13 The Tradeoffs of Large-Scale Learning

Léon Bottou leon@bottou.org
NEC Laboratories of America
Princeton, NJ, USA

Olivier Bousquet olivier.bousquet@m4x.org
Google
Zurich, Switzerland

This chapter develops a theoretical framework that takes into account the effect of approximate optimization on learning algorithms. The analysis shows distinct tradeoffs for the case of small-scale and large-scale learning problems. Small-scale learning problems are subject to the usual approximation–estimation tradeoff. Large-scale learning problems are subject to a qualitatively different tradeoff involving the computational complexity of the underlying optimization algorithm in non-trivial ways. For instance, a mediocre optimization algorithm, stochastic gradient descent, is shown to perform very well on large-scale learning problems.

13.1 Introduction

The computational complexity of learning algorithms has seldom been taken into account by the learning theory. Valiant (1984) states that a problem is "learnable" when there exists a "probably approximately correct" learning algorithm *with polynomial complexity*. Whereas much progress has been made on the statistical aspect (e.g., Vapnik, 1982; Boucheron et al., 2005; Bartlett and Mendelson, 2006), very little has been said about the complexity side of this proposal (e.g., Judd, 1988).

Computational complexity becomes the limiting factor when one envisions

large amounts of training data. Two important examples come to mind:

- Data mining exists because competitive advantages can be achieved by analyzing the masses of data that describe the life of our computerized society. Since virtually every computer generates data, the data volume is proportional to the available computing power. Therefore, one needs learning algorithms that scale roughly linearly with the total volume of data.
- Artificial intelligence attempts to emulate the cognitive capabilities of human beings. Our biological brains can learn quite efficiently from the continuous streams of perceptual data generated by our senses, using limited amounts of sugar as a source of power. This observation suggests that there are learning algorithms whose computing time requirements scale roughly linearly with the total volume of data.

This chapter develops the ideas initially proposed by Bottou and Bousquet (2008). Section 13.2 proposes a decomposition of the test error where an additional term represents the impact of approximate optimization. In the case of small-scale learning problems, this decomposition reduces to the well-known tradeoff between approximation error and estimation error. In the case of large-scale learning problems, the tradeoff is more complex because it involves the computational complexity of the learning algorithm. Section 13.3 explores the asymptotic properties of the large-scale learning tradeoff for various prototypical learning algorithms under various assumptions regarding the statistical estimation rates associated with the chosen objective functions. This part clearly shows that the best optimization algorithms are not necessarily the best learning algorithms. Maybe more surprisingly, certain algorithms perform well regardless of the assumed rate of the statistical estimation error. Section 13.4 reports experimental results supporting this analysis.

13.2 Approximate Optimization

13.2.1 Setup

Following Duda and Hart (1973) and Vapnik (1982), we consider a space of input-output pairs $(x, y) \in \mathcal{X} \times \mathcal{Y}$ endowed with a probability distribution $P(x, y)$. The conditional distribution $P(y|x)$ represents the unknown relationship between inputs and outputs. The discrepancy between the predicted output $\hat{y}$ and the real output y is measured with a loss function $\ell(\hat{y}, y)$. Our

benchmark is the function f^* that minimizes the expected risk

$$E(f) = \int \ell(f(x), y)\, dP(x, y) = \mathbb{E}\,[\ell(f(x), y)],$$

that is,

$$f^*(x) = \arg\min_{\hat{y}} \mathbb{E}\,[\,\ell(\hat{y}, y)|\, x].$$

Although the distribution $P(x, y)$ is unknown, we are given a sample $\mathcal{S}$ of n independently drawn training examples (x_i, y_i), $i = 1 \ldots n$. We define the empirical risk

$$E_n(f) = \frac{1}{n} \sum_{i=1}^{n} \ell(f(x_i), y_i) = \mathbb{E}_n[\ell(f(x), y)].$$

Our first learning principle is choosing a family $\mathcal{F}$ of candidate prediction functions and finding the function $f_n = \arg\min_{f \in \mathcal{F}} E_n(f)$ that minimizes the empirical risk. Well-known combinatorial results (e.g., Vapnik, 1982) support this approach, provided that the chosen family $\mathcal{F}$ is sufficiently restrictive. Since the optimal function f^* is unlikely to belong to the family $\mathcal{F}$, we also define $f^*_{\mathcal{F}} = \arg\min_{f \in \mathcal{F}} E(f)$. For simplicity, we assume that f^*, $f^*_{\mathcal{F}}$, and f_n are well defined and unique.

We can then decompose the excess error as

$$\begin{aligned} \mathcal{E} &= \mathbb{E}\,[E(f^*_{\mathcal{F}}) - E(f^*)] \;+\; \mathbb{E}\,[E(f_n) - E(f^*_{\mathcal{F}})] \\ &= \mathcal{E}_{\text{app}} + \mathcal{E}_{\text{est}}, \end{aligned} \tag{13.1}$$

where the expectation is taken with respect to the random choice of training set. The *approximation error* $\mathcal{E}_{\text{app}}$ measures how closely functions in $\mathcal{F}$ can approximate the optimal solution f^*. The *estimation error* $\mathcal{E}_{\text{est}}$ measures the effect of minimizing the empirical risk $E_n(f)$ instead of the expected risk $E(f)$. The estimation error is determined by the number of training examples and by the capacity of the family of functions (Vapnik, 1982). Large families[1] of functions have *smaller approximation errors* but lead to *higher estimation errors*. This tradeoff has been extensively discussed in the literature (Vapnik, 1982; Boucheron et al., 2005) and has led to excess errors that scale between the inverse and the inverse square root of the number of

1. We often consider nested families of functions of the form $F_c = \{f \in \mathcal{H},\ \Omega(f) \leq c\}$. Then, for each value of c, function f_n is obtained by minimizing the regularized empirical risk $E_n(f) + \lambda\Omega(f)$ for a suitable choice of the Lagrange coefficient λ. We can then control the estimation-approximation tradeoff by choosing λ instead of c.

examples (Zhang, 2004; Steinwart and Scovel, 2005).

