Optimization of neural networks

Dmitry Yarotsky

Parametrized predictive models

True response function: $y = f(\mathbf{x})$, where $\mathbf{x}$ is the input vector

- $y \in \mathbb{R}$ for regression
- $y \in \{0, 1\}$ for binary classification

Predictive model: $y = \widetilde{f}(\mathbf{x}, \mathbf{W})$, and $\mathbf{W}$ are model parameters (e.g., network weights)

"Soft classification": $\widetilde{f}(\mathbf{x}, \mathbf{W}) \in [0, 1]$

Model training as a parametric optimization

Loss function: $L(\mathbf{W}) = \int l(f(\mathbf{x}), \widetilde{f}(\mathbf{x}, \mathbf{W}))d\mu(\mathbf{x})$

Sample average measure μ :

- $d\mu(\mathbf{x}) = \frac{1}{N}\sum_{n=1}^{N} \delta(\mathbf{x} - \mathbf{x}_n)$ with Dirac's delta — "finite training set" scenario
- $d\mu(\mathbf{x}) = p(\mathbf{x})d\mathbf{x}$ with some (e.g. Gaussian) density $p(\mathbf{x})$ — "population average" scenario

Function $l(f(\mathbf{x}), \widetilde{f}(\mathbf{x}, \mathbf{W}))$ measures the discrepancy between f and $\widetilde{f}$, e.g.:

- Regression: $l(y, \widetilde{y}) = \frac{1}{2}(y - \widetilde{y})^2$
- Classification: $l(y, \widetilde{y}) = -y \log \widetilde{y} - (1 - y)\log(1 - \widetilde{y})$

Model training:

$$L(\mathbf{W}) \longrightarrow \min_{\mathbf{W}}$$

Gradient-based optimization

- $\mathbf{W}$ high-dimensional
- $L(\mathbf{W})$ non-smooth, non-nonconvex

Most popular approach: gradient-based optimization and its modifications

Basic gradient descent with learning rate $\alpha \in (0,1)$:

$$\mathbf{W}^{(n+1)} = \mathbf{W}^{(n)} - \alpha \nabla_{\mathbf{W}} L(\mathbf{W}^{(n)})$$

Gradient descent with Nesterov momentum $(\alpha, \beta \in (0,1))$:

$$\begin{aligned} \mathbf{W}^{(n+1)} &= \mathbf{W}^{(n)} - \mathbf{V}^{(n)} \\ \mathbf{V}^{(n+1)} &= \alpha \nabla_{\mathbf{W}} L(\mathbf{W}^{(n)}) + \beta \mathbf{V}^{(n)} \end{aligned}$$

Exercise: how can we interprete the coefficients α and β?

Computation of $\nabla_{\mathbf{W}} L$: "Error backpropagation"

$$\nabla_{\mathbf{W}} L(\mathbf{W}) = \int \frac{\partial l}{\partial \widetilde{y}}(f(\mathbf{x}), \widetilde{f}(\mathbf{x}, \mathbf{W})) \cdot \nabla_{\mathbf{W}} \widetilde{f}(\mathbf{x}, \mathbf{W}) d\mu(\mathbf{x})$$

$$\nabla_{\mathbf{W}} \widetilde{f} = (\nabla_{\mathbf{w}_1} \widetilde{f}, \ldots, \nabla_{\mathbf{w}_K} \widetilde{f})$$

$\frac{\partial l}{\partial \widetilde{y}}(f(\mathbf{x}), \widetilde{f}(\mathbf{x}, \mathbf{W}))$: directly computed from y and $\widetilde{y}$

$\widetilde{y} = \widetilde{f}(\mathbf{x}, \mathbf{W})$: "forward propagation"

To find $\nabla_{\mathbf{W}} \widetilde{f}$: use layerwise representation

$$\widetilde{f}(\mathbf{x}, \mathbf{W}) = g_K(g_{K-1}(\ldots g_1(\mathbf{x}, \mathbf{w}_1), \ldots \mathbf{w}_{K-1}), \mathbf{w}_K)$$

$\mathbf{z}_k$: output of the k'th layer (known from "forward propagation")

$$\mathbf{z}_k = g_k(\mathbf{z}_{k-1}, \mathbf{w}_k)$$

$$\begin{array}{ccccccccc}
g_K & \longleftarrow & g_{K-1} & \longleftarrow & \cdots & \longleftarrow & g_1 & \longleftarrow & \mathbf{x} \\
\uparrow & & \uparrow & & & & \uparrow & & \\
\mathbf{w}_K & & \mathbf{w}_{K-1} & & & & \mathbf{w}_1 & &
\end{array}$$

