Implicit regularization and acceleration in machine learning

Lorenzo Rosasco
MaLGa- Machine learning Genova Center
Università di Genova
MIT
IIT

Joint work with: N. Pagliana, MaLGa - DIMA - UniGe

Mathematics of Deep Learning - TU Delft
There seems to be a puzzle

Figure 2: Effects of implicit regularizers on generalization performance. Figure 2a shows the training and testing accuracy on ImageNet. The shaded area indicates the accumulative best test accuracy, as a reference of potential performance gain for early stopping. However, on the CIFAR10 dataset, we do not observe any potential benefit of early stopping.

Batch normalization (Ioffe & Szegedy, 2015) is an operator that normalizes the layer responses within each mini-batch. It has been widely adopted in many modern neural network architectures such as Inception (Szegedy et al., 2016) and Residual Networks (He et al., 2016). Although not explicitly designed for regularization, batch normalization is usually found to improve the generalization performance. The Inception architecture uses a lot of batch normalization layers. To test the impact of batch normalization, we create a “Inception w/o BatchNorm” architecture that is exactly the same as Inception in Figure 3, except with all the batch normalization layers removed. Figure 2b shows the training and testing accuracy on ImageNet for Inception and Inception w/o BatchNorm.

We say “potentially” because to make this statement rigorous, we need to have another isolated test set and test the performance there when we choose early stopping point on the first test set (acting like a validation set).

In Table 2 in the appendix, we show in parentheses the best test accuracy along the training process. It confirms that early stopping potentially improves the generalization performance. Figure 2a shows the training and testing accuracy on ImageNet. The shaded area indicates the accumulative best test accuracy, as a reference of potential performance gain for early stopping. However, on the CIFAR10 dataset, we do not observe any potential benefit of early stopping.

Batch normalization (Ioffe & Szegedy, 2015) is an operator that normalizes the layer responses within each mini-batch. It has been widely adopted in many modern neural network architectures such as Inception (Szegedy et al., 2016) and Residual Networks (He et al., 2016). Although not explicitly designed for regularization, batch normalization is usually found to improve the generalization performance. The Inception architecture uses a lot of batch normalization layers. To test the impact of batch normalization, we create a “Inception w/o BatchNorm” architecture that is exactly the same as Inception in Figure 3, except with all the batch normalization layers removed. Figure 2b shows the training and testing accuracy on ImageNet for Inception and Inception w/o BatchNorm.

We say “potentially” because to make this statement rigorous, we need to have another isolated test set and test the performance there when we choose early stopping point on the first test set (acting like a validation set).
Outline

Optimization for machine learning

Part I: Learning theory of (accelerated) optimization

Part II: More learning theory and some science of (accelerated) optimization
   Refined results: easy problems
   Refined results: hard problems
Optimization for machine learning

Training error

\[
\min_{w \in \mathbb{R}^p} \frac{1}{n} \sum_{i=1}^{n} \ell(f_w(x_i), y_i) + \lambda \|w\|^2
\]

Gradient methods

\[
\hat{w}_{t+1} = \hat{w}_t - \gamma_t \frac{1}{n} \sum_{i=1}^{n} \nabla \ell(f_{\hat{w}_t}(x_i), y_i) - 2\gamma_t \lambda \hat{w}_t
\]
Optimization for machine learning

Training error

$$\min_{w \in \mathbb{R}^p} \frac{1}{n} \sum_{i=1}^{n} \ell(f_w(x_i), y_i) + \lambda \|w\|^2$$

Gradient methods

$$\hat{w}_{t+1} = \hat{w}_t - \gamma_t \frac{1}{n} \sum_{i=1}^{n} \nabla \ell(f_{\hat{w}_t}(x_i), y_i) - 2\gamma_t \lambda \hat{w}_t$$

$$\lim_{t \to \infty} \frac{1}{n} \sum_{i=1}^{n} \ell(f_{\hat{w}_t}(x_i), y_i) + \lambda \|\hat{w}_t\|^2 = \min_{w \in \mathbb{R}^p} \frac{1}{n} \sum_{i=1}^{n} \ell(f_w(x_i), y_i) + \lambda \|w\|^2$$

$$\Rightarrow \text{Go faster! ...but where?}$$
Statistical machine learning

\[ \frac{1}{n} \sum_{i=1}^{n} \ell(f_w(x_i), y_i) \approx \mathbb{E}_{x,y}[\ell(f_w(x), y)] \]