13.2.2 Optimization Error

Finding f_n by minimizing the empirical risk $E_n(f)$ is often a computationally expensive operation. Since the empirical risk $E_n(f)$ is already an approximation of the expected risk $E(f)$, it should not be necessary to carry out this minimization with great accuracy. For instance, we could stop an iterative optimization algorithm long before its convergence.

Let us assume that our minimization algorithm returns an approximate solution $\tilde{f}_n$ such that $E_n(\tilde{f}_n) < E_n(f_n) + \rho$ where $\rho \geq 0$ is a predefined tolerance. An additional term $\mathcal{E}_{\text{opt}} = \mathbb{E}\big[E(\tilde{f}_n) - E(f_n)\big]$ then appears in the decomposition of the excess error $\mathcal{E} = \mathbb{E}\big[E(\tilde{f}_n) - E(f^*)\big]$:

$$\begin{aligned} \mathcal{E} &= \mathbb{E}\left[E(f^*_{\mathcal{F}}) - E(f^*)\right] + \mathbb{E}\left[E(f_n) - E(f^*_{\mathcal{F}})\right] + \mathbb{E}\big[E(\tilde{f}_n) - E(f_n)\big] \\ &= \mathcal{E}_{\text{app}} + \mathcal{E}_{\text{est}} + \mathcal{E}_{\text{opt}}. \end{aligned} \tag{13.2}$$

We call this additional term the *optimization error*. It reflects the impact of the approximate optimization on the generalization performance. Its magnitude is comparable to ρ (see section 13.3.1).

13.2.3 The Approximation–Estimation–Optimization Tradeoff

This decomposition leads to a more complicated compromise. It involves three variables and two constraints. The constraints are the maximal number of available training examples and the maximal computation time. The variables are the size of the family of functions $\mathcal{F}$, the optimization accuracy ρ, and the number of examples n. This is formalized by the following optimization problem:

$$\min_{\mathcal{F},\rho,n} \quad \mathcal{E} = \mathcal{E}_{\text{app}} + \mathcal{E}_{\text{est}} + \mathcal{E}_{\text{opt}} \quad \text{subject to} \left\{ \begin{array}{rcl} n & \leq & n_{\max} \\ T(\mathcal{F},\rho,n) & \leq & T_{\max} \end{array} \right. \tag{13.3}$$

The number n of training examples is a variable because we could choose to use only a subset of the available training examples in order to complete the optimization within the alloted time. This happens often in practice. Table 13.1 summarizes the typical evolution of the quantities of interest as the three variables $\mathcal{F}$, n, and ρ increase.

The solution of the optimization program (13.3) depends critically on which budget constraint is active: constraint $n < n_{\max}$ on the number of examples, or constraint $T < T_{\max}$ on the training time.

Table 13.1: Typical variations when $\mathcal{F}$, n, and ρ increase

		$\mathcal{F}$	n	ρ
$\mathcal{E}_{\text{app}}$	(approximation error)	$\searrow$		
$\mathcal{E}_{\text{est}}$	(estimation error)	$\nearrow$	$\searrow$	
$\mathcal{E}_{\text{opt}}$	(optimization error)	$\cdots$	$\cdots$	$\nearrow$
T	(computation time)	$\nearrow$	$\nearrow$	$\searrow$

- We speak of a *small-scale learning problem* when (13.3) is constrained by the maximal number of examples $n_{\max}$. Since the computing time is not limited, we can reduce the optimization error $\mathcal{E}_{\text{opt}}$ to insignificant levels by choosing a ρ that is arbitrarily small. The excess error is then dominated by the approximation and estimation errors, $\mathcal{E}_{\text{app}}$ and $\mathcal{E}_{\text{est}}$. Taking $n = n_{\max}$, we recover the approximation-estimation tradeoff that is the object of abundant literature.

- We speak of a *large-scale learning problem* when (13.3) is constrained by the maximal computing time $T_{\max}$. Approximate optimization, that is, choosing $\rho > 0$, possibly can achieve better generalization because more training examples can be processed during the allowed time. The specifics depend on the computational properties of the chosen optimization algorithm through the expression of the computing time $T(\mathcal{F}, \rho, n)$.

13.3 Asymptotic Analysis

In section 13.2.2, we extended the classical approximation–estimation tradeoff by taking the optimization error into account. We gave an objective criterion to distiguish small-scale and large-scale learning problems. In the small-scale case, we recovered the classical tradeoff between approximation and estimation. The large-scale case is substantially different because it involves the computational complexity of the learning algorithm. In order to clarify the large-scale learning tradeoff with sufficient generality, this section makes several simplifications:

- We are studying upper bounds of the approximation, estimation, and optimization errors (13.2). It is often accepted that these upper bounds give a realistic idea of the actual convergence rates (Vapnik et al., 1994; Bousquet, 2002; Tsybakov, 2004; Bartlett et al., 2006). Another way to find comfort in this approach is to say that we study guaranteed convergence rates instead of the possibly pathological special cases.

- We are studying the asymptotic properties of the tradeoff when the problem size increases. Instead of carefully balancing the three terms, we write $\mathcal{E} = \mathcal{O}(\mathcal{E}_{\text{app}}) + \mathcal{O}(\mathcal{E}_{\text{est}}) + \mathcal{O}(\mathcal{E}_{\text{opt}})$ and only need to ensure that the three terms decrease with the same asymptotic rate.
- We are considering a fixed family of functions $\mathcal{F}$, and therefore avoid taking into account the approximation error $\mathcal{E}_{\text{app}}$. This part of the tradeoff covers a wide spectrum of practical realities, such as choosing models and features. In the context of this work, we do not believe we can meaningfully address this without discussing, for instance, the thorny issue of feature selection. Instead, we focus on the choice of optimization algorithm.
- Finally, in order to keep this chapter short, we consider that the family of functions $\mathcal{F}$ is linearly parameterized by a vector $w \in \mathbb{R}^d$. We also assume that x, y, and w are bounded, ensuring that there is a constant B such that $0 \leq \ell(f_w(x), y) \leq B$ and $\ell(\cdot, y)$ is Lipschitz.

