Foundations of Deep Learning

Lecture 10

Regularization

Aurelien Lucchi

Fall 2024

Today

Perspective of generalization in ML:

- Over-parametrized models overfit to the training set.
- We will see that this picture is somewhat more complicated (Double Descent phenomenon)

One technique that is commonly used to avoid/control over-fitting is that of **regularization**:

- Classical approach: penalty term on the norm of the weights
- Implicit bias of gradient descent
- Dropout, consists of randomly dropping neurons by sampling them in an i.i.d. fashion from a Bernoulli distribution

Over-parametrized Models

Classical textbook: Over-parametrized models overfit
$\rightsquigarrow$ low training error and large test error.

Deep Learning Models:

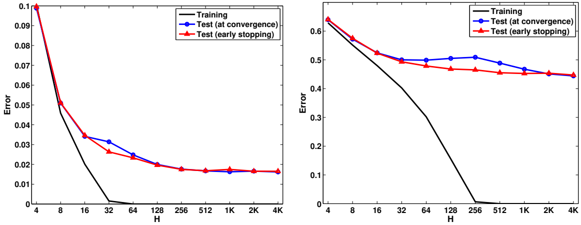


Figure: Training and test error for 2-layer NNs with different number of hidden units trained on MNIST and CIFAR-10 [NTS14].

While the training error reaches zero, the test error keeps decreasing (no sign of overfitting). $\rightsquigarrow$ It seems that **some sort of regularization is at play!?**

Section 1

Implicit Bias of Gradient Descent (GD)

Linear Regression

Consider a training dataset $D = (\mathbf{x}_i, y_i)_{i=1}^{n}$ where $\mathbf{x}_i \in \mathbb{R}^d$ are some given features and $y_i \in \{-1, +1\}$ are the corresponding labels. The least-squares objective is then defined as:

$$\mathcal{L}(\mathbf{w}) = \frac{1}{n} \sum_{i=1}^{n} (y_i - \mathbf{w}^T \mathbf{x}_i)^2,$$

where $\mathbf{w} \in \mathbb{R}^d$. We can also rewrite the equation above in a vector form as

$$\mathcal{L}(\mathbf{w}) = \frac{1}{n} \left\| \mathbf{y} - \mathbf{X}\mathbf{w} \right\|^2,$$

where $\mathbf{y} = [y_1 \dots y_n] \in \mathbb{R}^n$ and $\mathbf{X}$ is matrix whose rows are the $\mathbf{x}_i' s$, i.e. $\mathbf{X} = [\mathbf{x}_1 \dots \mathbf{x}_n]^\top \in \mathbb{R}^{n \times d}$.

Linear Regression

Minimum-norm solution found by gradient descent

Assume that $d \gg n$ (over-parametrized setting), then many $\mathbf{w}$'s satisfy $\mathbf{X}\mathbf{w} = \mathbf{y}$ and all have $L(\mathbf{w}) = 0$ and are global minima of the problem. Which solution do we converge to?

Theorem 1 ([GLSS18])
Consider the iterate of gradient descent on the least-square regression loss. Then, if $d > n$,

$$\mathbf{w}_t \to \underset{\mathbf{w}|\mathbf{X}\mathbf{w}=\mathbf{y}}{\operatorname{argmin}} \|\mathbf{w} - \mathbf{w}_0\|_2.$$

Linear Regression

Minimum-norm solution found by gradient descent

Assume that $d \gg n$ (over-parametrized setting), then many $\mathbf{w}$'s satisfy $\mathbf{X}\mathbf{w} = \mathbf{y}$ and all have $L(\mathbf{w}) = 0$ and are global minima of the problem. Which solution do we converge to?

Theorem 1 ([GLSS18])
Consider the iterate of gradient descent on the least-square regression loss. Then, if $d > n$,

$$\mathbf{w}_t \to \underset{\mathbf{w}|\mathbf{X}\mathbf{w}=\mathbf{y}}{\operatorname{argmin}} \|\mathbf{w} - \mathbf{w}_0\|_2.$$

- If $\mathbf{w}_0 = 0$, $\mathbf{w}$ converges to the minimum norm solution.
- In general, $\mathbf{w}$ converges to the global minimum which is the closest to the initialization in terms of ℓ_2 distance.

Proof

Logistic regression

- Regression setting: gradient descent converges to the minimum-norm solution
- Next, we will see that in a classification setting (logistic regression), gradient descent with arbitrary initialization converges to the **maximum margin classifier** on separable data.

Logistic regression

Assumption 1

The dataset is linearly separable:
$\exists \mathbf{w}_*$ *such that* $\forall n: \ \mathbf{w}_*^\top \mathbf{x}_n > 0$.

