Optimization Methods for Large-Scale Machine Learning

V. Estellers

WS 2017

What this lectures are about

The optimization problems that result from training a large-scale machine learning model have characteristics that make the stochastic gradient (SG) method more effective than conventional gradient-based nonlinear optimization techniques.

1. Characteristic of optimization of large-scale machine learning models
2. Stochastic gradient algorithm
3. Analysys of SG algorithm
4. Improved SG convergence with noise-reduction techniques
5. Improved SG convergence with second-order derivatives

References

The lectures are organized following:
Bottou, L., Curtis, F. E., and Nocedal, J. *Optimization Methods for Large-Scale Machine Learning.* 2016.http://arxiv.org/abs/1606.04838
We will also cover some material from:

1. Gower, R. M., Roux, N. Le, and Bach, F. *Tracking the gradients using the Hessian: A new look at variance reducing stochastic methods.*, 2017.
2. Roux, N. Le, Schmidt, M., and Bach, F. *A Stochastic Gradient Method with an Exponential Convergence Rate for Strongly-Convex Optimization with Finite Training Sets.* 2012.

Optimization Problems in Machine Learning

We illustrate how optimization problems arise in machine learning and what makes them challenging with two case studies:

1. linear regressor with bag-of-words features for text classification
2. open-ended deep neural network for speech and image recognition.

Both problems have some common characteristics:

Large-scale: models described by a large number of parameters.

Stochastic: models designed to make decisions on unseen data..

They differ in the optimization problem: (1) convex, (2) nonconvex.

Text Classification via Convex Optimization

Text classification: assigning a predefined class to a natural language text based on its contents. For example, determine if a text discusses politics.

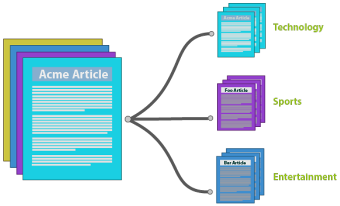


Fig.: http://blog.thedigitalgroup.com/rajendras/2015

Given a set of examples $\{(x_1, y_1), \ldots, (x_n, y_n)\}$, where

- feature vector x_i of a text document (e.g., the words it includes).
- scalar label y_i indicating if the document belongs $(y_i = 1)$ or not $(y_i = 1)$ to a particular class.

Construct a classifier that predicts the class of an unseen text.

First Solution: Minimizing Empirical Risk

Design a prediction function h s.t. $h(x)$ predicts the text document.

Performance measure: how often $h(x_i)$ differs from the prediction y_i.

Search h that minimizes the frequency of observed misclassifications:

$$R_n(h) = \frac{1}{n}\sum_{i=1}^{n} \mathbb{1}[h(x_i) \neq y_i], \quad \text{where} \quad \mathbb{1}[A] = \begin{cases} 1 & \text{if } A \text{ is true,} \\ 0 & \text{otherwise.} \end{cases} \tag{1.1}$$

R_n is the empirical risk of misclassification.

Minimizing Empirical Risk is Not Enough

Rote memorization with

$$h^{\text{rote}}(x) = \begin{cases} y_i & \text{if } x = x_i \text{ for some } i \in \{1, \dots, n\}, \\ \pm 1 & \text{(arbitrarily) otherwise.} \end{cases} \tag{1.2}$$

minimizes the empirical risk but has no guarantees on unseen documents.

The prediction function should generalizes the concepts learned from the examples. To this goal, we choose

- parametric functions satisfying certain smoothness conditions
- use cross-validation to choosing between classes of prediction functions

Minimizing Expected Risk with Cross-validation

Cross-validation minimizes the expected risk by splitting examples into:

- training set to optimize the parameters of h by minimizing R_n. The selects a candidate for each class of parametric functions $h_1, \ldots, h_k$
- validation set to estimate the performance of $h_1, \ldots, h_k$. This selects the best candidate h^*
- testing set to estimate the performance of h^*

Cross-validation has shown the success of **bag-of-words** approach for text classification.

