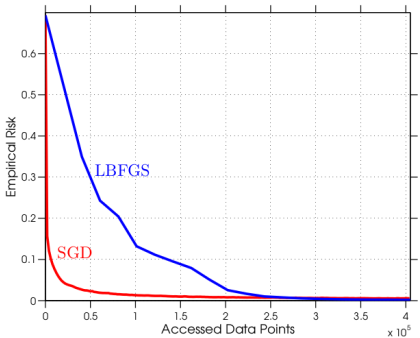
$$\boldsymbol{w}_{k+1} \leftarrow \boldsymbol{w}_k - \alpha_k \nabla R_n(\boldsymbol{w}_k) = \boldsymbol{w}_k - \frac{\alpha_k}{n} \sum_{i=1}^{n} \nabla f_i(\boldsymbol{w}_k)$$

- more expensive
- can use all deterministic gradient-based optimization methods
- the sum structure opens up to parallelization

Analogues in simulation: stochastic approximation (SA) and sample average approximation (SAA)

Stochastic Gradient

- In case of redundancy using all of the sample data in every iteration is inefficient
- Comparison of the performance of a batch L-BFGS method on number of evaluations of a sample gradient $\nabla f_{i_k}(\boldsymbol{w}_k)$. Each set of n consecutive accesses is called an **epoch**.
- The batch method performs only one step per epoch while SG performs n steps per epoch.



the fast initial improvement achieved by SG, followed by a drastic slowdown after 1 or 2 epochs, is common in practice

SG more sensitive to α_k and starting point

if more epochs, batch may become better

Theoretical Motivations

- a batch approach can minimize R_n at a fast rate; e.g., if R_n is strongly convex. A batch gradient method, then there exists a constant $\rho \in (0,1)$ such that, for all $k \in \mathbb{N}$, the training error follows **linear convergence**

$$R_n(\boldsymbol{w}_k) - R_n^* \leq \mathcal{O}(\rho^k),$$

- rate of convergence of a basic stochastic method is slower than for a batch gradient; e.g., if R_n is strictly convex and each i_k is drawn uniformly from $\{1, \ldots, n\}$, then for all $k \in \mathbb{N}$, SG satisfies the **sublinear convergence property**

$$\mathbb{E}[R_n(\boldsymbol{w}_k) - R_n^*] = \mathcal{O}(1/k).$$

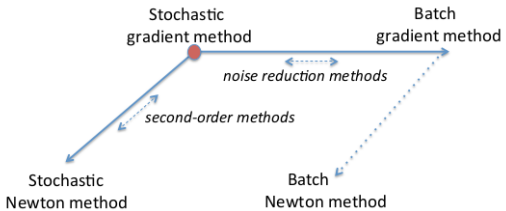
neither the per-iteration cost nor the right-hand side depends on the sample set size n

- in a stochastic optimization setting, SG yields for the expected risk the same convergence rate once substituted $\nabla f_{i_k}(\boldsymbol{w}_k)$ replaced by $\nabla f(\boldsymbol{w}_k; \xi_k)$ with each ξ_k drawn independently according to the distribution P

$$\mathbb{E}[R(\boldsymbol{w}_k) - R^*] = \mathcal{O}(1/k).$$

If $n >> k$ up to iteration k minizming R_n same as minimizing R

Beyond SG: Noise Reduction and Second-Order Methods



- on horizontal axis methods that try to improve rate of convergence
- on vertical axis, methods that try to overcome non-linearity and ill-conditioning

Minibatch Approach small subset of samples, call it $\mathcal{S}_k \subseteq \{1, \ldots, n\}$, chosen randomly in each iteration:

$$\boldsymbol{w}_{k+1} \leftarrow \boldsymbol{w}_k - \frac{\alpha_k}{|\mathcal{S}_k|} \sum_{i \in \mathcal{S}} \nabla f_i(\boldsymbol{w}_k)$$

due to the reduced variance of the stochastic gradient estimates, the method is easier to tune in terms of choosing the stepsizes $\{\alpha_k\}$.
dynamic sample size and gradient aggregation methods, both of which aim to improve the rate of convergence from sublinear to linear