We first explain how the uniform convergence bounds provide convergence rates that take the optimization error into account. Then we discuss and compare the asymptotic learning properties of several optimization algorithms.

13.3.1 Convergence of the Estimation and Optimization Errors

The optimization error $\mathcal{E}_{\text{opt}}$ depends on the optimization accuracy ρ. However, the accuracy ρ involves the empirical quantity $E_n(\tilde{f}_n) - E_n(f_n)$, whereas the optimization error $\mathcal{E}_{\text{opt}}$ involves its expected counterpart $E(\tilde{f}_n) - E(f_n)$. This section discusses the impact of the optimization error $\mathcal{E}_{\text{opt}}$ and of the accuracy ρ on generalization bounds that leverage the uniform convergence concepts pioneered by Vapnik and Chervonenkis (e.g., Vapnik, 1982).

Following Massart (2000), in the following discussion we use the letter c to refer to any positive constant. Successive occurrences of the letter c do not necessarily imply that the constants have identical values.

13.3.1.1 Simple Uniform Convergence Bounds

Recall that we assume that $\mathcal{F}$ is linearly parameterized by $w \in \mathbb{R}^d$. Elementary uniform convergence results then state that

$$\mathbb{E}\left[\sup_{f \in \mathcal{F}} |E(f) - E_n(f)|\right] \leq c\sqrt{\frac{d}{n}},$$

where the expectation is taken with respect to the random choice of the training set.[2] This result immediately provides a bound on the estimation error:

$$\begin{aligned}\mathcal{E}_{\text{est}} &= \mathbb{E}\left[\left(E(f_n)-E_n(f_n)\right)+\left(E_n(f_n)-E_n(f_{\mathcal{F}}^*)\right)+\left(E_n(f_{\mathcal{F}}^*)-E(f_{\mathcal{F}}^*)\right)\right] \\ &\leq 2\,\mathbb{E}\left[\sup_{f\in\mathcal{F}}|E(f)-E_n(f)|\right] \leq c\sqrt{\frac{d}{n}}.\end{aligned}$$

This same result also provides a combined bound for the estimation and optimization errors:

$$\begin{aligned}\mathcal{E}_{\text{est}}+\mathcal{E}_{\text{opt}} &= \mathbb{E}\big[E(\tilde{f}_n)-E_n(\tilde{f}_n)\big]+\mathbb{E}\big[E_n(\tilde{f}_n)-E_n(f_n)\big] \\ &+ \mathbb{E}\left[E_n(f_n)-E_n(f_{\mathcal{F}}^*)\right]+\mathbb{E}\left[E_n(f_{\mathcal{F}}^*)-E(f_{\mathcal{F}}^*)\right] \\ &\leq c\sqrt{\frac{d}{n}}+\rho+0+c\sqrt{\frac{d}{n}} = \mathcal{O}\left(\rho+\sqrt{\frac{d}{n}}\right).\end{aligned}$$

Unfortunately, this convergence rate is known to be pessimistic in many important cases. More sophisticated bounds are required.

13.3.1.2 Faster Rates in the Realizable Case

When the loss function $\ell(\hat{y},y)$ is positive, with probability $1-e^{-\tau}$ for any $\tau>0$, relative uniform convergence bounds (e.g., Vapnik, 1982) state that

$$\sup_{f\in\mathcal{F}}\frac{E(f)-E_n(f)}{\sqrt{E(f)}}\leq c\sqrt{\frac{d}{n}\log\frac{n}{d}+\frac{\tau}{n}}\,.$$

This result is very useful because it provides faster convergence rates $\mathcal{O}(\log n/n)$ in the *realizable case*, that is, when $\ell(f_n(x_i),y_i)=0$ for all training examples (x_i,y_i). We then have $E_n(f_n)=0$, and $E_n(\tilde{f}_n)\leq\rho$, and we can write

$$E(\tilde{f}_n)-\rho\leq c\sqrt{E(\tilde{f}_n)}\,\sqrt{\frac{d}{n}\log\frac{n}{d}+\frac{\tau}{n}}\,.$$

Viewing this as a second-degree polynomial inequality in variable $\sqrt{E(\tilde{f}_n)}$, we obtain

2. Although the original Vapnik-Chervonenkis bounds have the form $c\sqrt{\frac{d}{n}\log\frac{n}{d}}$, the logarithmic term can be eliminated using the "chaining" technique (e.g., Bousquet, 2002).

$$E(\tilde{f}_n) \leq c\left(\rho + \frac{d}{n}\log\frac{n}{d} + \frac{\tau}{n}\right).$$

Integrating this inequality using a standard technique (see, e.g., Massart, 2000), we obtain a better convergence rate of the combined estimation and optimization error:

$$\mathcal{E}_{\text{est}} + \mathcal{E}_{\text{opt}} = \mathbb{E}\left[E(\tilde{f}_n) - E(f^*_{\mathcal{F}})\right] \leq \mathbb{E}\left[E(\tilde{f}_n)\right] = c\left(\rho + \frac{d}{n}\log\frac{n}{d}\right).$$

13.3.1.3 Fast Rate Bounds

Many authors (e.g., Bousquet, 2002; Bartlett and Mendelson, 2006; Bartlett et al., 2006) obtain fast statistical estimation rates in more general conditions. These bounds have the general form

$$\mathcal{E}_{\text{app}} + \mathcal{E}_{\text{est}} \leq c\left(\mathcal{E}_{\text{app}} + \left(\frac{d}{n}\log\frac{n}{d}\right)^{\alpha}\right) \quad \text{for} \quad \frac{1}{2} \leq \alpha \leq 1\,. \tag{13.4}$$

This result holds when one can establish the following variance condition:

$$\forall f \in \mathcal{F} \quad \mathbb{E}\left[\left(\ell(f(X),Y) - \ell(f^*_{\mathcal{F}}(X),Y)\right)^2\right] \leq c\left(E(f) - E(f^*_{\mathcal{F}})\right)^{2-\frac{1}{\alpha}}. \tag{13.5}$$

The convergence rate of (13.4) is described by the exponent α, which is determined by the quality of the variance bound (13.5). Works on fast statistical estimation identify two main ways to establish such a variance condition.