Linear Regression with Bag-of-Words Features

Bag-of-Words features:

- represents a text document by a feature vector $x \in \mathbb{R}^d$, where each component measures the appearance of a specific word.
- very sparse vectors of high-dimensionality.

Affine prediction function classifies the documents:

$$h(x; w, \tau) = w^T x - \tau \tag{1.3}$$

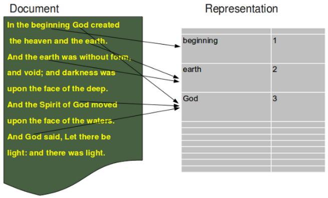

Document
In the beginning God created
the heaven and the earth.
And the earth was without form,
and void; and darkness was
upon the face of the deep.
And the Spirit of God moved
upon the face of the waters.
And God said, Let there be
light: and there was light.
Representation
beginning
1
earth
2
God
3

Fig.: https://www.python-kurs.eu/text_klassifikation_python.php

Optimization of the Model

Finding w, τ that minimize the empirical risk of misclassification

$$R_n(h) = \frac{1}{n} \sum_{i=1}^{n} \mathrm{sign}(-h(x_i; w, \tau) \cdot y_i) \tag{1.4}$$

is difficult because the sign is discontinuous, takes discrete values, and results in a combinatorial problem. For this reason, we approximate it by a continuous loss function that we can minimize effective like

$$\ell(h, y) = \log(1 + \exp(h(x_i)y)). \tag{1.5}$$

Classes of prediction functions h_λ are determined by a regularization term

$$\min_{(w,\tau)\in\mathbb{R}^{d+1}} \frac{1}{n} \sum_{i=1}^{n} \log(1 + \exp(h(x_i)y_i)) + \frac{\lambda}{2}\|w\|^2. \tag{1.6}$$

Optimizing the model parameters with various $\lambda_1, \ldots, \lambda_k$ on the training set gives the candidate solution $h_{\lambda_1}, \ldots, h_{\lambda_k}$. The final solution is the candidate with best performance on the validation set.

Perceptual Tasks via Deep Networks

Deep/Convolutional neural networks have recently achieved spectacular success on perceptual problems such as speech and image recognition.

They are essentially the same types of networks from the 90s, but their successes is now possible due to the availability of larger datasets and computational resources.

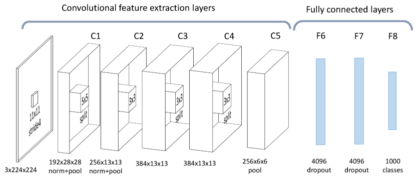


Fig.: Architecture for image recognition. The 2012 ILSVRC winner consists of eight layers: each layer performs a linear transformation followed by nonlinear transformations.

Neural Networks

DNN/CNNs construct a 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.

$$x_i^{(j)} = s(W_j\, x_i^{(j-1)} + b_j) \in \mathbb{R}^{d_j}, \tag{1.7}$$

where $x_i^{(0)} = x_i$ and

- $x_i^{(j)}$ is the input vector to layer j
- j-th layer parameters: matrix $W_j \in \mathbb{R}^{d_j \times d_{j-1}}$ and vector $b_j \in \mathbb{R}^{d_j}$
- s componentwise nonlinear activation/pooling function.

Deep Neural Networks

Neural Networks use simple activation functions, like the sigmoid or the rectified linear unit (ReLU)

$$s(x) = 1/(1 + \exp(-x)) \qquad\qquad s(x) = \max\{0, x\}.$$

CNNs are networks where layers have

circulant matrices W_j, s.t. $W_j x_i^{(j1)}$ is an image convolution.

activation functions rectify, normalize, or subsample images.

The output vector $x_i^{(J)}$ is the prediction function value $h(x_i; w)$, where $w = \{(W_1, b_1), \ldots, (W_J, b_J)\}$ collects the parameters of all the layers.