Assumption 2

$\ell(u)$ *is a positive, differentiable, monotonically decreasing to zero, (so* $\forall u: \ \ell(u) > 0, \ell'(u) < 0$, $\lim_{u\to\infty} \ell(u) = \lim_{u\to\infty} \ell'(u) = 0$*), a* β*-smooth function, i.e. its derivative is* β*-Lipshitz and* $\lim_{u\to-\infty} \ell'(u) \neq 0$.

Assumption 2 includes many common loss functions, including the logistic, exp-loss, ...

Logistic regression

Main question: Can we characterize the direction in which $\mathbf{w}(t)$ diverges?
That is, does the limit $\lim_{t\to\infty} \mathbf{w}(t) / \|\mathbf{w}(t)\|$ always exist, and if so, what is it?

In order to analyze this limit, we will need to make a further assumption on the tail of the loss function...

Logistic regression

Assumption 3 (Informal, see notes)

The negative loss derivative $-\ell'(u)$ has a tight exponential tail

Examples Exponential loss $\ell(u) = e^{-u}$ and logistic loss $\ell(u) = \log(1 + e^{-u})$ (where $u = \mathbf{w}^\top \mathbf{x}_i$) both follow this assumption with $a = c = 1$.

Logistic loss: observe lower values of the loss for larger values of $u = \mathbf{w}^\top \mathbf{x}_i$. This intuitively explains why gradient descent enforces $\mathbf{w}(t) \to \infty$ in order to minimize the loss.

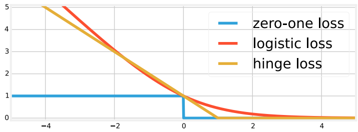


Figure: Source: `https://fa.bianp.net`.

Review: Karush-Kuhn-Tucker (KKT) Conditions

Consider the following constrained optimization problem:

$$\begin{aligned} \text{Minimize} \quad & f(\mathbf{x}) \\ \text{subject to} \quad & g_i(\mathbf{x}) \leq 0, \quad i = 1, 2, \ldots, m \\ & h_j(\mathbf{x}) = 0, \quad j = 1, 2, \ldots, p \end{aligned}$$

where $\mathbf{x}$ is the vector of decision variables, $f(\mathbf{x})$ is the objective function, $g_i(\mathbf{x})$ are inequality constraints, and $h_j(x)$ are equality constraints.

The KKT conditions are conditions that characterize optimality of the above problem.

Review: (linear) Support Vector Machine

Classification setting We are given a training dataset of n training points $\{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$ where $y_i \in \{-1, +1\}$.

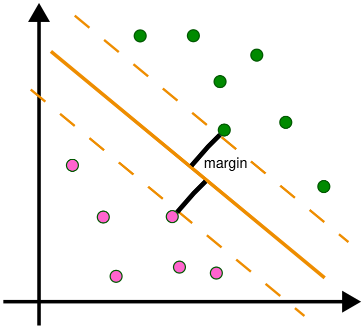
SVM solution:

Linear SVM solves the problem:

$$\underset{\mathbf{w},\, b}{\text{minimize}} \quad \|\mathbf{w}\|_2^2$$

$$\text{subject to} \quad y_i(\mathbf{w}^\top \mathbf{x}_i - b) \geq 1$$

$$\forall i \in \{1, \ldots, n\}$$



Main theorem

Theorem 2 ([SHN+18])

Under Assumptions 1-3, any stepsize $\eta < 2\beta^{-1}\sigma_{\max}^{-2}(\mathbf{X})$ *and any* $\mathbf{w}(0)$*, the GD iterates will behave as:*

$$\mathbf{w}(t) = \hat{\mathbf{w}} \log t + \boldsymbol{\rho}(t) \ ,$$

where $\hat{\mathbf{w}}$ *is the* L_2 *max margin vector:*

$$\hat{\mathbf{w}} = \underset{\mathbf{w}\in\mathbb{R}^d}{\operatorname{argmin}} \|\mathbf{w}\|^2 \ \text{s.t.}\ \mathbf{w}^\top \mathbf{x}_n \geq 1,$$

and the residual grows at most as $\|\boldsymbol{\rho}(t)\| = \mathcal{O}(\log\log(t))$*, and so*

$$\lim_{t\to\infty} \frac{\mathbf{w}(t)}{\|\mathbf{w}(t)\|} = \frac{\hat{\mathbf{w}}}{\|\hat{\mathbf{w}}\|}.$$

Furthermore, for almost all data sets (all except measure zero), the residual $\rho(t)$ *is bounded.*

Proof idea

Convergence rate

Asymptotically: converge to the max-margin SVM solution.

What about the rate of convergence?