- Exploiting the strict convexity of certain loss functions (Bartlett et al., 2006, theorem 12). For instance, Lee et al. (1998) establish a $\mathcal{O}(\log n/n)$ rate using the squared loss $\ell(\hat{y}, y) = (\hat{y} - y)^2$.
- Making assumptions on the data distribution. In the case of pattern recognition problems, for instance, the Tsybakov condition indicates how cleanly the posterior distributions $P(y|x)$ cross near the optimal decision boundary (Tsybakov, 2004; Bartlett et al., 2006). The realizable case discussed in section 13.3.1.2 can be viewed as an extreme example of this.

Despite their much greater complexity, fast rate estimation results can accommodate the optimization accuracy ρ, using essentially the methods illustrated in sections 13.3.1.1 and 13.3.1.2. We then obtain a bound of the form

$$\mathcal{E} = \mathcal{E}_{\text{app}} + \mathcal{E}_{\text{est}} + \mathcal{E}_{\text{opt}} = \mathbb{E}\left[E(\tilde{f}_n) - E(f^*)\right] \leq c\left(\mathcal{E}_{\text{app}} + \left(\frac{d}{n}\log\frac{n}{d}\right)^{\alpha} + \rho\right). \tag{13.6}$$

For instance, a general result with $\alpha = 1$ is provided by Massart (2000, theorem 4.2). Combining this result with standard bounds on the complexity of classes of linear functions (e.g., Bousquet, 2002) yields the following result:

$$\mathcal{E} = \mathcal{E}_{\text{app}} + \mathcal{E}_{\text{est}} + \mathcal{E}_{\text{opt}} = \mathbb{E}\left[E(\tilde{f}_n) - E(f^*)\right] \leq c\left(\mathcal{E}_{\text{app}} + \frac{d}{n}\log\frac{n}{d} + \rho\right). \tag{13.7}$$

See also Mendelson (2003), and Bartlett and Mendelson (2006) for more bounds taking the optimization accuracy into account.

13.3.2 Gradient Optimization Algorithms

We now discuss and compare the asymptotic learning properties of four gradient optimization algorithms. Recall that the family of function $\mathcal{F}$ is linearly parameterized by $w \in \mathbb{R}^d$. Let $w^*_{\mathcal{F}}$ and w_n correspond to the functions $f^*_{\mathcal{F}}$ and f_n defined in section 13.2.1. In this section, we assume that the functions $w \mapsto \ell(f_w(x), y)$ are convex and twice differentiable with continuous second derivatives. For simplicity we also assume that the empirical const function $C(w) = E_n(f_w)$ has a single minimum, w_n.

Two matrices play an important role in the analysis: the Hessian matrix H and the gradient covariance matrix G, both measured at the empirical optimum w_n:

$$H = \frac{\partial^2 C}{\partial w^2}(w_n) = \mathbb{E}_n\left[\frac{\partial^2 \ell(f_{w_n}(x), y)}{\partial w^2}\right], \tag{13.8}$$

$$G = \mathbb{E}_n\left[\left(\frac{\partial \ell(f_{w_n}(x), y)}{\partial w}\right)\left(\frac{\partial \ell(f_{w_n}(x), y)}{\partial w}\right)'\right]. \tag{13.9}$$

The relation between these two matrices depends on the chosen loss function. In order to summarize them, we assume that there are constants $\lambda_{\max} \geq \lambda_{\min} > 0$ and $\nu > 0$ such that, for any $\eta > 0$, we can choose the number of examples n large enough to ensure that the following assertion is true with probability greater than $1 - \eta$:

$$\text{tr}(G\,H^{-1}) \leq \nu \quad \text{and} \quad \text{EigenSpectrum}(H) \subset [\,\lambda_{\min}\,,\,\lambda_{\max}\,]\,. \tag{13.10}$$

The condition number $\kappa = \lambda_{\max}/\lambda_{\min}$ provides a convenient measure of the difficulty of the optimization problem (Dennis Jr. and Schnabel, 1983).

The assumption $\lambda_{\min} > 0$ avoids complications with stochastic gradient algorithms. This assumption is weaker than strict convexity because it applies only in the vicinity of the optimum. For instance, consider a loss function obtained by smoothing the well-known hinge loss $\ell(z, y) = \max\{0, 1-yz\}$ in a small neighborhood of its non-differentiable points. Function $C(w)$ is then piecewise linear with smoothed edges and vertices. It is not strictly convex. However, its minimum is likely to be on a smoothed vertex with a non singular Hessian. When we have strict convexity, the argument of Bartlett et al. (2006, theorem 12) yields fast estimation rates $\alpha \approx 1$ in (13.4) and (13.6). That is not necessarily the case here.

The four algorithms considered in this chapter use information about the gradient of the cost function to iteratively update their current estimate $w(t)$ of the parameter vector.