"Error backpropagation"

$$\begin{tikzcd} g_K & g_{K-1} \arrow[l] & \cdots \arrow[l] & g_1 \arrow[l] & \mathbf{x} \arrow[l] \\ \mathbf{w}_K \arrow[u] & \mathbf{w}_{K-1} \arrow[u] & & \mathbf{w}_1 \arrow[u] & \end{tikzcd}$$

$$\mathbf{z}_k = g_k(\mathbf{z}_{k-1}, \mathbf{w}_k)$$

$$\begin{aligned}
\nabla_{\mathbf{w}_K} \widetilde{f}(\mathbf{x}, \mathbf{W}) &= \frac{\partial g_K}{\partial \mathbf{w}_K}(\mathbf{z}_{K-1}, \mathbf{w}_K) \\
\nabla_{\mathbf{w}_{K-1}} \widetilde{f}(\mathbf{x}, \mathbf{W}) &= \nabla_{\mathbf{w}_K} g_K(g_{K-1}(\mathbf{z}_{K-2}, \mathbf{w}_{K-1}), \mathbf{w}_K) \\
&= \frac{\partial g_K}{\partial \mathbf{z}_{K-1}}(\mathbf{z}_{K-1}, \mathbf{w}_K) \cdot \frac{\partial g_{K-1}}{\partial \mathbf{w}_{K-1}}(\mathbf{z}_{K-2}, \mathbf{w}_{K-1}) \\
&\quad \cdots \\
\nabla_{\mathbf{w}_k} \widetilde{f}(\mathbf{x}, \mathbf{W}) &= \frac{\partial g_K}{\partial \mathbf{z}_{K-1}}(\mathbf{z}_{K-1}, \mathbf{w}_K) \cdots \frac{\partial g_{k+1}}{\partial \mathbf{z}_k}(\mathbf{z}_k, \mathbf{w}_{k+1}) \\
&\quad \cdot \frac{\partial g_k}{\partial \mathbf{w}_k}(\mathbf{z}_{k-1}, \mathbf{w}_k)
\end{aligned}$$

Basic theory: convergence of gradient descent

See e.g. Yu. Nesterov, Introductory Lectures on Convex Programming Volume I: Basic course.

- **Global minima**: $L(\mathbf{W}_*) = \min_{\mathbf{W}\in\mathbb{R}^W} L(\mathbf{W})$
- **Local minima**: $L(\mathbf{W}_*) = \min_{\mathbf{W}\in U} L(\mathbf{W})$, where U is an open neighborhood of $\mathbf{W}_*$
- **Stationary points**: $\nabla_{\mathbf{W}} L(\mathbf{W}_*) = 0$ (assuming $L(\mathbf{W})$ is smooth)

In general, gradient descent converges to a stationary point.

Proposition

Suppose function L is lower bounded and $L \in \mathcal{W}^{2,\infty}(\mathbb{R}^W)$, so that $|\nabla L(\mathbf{a}) - \nabla L(\mathbf{b})| \le M|\mathbf{a}-\mathbf{b}|$ with some Lipschitz constant M. Let $\alpha < \frac{2}{M}$. Then $\nabla L(\mathbf{W}^{(n)}) \to 0$, and $\min_{n=1,\ldots,N} |\nabla L(\mathbf{W}^{(n)})| = O(N^{-1/2})$.

Proof

$$
\begin{aligned}
L(\mathbf{W}^{(n+1)}) &= L(\mathbf{W}^{(n)}) + \left\langle \mathbf{W}^{(n+1)} - \mathbf{W}^{(n)}, \int_0^1 \nabla L(\mathbf{W}^{(n)} + t(\mathbf{W}^{(n+1)} - \mathbf{W}^{(n)}))dt \right\rangle \\
&\le L(\mathbf{W}^{(n)}) + \left\langle \mathbf{W}^{(n+1)} - \mathbf{W}^{(n)}, \nabla L(\mathbf{W}^{(n)}) \right\rangle \\
&\quad + |\mathbf{W}^{(n+1)} - \mathbf{W}^{(n)}| \int_0^1 Mt|\mathbf{W}^{(n+1)} - \mathbf{W}^{(n)}|dt \\
&\le L(\mathbf{W}^{(n)}) + \left\langle \mathbf{W}^{(n+1)} - \mathbf{W}^{(n)}, \nabla L(\mathbf{W}^{(n)}) \right\rangle + \frac{M}{2}|\mathbf{W}^{(n+1)} - \mathbf{W}^{(n)}|^2 \\
&\le L(\mathbf{W}^{(n)}) + (-\alpha + \frac{M}{2}\alpha^2)|\nabla L(\mathbf{W}^{(n)})|^2 \\
&< L(\mathbf{W}^{(n)})
\end{aligned}
$$

if $\alpha < \frac{2}{M}$. Let $c = \alpha(1 - \frac{M}{2}\alpha) > 0$, then

$$
\sum_{n=1}^N |\nabla L(\mathbf{W}^{(n)})|^2 \le \frac{1}{c} \sum_{n=1}^N (L(\mathbf{W}^{(n)}) - L(\mathbf{W}^{(n+1)})) \le \frac{1}{c}(L(\mathbf{W}^{(1)}) - \min_{\mathbf{W}} L(\mathbf{W})),
$$

$$
\min_{n=1,\ldots,N} |\nabla L(\mathbf{W}^{(n)})| \le \frac{c^{-1/2}}{\sqrt{N}} \left(L(\mathbf{W}^{(1)}) - \min_{\mathbf{W}} L(\mathbf{W}) \right)^{1/2}
$$