Optimization of Deep Neural Networks

The optimization of DNN/CNN over the training set $\{(x_1, y_1), \ldots, (x_n, y_n)\}$ with a loss function ℓ define the problem

$$\min_{w \in \mathbb{R}^d} \ \frac{1}{n} \sum_{i=1}^{n} \ell(h(x_i; w), y_i). \tag{1.8}$$

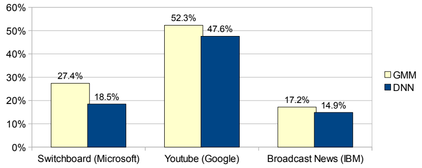
This problem is nonconvex. Finding a global optimum is intractable and we look for approximate solutions with gradient-based methods.

The gradient of the objective function of (1.8) can be computed efficiently by the chain rule (back propagation).

Convolutional neural networks

The training process of DNNs and CNNs requires extreme care to overcome the difficulties of large, nonlinear and nonconvex problems:

1. initialize the optimization process with a good starting point
2. monitor its progress to correct conditioning issues as they appear (vanishing gradients).



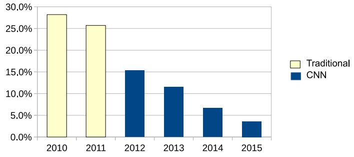


Fig.: Left:Word error rates reported by three different research groups on three standard speech recognition benchmarks. Right: Historical *top5* error rate of the annual winner of the ImageNet image classification challenge.

Formulation of a Supervised Classification Problem

Classification: choose a prediction function from an input space $\mathscr{X}$ to an output space $\mathscr{Y}$

$$h : \mathscr{X} \to \mathscr{Y}$$

s.t., given $x \in \mathscr{X}$, $h(x)$ offers an accurate prediction about the output y.

Supervised: h that generalizes the properties meaningful to determine y from x that can be learned from input-output examples $\{(x_i, y_i)\}_{i=1}^{n}$.

Problem: avoid rote memorization by choosing a prediction function h that minimizes a risk measure over a family of prediction functions $\mathscr{H}$.

Expected Risk instead of Empirical Risk

Let $\{(x_i, y_i)\}_{i=1}^n$ be samples from a joint probability distribution function $P(x, y)$. Rather than finding h that minimizes the empirical risk

$$R_n(h) = \frac{1}{n}\sum_{i=1}^{n} \mathbb{1}[h(x_i) = y_1] \qquad \mathbb{1}[A] = \begin{cases} 1 & \text{if A is true} \\ 0 & \text{otherwise} \end{cases}, \tag{1.9}$$

we find h that minimizes the expected misclassification risk over all possible inputs, i.e., an h that minimizes

$$R(h) = \mathbb{P}[h(x) \neq y] = \mathbb{E}[\mathbb{1}[h(x) \neq y]], \tag{1.10}$$

Stochastic problem (objective is an expectation) is substituted by a surrogate problem constructed from $\{(x_i, y_i)\}_{i=1}^n$ as we do not know P.

Choice of Prediction Function Family

We choose the family of functions $\mathscr{H}$ with three goals in mind:

1. $\mathscr{H}$ should contain functions that achieve a low empirical risk to avoid underfitting the data. $\Rightarrow$ select a rich family of functions
2. $\mathscr{H}$ should be selected to make the optimization problem solvable
3. $R(h) - R_n(h)$ should be small over all $h \in \mathscr{H}$. This gap might increase when $\mathscr{H}$ becomes too rich and overfits the training data.

Gap Between Expected and Empirical Risk

When the expected risk represents a misclassification probability, with probability at least $1-\eta$,

$$\sup_{h\in\mathscr{H}} |R(h) - R_n(h)| \le \mathscr{O}\left(\sqrt{\frac{1}{2n}\log\left(\frac{2}{\eta}\right) + \frac{d_{\mathscr{H}}}{n}\log\left(\frac{n}{d_{\mathscr{H}}}\right)}\right). \quad (1.11)$$

- $d_{\mathscr{H}}$: Vapnik-Chervonenkis dimension measures the capacity of $\mathscr{H}$
- fixed $d_{\mathscr{H}}$, the gap decreases by increasing number of examples (n).
- fixed n, the gap can widen for larger $d_{\mathscr{H}}$ (richer function families).