- *Gradient descent (GD)* iterates

$$w(t+1) \;=\; w(t) - \eta \frac{\partial C}{\partial w}(w(t)) \quad = \; w(t) - \eta \frac{1}{n} \sum_{i=1}^{n} \frac{\partial}{\partial w} \ell\big(f_{w(t)}(x_i), y_i\big)$$

where $\eta > 0$ is a small enough gain. GD is an algorithm with *linear convergence* (Dennis Jr. and Schnabel, 1983): when $\eta = 1/\lambda_{\max}$, this algorithm requires $\mathcal{O}(\kappa \log(1/\rho))$ iterations to reach accuracy ρ. The exact number of iterations depends on the choice of the initial parameter vector.

- *Second-order gradient descent (2GD)* iterates

$$w(t+1) \;=\; w(t) - H^{-1} \frac{\partial C}{\partial w}(w(t)) \;=\; w(t) - \frac{1}{n} H^{-1} \sum_{i=1}^{n} \frac{\partial}{\partial w} \ell\big(f_{w(t)}(x_i), y_i\big)$$

where matrix H^{-1} is the inverse of the Hessian matrix (13.8). This is more favorable than Newton's algorithm because we do not evaluate the local Hessian at each iteration, but optimistically assume that an oracle has revealed in advance the value of the Hessian at the optimum. 2GD is a superlinear optimization algorithm with *quadratic convergence* (Dennis Jr. and Schnabel, 1983). When the cost is quadratic, a single iteration is sufficient. In the general case, $\mathcal{O}(\log \log(1/\rho))$ iterations are required to reach accuracy ρ.

- *Stochastic gradient descent (SGD)* picks a random training example (x_t, y_t) at each iteration and updates the parameter w on the basis of this example

only:

$$w(t+1) \;=\; w(t) - \frac{\eta}{t}\,\frac{\partial}{\partial w}\ell\big(f_{w(t)}(x_t), y_t\big).$$

Murata (1998, section 2.2) characterizes the mean $\mathbb{E}_{\mathcal{S}}[w(t)]$ and variance $\mathbb{V}\mathrm{ar}_{\mathcal{S}}[w(t)]$ with respect to the distribution implied by the random examples drawn from a given training set $\mathcal{S}$ at each iteration. Applying this result to the discrete training set distribution for $\eta = 1/\lambda_{\min}$, we have $\delta w(t)^2 = \mathcal{O}(1/t)$ where $\delta w(t)$ is a shorthand notation for $w(t) - w_n$.
We can then write

$$\begin{aligned}
\mathbb{E}_{\mathcal{S}}\big[\,C(w(t)) - \inf C\,\big] &= \mathbb{E}_{\mathcal{S}}\big[\mathrm{tr}\big(H\,\delta w(t)\,\delta w(t)'\big)\big] + \mathrm{o}\big(\tfrac{1}{t}\big) \\
&= \mathrm{tr}\,\big(\,H\,\mathbb{E}_{\mathcal{S}}[\delta w(t)]\,\mathbb{E}_{\mathcal{S}}[\delta w(t)]' + H\,\mathbb{V}\mathrm{ar}_{\mathcal{S}}[w(t)]\,\big) + \mathrm{o}\big(\tfrac{1}{t}\big) \\
&\le \tfrac{\mathrm{tr}(GH)}{t} + \mathrm{o}\big(\tfrac{1}{t}\big) \;\le\; \tfrac{\nu\kappa^2}{t} + \mathrm{o}\big(\tfrac{1}{t}\big)\,.
\end{aligned} \tag{13.11}$$

Therefore, the SGD algorithm reaches accuracy ρ after less than $\nu\kappa^2/\rho + \mathrm{o}(1/\rho)$ iterations on average. The SGD convergence is essentially limited by the stochastic noise induced by the random choice of one example at each iteration. Neither the initial value of the parameter vector w nor the total number of examples n appears in the dominant term of this bound! When the training set is large, one could reach the desired accuracy ρ measured on the whole training set without even visiting all the training examples. This is in fact a kind of generalization bound.

- *Second-order stochastic gradient descent (2SGD)* replaces the gain η with the inverse of the Hessian matrix H:

$$w(t+1) \;=\; w(t) - \frac{1}{t}\,H^{-1}\,\frac{\partial}{\partial w}\ell\big(f_{w(t)}(x_t), y_t\big).$$

Unlike standard gradient algorithms, using the second-order information does not change the influence of ρ on the convergence rate but improves the constants. Again using (Murata, 1998, theorem 4), accuracy ρ is reached after $\nu/\rho + \mathrm{o}(1/\rho)$ iterations.

For each of the four gradient algorithms, the first three columns of table 13.2 report the time for a single iteration, the number of iterations needed to reach a predefined accuracy ρ, and their product, the time needed to reach accuracy ρ. These asymptotic results are valid with probability 1, since the probability of their complement is smaller than η for any $\eta > 0$.

The fourth column bounds the time necessary to reduce the excess error $\mathcal{E}$ below $c\,(\mathcal{E}_{\mathrm{app}} + \varepsilon)$ where c is the constant from (13.6). This is computed by

Algorithm	Cost of one iteration	Iterations to reach ρ	Time to reach accuracy ρ	Time to reach $\mathcal{E} \leq c\,(\mathcal{E}_{\text{app}} + \varepsilon)$
GD	$\mathcal{O}(nd)$	$\mathcal{O}\Big(\kappa \log \frac{1}{\rho}\Big)$	$\mathcal{O}\Big(nd\kappa \log \frac{1}{\rho}\Big)$	$\mathcal{O}\Big(\frac{d^2\,\kappa}{\varepsilon^{1/\alpha}} \log^2 \frac{1}{\varepsilon}\Big)$
2GD	$\mathcal{O}\big(d^2 + nd\big)$	$\mathcal{O}\Big(\log\log \frac{1}{\rho}\Big)$	$\mathcal{O}\Big(\big(d^2 + nd\big) \log\log \frac{1}{\rho}\Big)$	$\mathcal{O}\Big(\frac{d^2}{\varepsilon^{1/\alpha}} \log \frac{1}{\varepsilon} \log\log \frac{1}{\varepsilon}\Big)$
SGD	$\mathcal{O}(d)$	$\frac{\nu\kappa^2}{\rho} + \mathrm{o}\Big(\frac{1}{\rho}\Big)$	$\mathcal{O}\Big(\frac{d\nu\kappa^2}{\rho}\Big)$	$\mathcal{O}\Big(\frac{d\,\nu\,\kappa^2}{\varepsilon}\Big)$
2SGD	$\mathcal{O}\big(d^2\big)$	$\frac{\nu}{\rho} + \mathrm{o}\Big(\frac{1}{\rho}\Big)$	$\mathcal{O}\Big(\frac{d^2\nu}{\rho}\Big)$	$\mathcal{O}\Big(\frac{d^2\,\nu}{\varepsilon}\Big)$