Test error

\[ \mathbb{E}_{x,y}[\ell(f_{\hat{w}_t}^\top(x), y)] \]
Error measures

Generalization error

\[ \frac{1}{n} \sum_{i=1}^{n} \ell(f_{\hat{w}_t}(x_i), y_i) - \mathbb{E}_{x,y}[\ell(f_{\hat{w}_t}(x), y)] \]

Excess risk

\[ \mathbb{E}_{x,y}[\ell(f_{\hat{w}_t}(x), y)] - \min_{w \in \mathbb{R}^p} \mathbb{E}_{x,y}[\ell(f_w(x), y)] \]
Outline

Optimization for machine learning

Part I: Learning theory of (accelerated) optimization

Part II: More learning theory and some science of (accelerated) optimization
- Refined results: easy problems
- Refined results: hard problems
Least squares learning

Solve

$$\min_{w \in \mathbb{R}^p} \mathbb{E}_{x,y} [(w^\top \Phi(x) - y)^2]$$

where $\Phi(x) \in \mathbb{R}^p$ and $p$ can be infinite.

Gradient descent$^1$

$$\hat{w}_{t+1} = \hat{w}_t - \alpha \nabla \hat{L}(\hat{w}_t), \quad \nabla \hat{L}(w) = \frac{2}{n} \sum_{i=1}^{n} \Phi(x_i)(w^\top \Phi(x_i) - y_i)$$

with

$$\alpha = \frac{1}{\sup_x \|\Phi(x)\|^2}.$$
Accelerated iterations

Heavy-ball

\[
\hat{w}_{t+1} = \hat{w}_t - \alpha_t \nabla \hat{L}(\hat{w}_t) + \beta_t (\hat{w}_t - \hat{w}_{t-1}).
\]

\(^2\)Reduces to Chebyshev iterative method for \(\nu = 1/2\).
Accelerated iterations

Heavy-ball

\[ \hat{w}_{t+1} = \hat{w}_t - \alpha_t \nabla \hat{L}(\hat{w}_t) + \beta_t (\hat{w}_t - \hat{w}_{t-1}) \]

In particular\(^2\) for \(\nu > 0\)

\[
\alpha_t = \frac{1}{\sup_x \|\Phi(x)\|^2} \frac{4(2t + 2\nu - 1)(t + \nu - 1)}{(t + 2\nu - 1)(2t + 4\nu - 1)}, \quad \beta_t = \frac{(t - 1)(2t - 3)(2t + 2\nu - 1)}{(t + 2\nu - 1)(2t + 4\nu - 1)(2t + 2\nu - 3)}.
\]

\(^2\)Reduces to Chebyshev iterative method for \(\nu = 1/2\).
Accelerated iterations

Heavy-ball

\[ \hat{w}_{t+1} = \hat{w}_t - \alpha_t \nabla \hat{L}(\hat{w}_t) + \beta_t (\hat{w}_t - \hat{w}_{t-1}). \]

In particular for \( \nu > 0 \)

\[ \alpha_t = \frac{1}{\sup_x \| \Phi(x) \|^2} \frac{4(2t + 2\nu - 1)(t + \nu - 1)}{(t + 2\nu - 1)(2t + 4\nu - 1)}, \quad \beta_t = \frac{(t - 1)(2t - 3)(2t + 2\nu - 1)}{(t + 2\nu - 1)(2t + 4\nu - 1)(2t + 2\nu - 3)}. \]

Nesterov's acceleration

\[ \hat{w}_{t+1} = \hat{v}_t - \alpha \nabla \hat{L}(\hat{v}_t), \quad \hat{v}_t = \hat{w}_t + \beta_t (\hat{w}_t - \hat{w}_{t-1}). \]

In particular for \( \beta > 1 \)

\[ \alpha = \frac{1}{\sup_x \| \Phi(x) \|^2}, \quad \beta_t = \frac{t - 1}{t + \beta}. \]

\(^2\)Reduces to Chebyshev iterative method for \( \nu = 1/2. \)
Basic result

Let

\[ L(w) = \mathbb{E}_{x,y}[(w^T x - y)^2], \quad L(w_*) = \min_{w \in \mathbb{R}^p} L(w) \]

**Theorem**
Assume \( \|\Phi(x)\|, |y| \leq 1 \) a.s.. Then w.h.p.

\[ L(\hat{w}_t) - L(w_*) \lesssim \frac{1}{t} + \frac{t}{n} \]

for GD, whereas

\[ L(\hat{w}_t) - L(w_*) \lesssim \frac{1}{t^2} + \frac{t^2}{n} \]

for Heavy-ball and Nesterov acc.
Corollary

For GD, if \( t = \sqrt{n} \),

\[
L(\hat{w}_t) - L(w_*) \lesssim \frac{1}{\sqrt{n}}.
\]

The same bound hold for for Heavy-ball and Nesterov acc. for \( t = n^{1/4} \).
Numerical illustration

Parameters of the plot in the left: space size $N = 10^4$, training points $n = 10^2$, $\gamma = 1$, noise $\sigma = 0.5$, step-size $\alpha \ll 0.9 / \max(\text{eigs}(\hat{K})) \leq \frac{1}{\sup_x \|\Phi(x)\|^2}$.