It turns out to be a very slow rate equal to

$$\left\| \frac{\mathbf{w}(t)}{\|\mathbf{w}(t)\|} - \frac{\hat{\mathbf{w}}}{\|\hat{\mathbf{w}}\|} \right\| = \mathcal{O}\left(\frac{1}{\log t}\right).$$

In contrast, the rate of convergence of the loss is of the order $\mathcal{O}\left(\frac{1}{t}\right)$, which is a much faster rate.

Section 2

Implicit Bias of Gradient Descent: Extension to Neural Networks

Setting

Consider depth-2 ReLU neural networks

$$f_{\boldsymbol{\theta}}(\mathbf{x}) = \sum_{j\in[m]} v_j \phi(\mathbf{w}_j^\top \mathbf{x} + b_j) \quad \phi(z) = \max\{0, z\}.$$

Homogeneous networks We say that a network is **homogeneous** if there exists $L > 0$ such that for every $\alpha > 0$ and $\boldsymbol{\theta}, \mathbf{x}$ we have $f(\alpha\boldsymbol{\theta}; \mathbf{x}) = \alpha^L f(\boldsymbol{\theta}; \mathbf{x})$
$\rightsquigarrow$ Note that depth-2 ReLU networks as defined above are homogeneous (with $L = 2$).

Setting

Let $S = \{(\mathbf{x}_i, y_i)\}_{i=1}^n \subseteq \mathbb{R}^d \times \{-1, 1\}$ be a binary classification training dataset. Let $f(\boldsymbol{\theta}; \cdot) : \mathbb{R}^d \to \mathbb{R}$ be a neural network parameterized by $\boldsymbol{\theta}$.

Empirical loss For a loss function $\ell : \mathbb{R} \to \mathbb{R}$ the empirical loss of $f(\boldsymbol{\theta}; \cdot)$ on the dataset S is

$$\mathcal{L}(\boldsymbol{\theta}) := \frac{1}{n} \sum_{i=1}^{n} \ell(y_i f(\boldsymbol{\theta}; \mathbf{x}_i)) \ . \tag{1}$$

Next, we consider minimizing either the exponential or the logistic loss over a binary classification dataset $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$ using gradient flow.

Main result

Theorem 3 (Paraphrased from [LL19, JT20])

Assume that there exists time t_0 such that $\mathcal{L}(\boldsymbol{\theta}(t_0)) < \frac{1}{n}$ (and thus $y_i f(\boldsymbol{\theta}(t_0); \mathbf{x}_i) > 0$ for every $\mathbf{x}_i$). Then, gradient flow converges in direction to a first-order stationary point (KKT point) of the following maximum margin problem in parameter space:

$$\min_{\boldsymbol{\theta}} \frac{1}{2} \|\boldsymbol{\theta}\|^2 \quad \textit{s.t.} \quad \forall i \in [n] \;\; y_i f(\boldsymbol{\theta}; \mathbf{x}_i) \geq 1 \ . \tag{2}$$

Moreover, $\mathcal{L}(\boldsymbol{\theta}(t)) \to 0$ and $\|\boldsymbol{\theta}(t)\| \to \infty$ as $t \to \infty$.

Section 3

Dropout

Dropout: main idea

Dropout is a popular regularization technique for neural networks proposed by [HSK$^+$12].

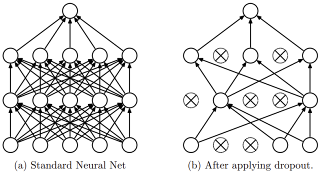


Figure: Figure from [SHK$^+$14].

Key idea: randomly "drop" subsets of units in the network.

More precisely: Define a "keep" probability π_i^l for each unit i in the layer l of the network.

How? This can be realized by sampling bit mask and zeroing out activations, so that standard backpropagation applies.

Dropout: intuition

Original motivation [HSK+12]:
... "overfitting is greatly reduced by randomly omitting half of the feature detectors... This prevents complex co-adaptations in which a feature detector is only helpful in the context of several other specific feature detectors. Instead, each neuron learns to detect a feature that is generally helpful for producing the correct answer given the combinatorially large variety of internal contexts in which it must operate".

Another intuition:
Dropout can be viewed as training an ensemble of multiple models, where each model is a subset of the full network with different sets of neurons active.

Analysis: the regression case

Interpretation: Dropout is equivalent to adding a regularization term to the loss function.