Table 13.2: Asymptotic results for gradient algorithms (with probability 1). Compare the second-to-last column (time to optimize) with the last column (time to reach the excess test error ε). n–number of examples; d–parameter dimension; for κ, ν see equation (13.10).

observing that choosing $\rho \sim \big(\frac{d}{n} \log \frac{n}{d}\big)^{\alpha}$ in (13.6) achieves the fastest rate for ε, with minimal computation time. We can then use the asymptotic equivalences $\rho \sim \varepsilon$ and $n \sim \frac{d}{\varepsilon^{1/\alpha}} \log \frac{1}{\varepsilon}$. Setting the fourth column expressions to $T_{\max}$ and solving for ε yields the *best excess error achieved by each algorithm* within the limited time $T_{\max}$. This provides the asymptotic solution of the estimation–optimization tradeoff (13.3) for large-scale problems satisfying our assumptions.

These results clearly show that the generalization performance of *large-scale learning systems* depends on both the statistical properties of the objective function and the computational properties of the chosen optimization algorithm. Their combination leads to surprising consequences:

- *The SGD and 2SGD results do not depend on the estimation rate α.* When the estimation rate is poor, there is less need to optimize accurately. That leaves time to process more examples. A potentially more useful interpretation leverages the fact that (13.11) is already a kind of generalization bound: its fast rate trumps the slower rate assumed for the estimation error.
- *Second-order algorithms bring few asymptotical improvements in ε.* Although the superlinear 2GD algorithm improves the logarithmic term, all four algorithms are dominated by the polynomial term in $(1/\varepsilon)$. However, there are important variations in the influence of the constants d, κ, and ν. These constants are very important in practice.
- *Stochastic algorithms (SGD, 2SGD) yield the best generalization performance despite showing the worst optimization performance* on the empirical cost. This phenomenon has already been described and observed in experiments (e.g., Bottou and Le Cun, 2004).

In contrast, since the optimization error $\mathcal{E}_{\text{opt}}$ of *small-scale learning systems* can be reduced to insignificant levels, their generalization performance is

Model	Algorithm	Training Time	Objective	Test Error
Hinge loss $\lambda = 10^{-4}$	SVMLight	23,642 secs	0.2275	6.02%
	SVMPerf	66 secs	0.2278	6.03%
	SGD	**1.4 secs**	0.2275	6.02%
Logistic loss $\lambda = 10^{-5}$	TRON ($\rho = 10^{-2}$)	30 secs	0.18907	5.68%
	TRON ($\rho = 10^{-3}$)	44 secs	0.18890	5.70%
	SGD	**2.3 secs**	0.18893	5.66%

Table 13.3: Results with linear Support Vector Machines on the RCV1 dataset.

determined solely by the statistical properties of the objective function.

13.4 Experiments

This section empirically compares SGD with other optimization algorithms on two well known machine learning tasks. The SGD C++ source code is available from `http://leon.bottou.org/projects/sgd`.

13.4.1 SGD for Support Vector Machines

We first consider a well-known text categorization task, the classification of documents belonging to the CCAT category in the RCV1-v2 dataset (Lewis et al., 2004). In order to collect a large training set, we swap the RCV1-v2 official training and testing sets. The resulting training sets and test sets contain 781,265 and 23,149 examples, respectively. The 47,152 TF/IDF features are recomputed on the basis of this new split. We use a simple linear model with the usual hinge loss Support Vector Machine objective function

$$\min_w \quad C(w,b) \;=\; \frac{\lambda}{2} + \frac{1}{n}\sum_{i=1}^{n} \ell(y_t(wx_t+b)) \quad \text{with} \;\; \ell(z) = \max\{0, 1-z\}\,.$$

The first two rows of table 13.3 replicate the results reported by Joachims (2006) for the same data and the same value of the hyperparameter λ.

The third row of table 13.3 reports results obtained with the SGD algorithm:

$$w_{t+1} = w_t - \eta_t \left(\lambda w + \frac{\partial \ell(y_t(wx_t+b))}{\partial w} \right) \quad \text{with} \;\; \eta_t = \frac{1}{\lambda(t+t_0)}\,.$$

The bias b is updated similarly. Since λ is a lower bound of the smallest eigenvalue of the Hessian, our choice of gains η_t approximates the optimal schedule (see section 13.3.2). The offset t_0 was chosen to ensure that the

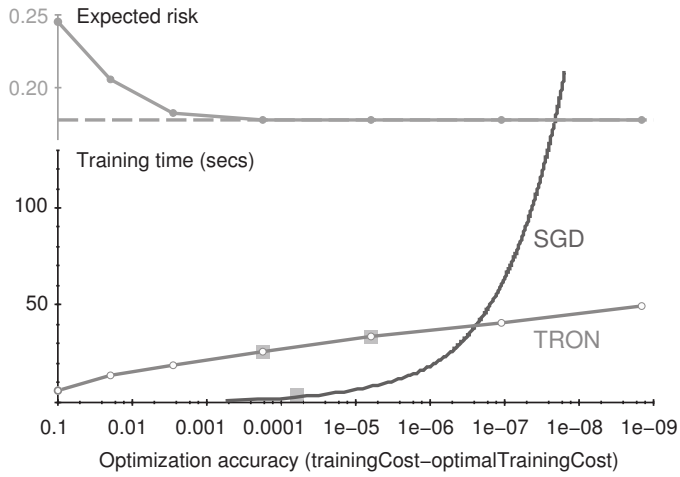

Figure 13.1: Training time and testing loss as a function of the optimization accuracy ρ for SGD and TRON (Lin et al., 2007)