Bound (1.11) is not used in practice because it is easier to estimate the gap with cross-validation than calculate the VC dimension of $\mathscr{H}$.

Structural Risk Minimization

Structural risk minimization: technique for choosing a prediction function.
Consider a nested families of function parametrized by function Ω

$$H_C = \{h \in \mathscr{H} \ : \ \Omega(h) \leq C\} \Rightarrow H_C \subset H_D \text{ for } C < D$$

1. Increasing C reduces the R_n because it enlarges the family of functions we can optimize over.
2. For large C, the $R_n - R$ increases because the prediction function overfits the training data.

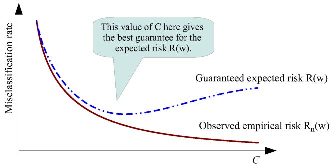


Fig.: The optimal empirical risk decreases when C increases. $|R(w) - R_n(w)|$ is bounded above by a quantity that increases with C. The value of C that offers the best guarantee on the expected risk increases with n.

In the following, we consider the problem...

Assume that the prediction function h is parameterized by a real vector $w \in \mathbb{R}^d$. This vector defines our optimization variable and the family of prediction functions

$$\mathscr{H} = \{h(\cdot; w)\colon \ \mathbb{R}^d \times \mapsto \mathbb{R}^{d_y} \mid w \in \mathbb{R}^d\}.$$

Given a loss function $\ell\colon \mathbb{R}^{d_y} \times \mathbb{R}^{d_y} \mapsto \mathbb{R}$ that measures the loss associated with the prediction $h(x; w)$ when the true label is y with $\ell(h(x; w), y)$, we define:

ξ: random variable that represents a sample or a set of samples $\{(x_i, y_i)\}_{i \in S}$.

$f(w; \xi) = \ell(h(w; \xi), y)$: the loss incurred for a given (w, ξ)

Expected and Empirical Risk

Let $P(x, y)$ be the probability distribution between inputs and outputs, we define **expected risk** by

$$R(w) = \int_{\mathbb{R}^{d_x} \times \mathbb{R}^{d_y}} \ell(h(x; w), y) \mathrm{d}P(x, y) = \mathbb{E}[\ell(h(x; w), y)] = \mathbb{E}[f(w; \xi)]$$

To minimize the expected risk, we need complete information about P. As this is not possible, we minimize the **empirical risk**

$$R_n(w) = \frac{1}{n} \sum_{i=1}^{n} \ell(h(x_i; w), y_i) = \frac{1}{n} \sum_{i=1}^{n} f(w; \xi[i])$$

that estimates the expected risk (in supervised classification) from n independently drawn input-output samples $\{\xi[i]\}_{i=1}^{n} = \{(x_i, y_i)\}_{i=1}^{n}$.

Stochastic Optimization for Empirical Risk Minimization

The stochastic gradient method (SG) minimizes the empirical risk R_n with the sequence:

$$w_{k+1} = w_k - \alpha_k \nabla f_{i_k}(w_k) \quad \forall k \in \mathbb{N}, \tag{2.12}$$

where w_1 is given, α_k is a positive stepsize, and i_k is chosen randomly. Characteristics:

1. Cheap iterations that only computate one gradient $\nabla f_{i_k}(w_k)$
2. The sequence is not determined uniquely by R_n, w_1, and stepsizes, but depends also on the random sequence $\{i_k\}$.
3. $-\nabla f_{i_k}(w_k)$ might not be a descent direction from w_k.

Batch Optimization for Empirical Risk Minimization

A batch approach minimizes the Empirical Risk directly. The simplest steepest descent or gradient method defines the sequence:

$$w_{k+1} = w_k - \alpha_k \nabla R_n(w_k) = w_k - \alpha_k \frac{1}{n} \sum_{i=1}^{n} \nabla f_i(w_k) \quad \forall k \in \mathbb{N}, \quad (2.13)$$

Characteristics:

1. Computing $\alpha_k \nabla R_n(w_k)$ is more expensive than $\alpha_k \nabla f_{i_k}(w_k)$ in SG.
2. By iterating over all samples, batch methods compute better steps.
3. It can use (quasi) Newton methods to speed up optimization of R_n.
4. The sum structure of R_n allows parallel or distributed updates.