Figure: Simulated data (ill-conditioned LS)

Figure: Pumadyn8nh dataset ($n = 8192$, $d = 7$), Gaussian kernel width $1.2$. 
Remarks

- Early stop after $\sqrt{n}$ iteration! Iterations control complexity/stability.
- Acceleration can suffer from instability.
- Iterates converge to minimal norm minimizer (implicit bias).
- Training error/generalization play no role.
- Proof based on spectral filtering/calculus (Engl et al. ’96, Neubauer ’16)
Remarks

- Early stop after $\sqrt{n}$ iteration! Iterations control complexity/stability.
- Acceleration can suffer from instability.
- Iterates converge to minimal norm minimizer (implicit bias).
- Training error/generalization play no role.
- Proof based on spectral filtering/calculus (Engl et al. ’96, Neubauer ’16)

We can see other behaviors in practice: explanation?
Outline

Optimization for machine learning

Part I: Learning theory of (accelerated) optimization

Part II: More learning theory and some science of (accelerated) optimization
  Refined results: easy problems
  Refined results: hard problems
Do we like assumptions or not?

- “Simple and Almost Assumption-Free Out-of-Sample Bound for ...”

- “…a more ambitious open problem (to find good bounds) is to find the correct characterization of “easiness” for real-world problem...”
Do we like assumptions or not?

▶ "Simple and Almost Assumption-Free Out-of-Sample Bound for ..."

▶ "...a more ambitious open problem (to find good bounds) is to find the correct characterization of “easiness” for real-world problem..."

We can see other behaviors in practice: explanation?
Refined assumption: easy problems

$$\Sigma = \mathbb{E}_x [\Phi(x)\Phi(x)^\top] \quad h = \mathbb{E}_{x,y} [\Phi(x)y]$$

Optimality condition

$$L(w_*) = \min_{w \in \mathbb{R}^p} \mathbb{E}_{x,y} [(w^\top \Phi(x) - y)^2] \iff \Sigma w_* = h.$$  

Error/source condition

$$w_* \in \text{Range}(\Sigma^s), \quad s \in [0, \infty)$$
Easy problems illustrated

\[ \mathcal{W}_* \]

Range (\(\Sigma\))
**Refined results**

**Theorem**

*Under the error/source condition, assume \( \|\Phi(x)\|, |y| \leq 1 \) a.s. Then w.h.p.*

\[
L(\hat{w}_t) - L(w_*) \lesssim \frac{1}{t^{2s+1}} + \frac{t}{n}
\]

*with \( s \in [0, \infty) \) for GD, whereas*

\[
L(\hat{w}_t) - L(w_*) \lesssim \frac{1}{t^{2(2s+1)}} + \frac{t^2}{n}
\]

*with \( s \in [0, \nu) \) for Heavy-ball and with \( s = 0 \) for Nesterov acc.*
Refined results (cont.)

Corollary

*For GD with* $s \in [0, \infty)$, *choosing* $t = n^{\frac{1}{2s+2}}$,

$$L(\hat{w}_t) - L(w_*) \lesssim \frac{1}{n^{\frac{2s+1}{2s+2}}}.$$  

*The same bound hold for Heavy-ball with* $s \in [0, \nu)$ *and for Nesterov acc. with* $s = 0$ *choosing* $t = \sqrt{n^{\frac{1}{2s+2}}}$.

*Acceleration can suffer from slow rates for easy problems.*
Numerical illustration

Parameters: space size $N = 10^4$, training points $n = 10^2$, $\gamma = 1$, noise $\sigma = 0.2$, step-size $\alpha = 0.9 / \max(\text{eigs}(\hat{K})) \leq \frac{1}{\sup_x \|\Phi(x)\|^2}$. 

Figure: $s = 0$

Figure: $s = 3/2$
Numerical illustration

Parameters: space size \( N = 10^4 \), training points \( n = 10^2 \), \( \gamma = 1 \), noise \( \sigma = 0.5 \), step-size \( \alpha = 0.9 / \max(\text{eigs}(\hat{K})) \leq \frac{1}{\sup_x \| \Phi(x) \|_2} \).