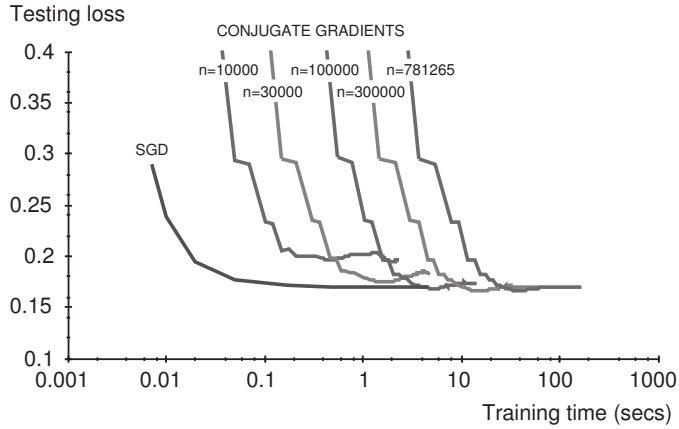

Figure 13.2: Testing loss versus training time for SGD, and for conjugate gradients running on subsets of the training set

initial gain is comparable with the expected size of the parameter w. The results clearly indicate that SGD offers a good alternative to the usual Support Vector Machine solvers.

Comparable results were obtained by Shalev-Shwartz et al. (2007), using an algorithm that essentially amounts to a stochastic gradient corrected by a projection step. Our results indicate that the projection step is not an essential component of this performance.

Table 13.3 also reports results obtained with the logistic loss $\ell(z) = \log(1 + e^{-z})$ in order to avoid the issues related to the nondifferentiability of the hinge loss. Note that this experiment uses a much better value for λ. Our comparison points were obtained with a state-of-the-art superlinear optimizer (Lin et al., 2007), using the stopping criteria $\rho = 10^{-2}$ and $\rho = 10^{-3}$. The very simple SGD algorithm clearly learns faster.

Figure 13.1 shows how much time each algorithm takes to reach a given optimization accuracy. The superlinear algorithm TRON reaches the optimum with 10 digits of accuracy in less than one minute. The stochastic gradient starts more quickly but is unable to deliver such a high accuracy. The upper part of the figure clearly shows that the testing set loss stops decreasing long before the superlinear algorithm overcomes the SGD algorithm.

Figure 13.2 shows how the testing loss evolves with the training time. The stochastic gradient descent curve can be compared with the curves obtained using conjugate gradients[3] on subsets of the training examples with increasing sizes. Assume, for instance, that our computing time budget is one second. Running the conjugate gradient algorithm on a random subset of 30,000 training examples achieves a much better performance than running it on the whole training set. How to guess the right subset size a priori remains unclear. Meanwhile, running the SGD algorithm on the full training set reaches the same testing set performance much faster.

13.4.2 SGD for Conditional Random Fields

The CoNLL 2000 chunking task (Tjong Kim Sang and Buchholz, 2000) consists of dividing a sentence into syntactically correlated segments such as noun phrase, verb phrase, and so on. The training set contains 8936 sentences divided into 106,978 segments. Error measurements are performed using a separate set of 2012 sentences divided into 23,852 segments. Results are

3. This experimental setup was suggested by Olivier Chapelle (personal communication). His variant of the conjugate gradient algorithm performs inexact line searches using a single inexpensive Newton step. This is effective because exact line searches usually demand many function evaluations which are expensive when the training set is large.

Algorithm	Training Time	Training Cost	Test F1 Score
CRF++/L-BFGS	4335 secs	**9042**	93.74%
CRF SGD	**568 secs**	9098	93.75%

Table 13.4: Results for Conditional Random Fields on the CoNLL 2000 chunking task

traditionally reported using an F1 measure that takes into account both the segment boundaries and the segment classes.

The chunking task has been successfully approached using Conditional Random Fields (Lafferty et al., 2001; Sha and Pereira, 2003) to tag the words with labels indicating the class and the boundaries of each segment. Our baseline is the Conditional Random Field model provided with the CRF++ software (Kudo, 2007). Our CRF SGD implementation replicates the features of the CRF++ software but uses SGD to optimize the Conditional Random Field objective function. The model contains 1,679,700 parameters in both cases.

Table 13.4 compares the training time, the final training cost, and the test performance of the model when trained using the standard CRF++ L-BFGS optimizer and the SGD implementation. The SGD version runs considerably faster.

Comparable speeds were obtained by Vishwanathan et al. (2006), using a stochastic gradient with a novel adaptive gain scheduling method. Our results indicate that this adaptive gain is not the essential component of this performance. The main cause lies with the fundamental tradeoffs outlined in this chapter.

13.5 Conclusion

Taking into account budget constraints on both the number of examples and the computation time, we find *qualitative differences* between the generalization performance of small-scale learning systems and large-scale learning systems. The generalization properties of large-scale learning systems depend on both the statistical properties of the objective function and the computational properties of the optimization algorithm. We illustrate this fact with some asymptotic results on gradient algorithms.

This framework leaves room for considerable refinements. Shalev-Shwartz and Srebro (2008) rigorously extend the analysis to regularized risk formulations with linear parameterization and find again that, for learning purposes, SGD algorithms are often more attractive than standard primal or dual al-

gorithms with good optimization complexity (Joachims, 2006; Hush et al., 2006). It could also be interesting to investigate how the choice of a surrogate loss function (Zhang, 2004; Bartlett et al., 2006) impacts the large-scale case.