Formulation in terms of stopping condition

Assume the **stopping condition**: $|\nabla L(\mathbf{W}^{(n)})| < \epsilon$.

Then, optimization terminates in $O\big(\frac{L(\mathbf{W}^{(1)})-\min_{\mathbf{W}} L(\mathbf{W})}{\epsilon^2}\big)$ steps.

Exercise: What is the optimal value of α, assuming M is known?

Exercise: Give an example of gradient descent converging to a stationary point which is not a (local or global) minimum.

Linearization and spectral analysis

Suppose that $L \in C^2(\mathbb{R})$ and $\mathbf{W}_*$ is a stationary point.
For $\mathbf{W}$ near $\mathbf{W}_*$:

$$\nabla L(\mathbf{W}) = D^2 L(\mathbf{W}_*) \cdot (\mathbf{W} - \mathbf{W}_*) + o(|\mathbf{W} - \mathbf{W}_*|),$$

where $D^2 L(\mathbf{W}_*)$ is the Hessian matrix. Optimization iterates:

$$\begin{aligned}
\mathbf{W}^{(n+1)} - \mathbf{W}_* &= \mathbf{W}^{(n)} - \mathbf{W}_* - \alpha \nabla L(\mathbf{W}^{(n)}) \\
&= \mathbf{W}^{(n)} - \mathbf{W}_* - \alpha D^2 L(\mathbf{W}_*) \cdot (\mathbf{W}^{(n)} - \mathbf{W}_*) + o(|\mathbf{W}^{(n)} - \mathbf{W}_*|) \\
&= (1 - \alpha D^2 L(\mathbf{W}_*)) \cdot (\mathbf{W}^{(n)} - \mathbf{W}_*) + o(|\mathbf{W}^{(n)} - \mathbf{W}_*|)
\end{aligned}$$

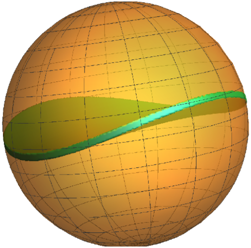
Convergence is determined by eigenvalues of $D^2 L(\mathbf{W}_*)$:

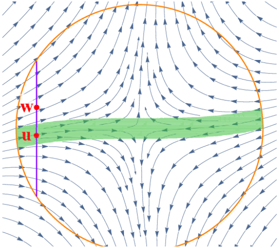
- positive: convergence
- negative: divergence

Evasion of saddle points

Saddle points: $D^2L(\mathbf{W}_*)$ has both positive and negative eigenvalues

Typically, saddles are evaded by optimization, due to the presence of diverging components in $\mathbf{W}^{(n)} - \mathbf{W}_*$. The manifold of converging $\mathbf{W}^{(n)}$ has Lebesgue measure 0.[1]



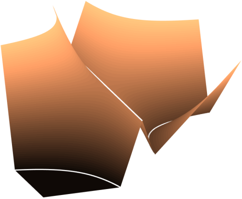


Chi Jin and M. Jordan, How to Escape Saddle Points Efficiently: Saddle points can slow down optimization; perturbing the GD can help.

[1] B. Recht, Saddles Again

Real-life ANNs

- No smoothness, in general (e.g. with ReLU): local minima of $L(\mathbf{W})$ are non-differentiable



Th. Laurent, J. von Brecht, The Multilinear Structure of ReLU Networks, arXiv:1712.10132

- Large size of the network and its structure are important

Empirical observations of real-life ANNs

From A. Choromanska et al., The Loss Surfaces of Multilayer Networks, arXiv:1412.0233:

- Large networks train well despite their size. Optimization can terminate at different local minima, but they seem to be equivalent and yield similar performance on a test set.
- The probability of finding a bad (high value) local minimum is non-zero for small-size networks and decreases quickly with network size.
- Struggling to find the global minimum on the training set (as opposed to one of the many good local ones) is not useful in practice and may lead to overfitting.