Figure: \( s = 0 \)

Figure: \( s = 50 \)
So far

- Large function class/simple target function: instability and slow rate?

  Gradient descent might catch up.

- What about small function class/complex target function?
Refined assumption: hard problems

Let

$$\bar{\Sigma} f(x) = \mathbb{E}_x[\Phi(x)f(x)]$$

General source condition

$$\mathbb{E}[y|x] \in \text{Range}(\log(\bar{\Sigma})).$$

Eigendecay

$$\sigma_j(\Sigma) \sim e^{-j}.$$ 

Example: Learn a smooth (Sobolev) function with a Gaussian kernel (fixed width!).
Hard problems illustrated

\[ \mathcal{W}_* \]

Range(\(\Sigma\))
Refined results

**Theorem**
*Under the error/source condition, assume \( \|\Phi(x)\|, |y| \leq 1 \text{ a.s.} \). Then w.h.p.*

\[
L(\hat{w}_t) - L(w_*) \lesssim \frac{1}{\log(t)} + \frac{\log(t)}{n} + \frac{t}{n^2}
\]

*with for GD, whereas for*

\[
L(\hat{w}_t) - L(w_*) \lesssim \frac{1}{2\log(t)} + \frac{2\log(t)}{n} + \frac{t^2}{n^2}
\]

*for Heavy-ball and for Nesterov acc.*
Refined results (cont.)

Corollary

For GD choosing $t \sim n^\alpha$, $\alpha < 2$

$$L(\hat{w}_t) - L(w_*) \lesssim \frac{1}{\log(n)}.$$

The same bound hold for Heavy-ball and for Nesterov acc. with $t \sim \sqrt{n^\alpha}$, $\alpha < 2$. 
Numerical illustration

Parameters: space size $N = 10^4$, training points $n = 10^2$, $\gamma = 1$, source condition logarithmic, noise $\sigma = 0.2$, step-size $\alpha = 0.9/\max(\text{eigs}(\hat{K})) \leq \frac{1}{\sup_x \|\Phi(x)\|^2}$.

Figure: Simulation of the test error in the case $\sigma_i \approx e^{-\gamma i}$

Figure: Simulation of the test error in the case $\sigma_i \approx e^{-\gamma i}$ (zoom)
The behavior of an algorithms depending on modeling assumptions.

Which assumptions are good depends on data.

Looking at different assumptions allows to explaining different empirical behaviors.
Wrapping up

- Optimization for machine leads to new algorithms: implicit regularization.
- Different behaviors depending on easy/hard learning problems.
- TBD: high/low dimension and SNR, classification; nonlinear parameterization...

\[
L(\hat{\mathbf{w}}_t) - L(\mathbf{w}_*) \lesssim \frac{1}{\log(t)} + \frac{\log(t)}{n^2} + \frac{\text{const.}}{n^2}
\]
Outline

Spectral filtering
Spectral filtering

\[ \hat{\Sigma} = \frac{1}{n} \sum_{i=1}^{n} \Phi(x_i)\Phi(x_i)^\top, \quad \hat{h} = \frac{1}{n} \sum_{i=1}^{n} \Phi(x_i)y_i \]

GD Filter

\[ \hat{w}_{t+1} = \hat{w}_t - \alpha \nabla \hat{L}(\hat{w}_t) = \alpha \sum_{j=0}^{t} (1 - \alpha \hat{\Sigma})^j \hat{h} \]

For \( t \) large,

\[ g_t(\hat{\Sigma}) = \alpha \sum_{j=0}^{t} (1 - \alpha \hat{\Sigma})^j \approx \hat{\Sigma}^{-1} \]

\[ g_t(\hat{\Sigma}) = \alpha \sum_{j=0}^{t} (1 - \alpha \hat{\Sigma})^j \approx \hat{\Sigma}^{-1} \]
Spectral filters

Definition

$\{g_{\lambda}\}_{\lambda \in (0,1]}$ is a spectral filtering function if there exists $E, F_0, q, (F_s)_{s=0}^q < \infty$ s.t., for any $\lambda \in (0, 1]$

(i) $\sup_{\sigma \in (0, \kappa^2]} |g_{\lambda}(\sigma)| \leq \frac{E}{\lambda}.$

(ii) Let $r_{\lambda}(\sigma) = 1 - \sigma g_{\lambda}(\sigma)$, for $s \in [0, q)$

$\sup_{\sigma \in (0, \kappa^2]} |r_{\lambda}(\sigma)\sigma^s| \leq F_s \lambda^s.$

The parameter $q$ is called qualification.