13.6 References

P. L. Bartlett and S. Mendelson. Empirical minimization. *Probability Theory and Related Fields*, 135(3):311–334, 2006.

P. L. Bartlett, M. I. Jordan, and J. D. McAuliffe. Convexity, classification and risk bounds. *Journal of the American Statistical Association*, 101(473):138–156, 2006.

L. Bottou and O. Bousquet. The tradeoffs of large scale learning. In J. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems 20*, pages 161–168. MIT Press, 2008.

L. Bottou and Y. Le Cun. Large scale online learning. In S. Thrun, L. K. Saul, and B. Schölkopf, editors, *Advances in Neural Information Processing Systems 16*. MIT Press, Cambridge, MA, 2004.

S. Boucheron, O. Bousquet, and G. Lugosi. Theory of classification: a survey of recent advances. *ESAIM: Probability and Statistics*, 9:323–375, 2005.

O. Bousquet. *Concentration Inequalities and Empirical Processes Theory Applied to the Analysis of Learning Algorithms*. PhD thesis, Ecole Polytechnique, 2002.

J. E. Dennis Jr. and R. B. Schnabel. *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*. Prentice-Hall, Englewood Cliffs, NJ, 1983.

R. O. Duda and P. E. Hart. *Pattern Classification and Scene Analysis*. John Wiley, 1973.

D. Hush, P. Kelly, C. Scovel, and I. Steinwart. QP algorithms with guaranteed accuracy and run time for support vector machines. *Journal of Machine Learning Research*, 7:733–769, 2006.

T. Joachims. Training linear SVMs in linear time. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 217–226, Philadelphia, PA, August 2006. ACM Press.

J. S. Judd. On the complexity of loading shallow neural networks. *Journal of Complexity*, 4(3):177–192, 1988.

T. Kudo. CRF++: Yet another CRF toolkit, 2007. `http://crfpp.sourceforge.net`.

J. D. Lafferty, A. McCallum, and F. C. N. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In C. E. Brodley and A. P. Danyluk, editors, *Proceedings of the Eighteenth International Conference on Machine Learning*, pages 282–289. Morgan Kaufmann, 2001.

W. S. Lee, P. L. Bartlett, and R. C. Williamson. The importance of convexity in learning with squared loss. *IEEE Transactions on Information Theory*, 44(5): 1974–1980, 1998.

D. D. Lewis, Y. Yang, T. G. Rose, and F. Li. RCV1: A new benchmark collection for

text categorization research. *Journal of Machine Learning Research*, 5:361–397, 2004.

C.-J. Lin, R. C. Weng, and S. S. Keerthi. Trust region Newton methods for large-scale logistic regression. In Z. Ghahramani, editor, *Proceedings of the 24th International Conference on Machine Learning*, pages 561–568, Corvallis, OR, June 2007. ACM Press.

P. Massart. Some applications of concentration inequalities to statistics. *Annales de la Faculté des Sciences de Toulouse*, series 6, 9(2):245–303, 2000.

S. Mendelson. A few notes on statistical learning theory. In S. Mendelson and A. J. Smola, editors, *Advanced Lectures in Machine Learning*, volume 2600 of *Lecture Notes in Computer Science*, pages 1–40. Springer-Verlag, New York, 2003.

N. Murata. A statistical study of on-line learning. In D. Saad, editor, *Online Learning and Neural Networks*. Cambridge University Press, 1998.

F. Sha and F. Pereira. Shallow parsing with conditional random fields. In *Proceedings of the 2003 Human Language Technology Conference and 4th Meeting of the North American Chapter of the Association for Computational Linguistics*, 2003.

S. Shalev-Shwartz and N. Srebro. SVM optimization: inverse dependence on training set size. In *Proceedings of the 25th International Conference on Machine Learning*, pages 928–935. ACM Press, 2008.

S. Shalev-Shwartz, Y. Singer, and N. Srebro. Pegasos: Primal estimated subgradient solver for SVM. In Z. Ghahramani, editor, *Proceedings of the 24th International Conference on Machine Learning*, pages 807–814. ACM Press, June 2007.

I. Steinwart and C. Scovel. Fast rates for support vector machines. In P. Auer and R. Meir, editors, *Proceedings of the 18th Conference on Learning Theory (COLT 2005)*, volume 3559 of *Lecture Notes in Computer Science*, pages 279–294, Bertinoro, Italy, June 2005. Springer-Verlag.

E. F. Tjong Kim Sang and S. Buchholz. Introduction to the CoNLL-2000 Shared Task: Chunking. In C. Cardie, W. Daelemans, C. Nedellec, and E. Tjong Kim Sang, editors, *Proceedings of CoNLL-2000 and LLL-2000*, pages 127–132, 2000.

A. B. Tsybakov. Optimal aggregation of classifiers in statistical learning. *Annals of Statististics*, 32(1):135–166, 2004.

L. G. Valiant. A theory of the learnable. *Proceedings of the 16th Annual ACM Symposium on the Theory of Computing*, pages 436–445, 1984.

V. N. Vapnik. *Estimation of Dependences Based on Empirical Data*. Springer-Verlag, Berlin, 1982.

V. N. Vapnik, E. Levin, and Y. LeCun. Measuring the VC-dimension of a learning machine. *Neural Computation*, 6(5):851–876, 1994.

S. V. N. Vishwanathan, N. N. Schraudolph, M. W. Schmidt, and K. P. Murphy. Accelerated training of conditional random fields with stochastic gradient methods. In W. W. Cohen and A. Moore, editors, *Proceedings of the 23rd International Conference on Machine Learning*, pages 969–976. ACM Press, 2006.

T. Zhang. Statistical behavior and consistency of classification methods based on convex risk minimization. *The Annals of Statistics*, 32:56–85, 2004.