Conceptual pictures of the loss surface (conjectured)

From M. Baity-Jesi, Comparing Dynamics: Deep Neural Networks versus Glassy Systems: two alternatives

1. The loss landscape is very rough, has many isolated local minima, but GD tends to find good minima having low loss.
2. The loss function is highly nonlinear, but has few local minima, and the minima are connected. (Example: $L(w_1, w_2) = (w_2 - w_1^2)^2 + \epsilon w_1^2$.)

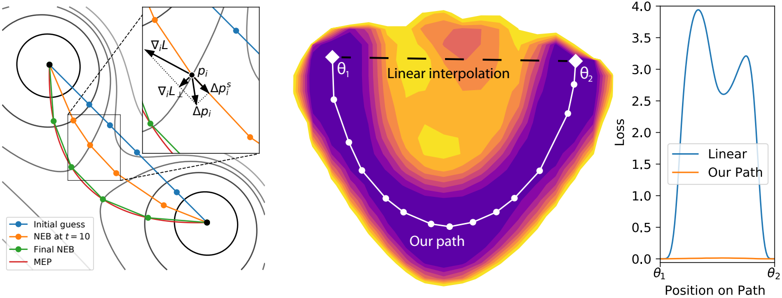
(Note: these conteptual pictures have only a limited value due to the "curse of dimensionality" in $\mathbb{R}^W$, lack of characterization of locality and depth of a local minimum, etc.)

Some current research directions

- Numerical studies of loss surface and gradient descent
- Direct analytic studies of simple (toy) scenarios:
  - Deep linear networks (no nonlinear activation)
  - Wide shallow networks with small training sets, pyramidal networks (no spurious local minima)
- Large-size limits:
  - Large-width limit: Gaussian approximation for signal propagation, connections to random matrices and spherical spin glasses
  - Phenomenological: Stochastic PDE and Langevin dynamics
- Specialized networks (e.g., convnets)

F. Draxler et al., Essentially No Barriers in Neural Network Energy Landscape, arXiv:1803.00885

- Two local minima are connected by a path, and then it is deformed to find a low loss trajectory
- ResNets and DenseNets on CIFAR10 and CIFAR100
- The optimized path: approximately constant loss



(Global description of the optimal set?)

I. Safran, O. Shamir, Spurious Local Minima are Common in Two-Layer ReLU Neural Networks, arXiv:1712.08968

Spurious local minimum W_0: $\min_{\mathbf{W}} L(\mathbf{W}) < L(\mathbf{W}_0) < L(\mathbf{W}')$ for $\mathbf{W}'$ in a small neighborhood of $\mathbf{W}_0$

Theorem

Consider the optimization problem

$$\min_{\mathbf{w}_1,\ldots,\mathbf{w}_n\in\mathbb{R}^k} \mathbb{E}_{\mathbf{x}\sim\mathcal{N}(\mathbf{0},I)}\Big(\sum_{i=1}^n(\mathbf{w}_i^\top\mathbf{x})_+ - \sum_{i=1}^k(\mathbf{v}_i^\top\mathbf{x})_+\Big)^2,$$

where $\mathbf{v}_1,\ldots,\mathbf{v}_k$ are orthogonal unit vectors in $\mathbb{R}^k$. Then for $n=k\in\{6,7,\ldots,20\}$ as well as $(k,n)\in\{(8,9),(10,11),\ldots,(19,20)\}$, this objective function has spurious local minima.

Proof: computer-assisted

Dependence on n, k

More spurious minima observed at larger k, but overparametrization (large n) appears to partly remove them.

Table: Spurious local minima found for $n = k$

k	n	% of runs converging to local minima	Average minimal eigenvalue	Average objective value
6	6	0.3%	0.0047	0.025
7	7	5.5%	0.014	0.023
8	8	12.6%	0.021	0.021
9	9	21.8%	0.027	0.02
10	10	34.6%	0.03	0.022
11	11	45.5%	0.034	0.022
12	12	58.5%	0.035	0.021
13	13	73%	0.037	0.022
14	14	73.6%	0.038	0.023
15	15	80.3%	0.038	0.024
16	16	85.1%	0.038	0.027
17	17	89.7%	0.039	0.027
18	18	90%	0.039	0.029
19	19	93.4%	0.038	0.031
20	20	94%	0.038	0.033

Table: Spurious local minima found for $n \neq k$

k	n	% of runs converging to local minima	Average minimal eigenvalue	Average objective value
8	9	0.1%	0.0059	0.021
10	11	0.1%	0.0057	0.018
11	12	0.1%	0.0056	0.017
12	13	0.3%	0.0054	0.016
13	14	1.5%	0.0015	0.038
14	15	5.5%	0.002	0.033
15	16	10.1%	0.004	0.032
16	17	18%	0.0055	0.031
17	18	20.9%	0.007	0.031
18	19	36.9%	0.0064	0.028
19	20	49.1%	0.0077	0.027