Typical Performance of Stochastic and Batch Methods

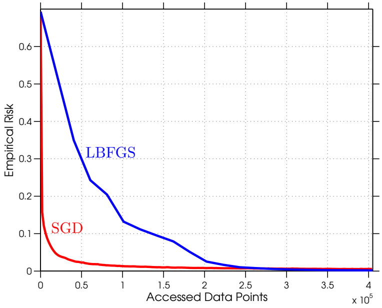


Fig.: Empirical risk R_n as a function of the number of accessed data points (ADP) for a batch L-BFGS method and the SG method on a binary classification problem with a logistic loss objective and the RCV1 dataset.

Stochastic vs Batch Methods

SG is used in machine learning when cannot afford to iterate over all samples to compute the next iterate.

SG uses the samples more efficient than a batch method. Intuitively:

-Consider a training set S with ten copies of a set S_s.

-In a batch approach, the iterations that use S as training set are ten times more expensive than iteration that only use one copy of S_s.

-In the SG method, the iterations using S and S_s as training sets cost the same.

-In reality, a training set does not consist of exact duplicates of sample data, but it has enough redundancy to make using all of the samples in every iteration inefficient.

Next week: Stochastic vs Batch Methods

Let R_n^* be the minimal value of R_n, then if R_n is strongly convex

- the error of a batch gradient method satisfies

$$R_n(w_k) - R_n^*| \leq \mathcal{O}(\rho^k), \quad \rho \in (0,1).$$

The number of iterations where the training error is above ϵ is proportional to $\log(\frac{1}{\epsilon})$, and the cost of ϵ-optimality is $\mathcal{O}(n \log(\frac{1}{\epsilon}))$.

- the SG error for i_k is drawn uniformly from $\{1, \ldots, n\}$ is

$$\mathbb{E}[Rn(w_k)R_n^*] = \mathcal{O}(\frac{1}{k}) \tag{2.14}$$

As it does not depend on n, the cost of ϵ-optimality is $\mathcal{O}(\frac{1}{\epsilon})$.

The SG cost $\mathcal{O}(\frac{1}{\epsilon})$ is smaller than the batch cost $\mathcal{O}(n \log(\frac{1}{\epsilon}))$ if n is large.

Next week: Stochastic vs Batch Methods

SG avoids overfitting in the sense that the minimizer of the empirical risk found by SG has some minimization guarantees on the expected risk.

By applying the SG iteration with $\nabla f(w_k; x_{i_k})$ replaced by $\nabla f(w_k; \xi_k)$ with each ξ_k drawn independently according to the distribution P,

$$\mathbb{E}[R(w_k) - R^*] = \mathcal{O}(\frac{1}{k}). \tag{2.15}$$

This is again a sublinear rate, but on the expected risk.

Other Methods

A mini-batch approach chooses a subset of samples $S_k \subset \{1, \ldots, n\}$ randomly in each iteration to improve the gradient estimate as follows:

$$w_{k+1} = w_k - \frac{\alpha_k}{|S_k|} \sum_{i \in S_k} \nabla f_i(w_k) \quad \forall k \in \mathbb{N}, \tag{2.16}$$

This allows some degree of parallelization and reduces the variance of the stochastic gradient estimates.

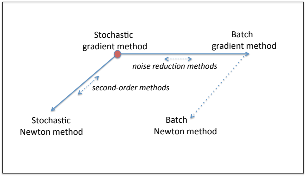


Fig.: Schematic of a two-dimensional spectrum of optimization methods for machine learning. The horizontal axis represents methods designed to control stochastic noise; the second axis, methods that deal with ill conditioning.