Model: Non-linear neural networks with k layers:

$$f_{\mathbf{W}}(\mathbf{x}) = \mathbf{W}_k \sigma(\mathbf{W}_{k-1} \sigma(\dots \mathbf{W}_2 \sigma(\mathbf{W}_1 \mathbf{x}) \dots)),$$

where $\mathbf{W}_i \in \mathbb{R}^{d_i \times d_{i-1}}$ are the weight matrices, $\mathbf{x} \in \mathbb{R}^{d_0}$ is the input and $\sigma(\cdot)$ is a entrywize activation function.

Simplification: Consider the case where we apply Dropout to the top layer of the network. Let $\mathbf{B}$ be a diagonal random matrix with i.i.d. diagonal elements drawn from a Bernoulli distribution, i.e. $B_{ii} \sim \frac{1}{1-p}\text{Ber}(1-p), i \in [d_{k-1}]$ for some dropout rate p.

Analysis: the regression case

Setting: Given training examples $(\mathbf{x}_i, \mathbf{y}_i)_{i=1}^n$ where each $(\mathbf{x}_i, \mathbf{y}_i) \sim \mathcal{D} = \mathcal{X} \times \mathcal{Y}$ and a loss function $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$.
$\longrightarrow$ Want to find $\mathbf{W}^* = \operatorname{argmin}_{\mathbf{W}} \mathbb{E}_D[\ell(f_{\mathbf{W}}(\mathbf{x}), \mathbf{y})]$.

Loss

$$\mathcal{L}(\mathbf{W}) := \frac{1}{n} \sum_{i=1}^{n} \|\mathbf{y}_i - \mathbf{W}_k \sigma(\mathbf{W}_{k-1} \sigma(\ldots \mathbf{W}_2 \sigma(\mathbf{W}_1 \mathbf{x}_i)\|^2.$$

$$\mathcal{L}_{Drop}(\mathbf{W}) := \frac{1}{n} \sum_{i=1}^{n} \mathbb{E}_{\mathbf{B}} \|\mathbf{y}_i - \mathbf{W}_k \mathbf{B} \sigma(\mathbf{W}_{k-1} \sigma(\ldots \mathbf{W}_2 \sigma(\mathbf{W}_1 \mathbf{x}_i)\|^2.$$

Goal Understand the explicit regularization introduced by Dropout.

Analysis: the regression case

Notation Let $a_{i,j} \in \mathbb{R}$ denote the output of the i-th hidden node in the j-th hidden layer on an input vector $\mathbf{x}$. Also let vector $\mathbf{a}_j \in \mathbb{R}^{d_j}$ denote the activation of the j-th later on the input $\mathbf{x}$.

We can then rewrite the Dropout objective as

$$\mathcal{L}_{Drop}(\mathbf{W}) := \frac{1}{n}\sum_{i=1}^{n} \mathbb{E}_{\mathbf{B}} \|\mathbf{y}_i - \mathbf{W}_k \mathbf{B} \mathbf{a}_{k-1}(\mathbf{x}_i)\|^2.$$

Analysis: the regression case

Dropout objective $\mathcal{L}_{Drop}(\mathbf{W})$ is a regularized version of the original objective $\mathcal{L}(\mathbf{W})$

Theorem 4 (Dropout regularizer in deep regression)

The population risk in the Dropout scenario can be written as

$$\mathcal{L}_{Drop}(\mathbf{W}) = \mathcal{L}(\mathbf{W}) + R(\mathbf{W}),$$

where $R(\mathbf{W})$ *is a regularizer term defined as*

$$R(\mathbf{W}) = \lambda \sum_{j=1}^{d_{k-1}} \|\mathbf{W}_k(:,j)\|^2 \hat{a}_j^2,$$

where $\hat{a}_j = \left(\frac{1}{n}\sum_{i=1}^{n} a_{j,k-1}(\mathbf{x}_i)^2\right)^{1/2}$ *and* $\lambda = \frac{p}{1-p}$.

Proof

Suriya Gunasekar, Jason Lee, Daniel Soudry, and Nathan Srebro.

Characterizing implicit bias in terms of optimization geometry.

In *International Conference on Machine Learning*, pages 1832–1841. PMLR, 2018.

Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov.

Improving neural networks by preventing co-adaptation of feature detectors.

arXiv preprint arXiv:1207.0580, 2012.

Ziwei Ji and Matus Telgarsky.

Directional convergence and alignment in deep learning.

Advances in Neural Information Processing Systems, 33:17176–17186, 2020.

Kaifeng Lyu and Jian Li.

Gradient descent maximizes the margin of homogeneous neural networks.

arXiv preprint arXiv:1906.05890, 2019.

Behnam Neyshabur, Ryota Tomioka, and Nathan Srebro.
In search of the real inductive bias: On the role of implicit regularization in deep learning.
arXiv preprint arXiv:1412.6614, 2014.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov.
Dropout: a simple way to prevent neural networks from overfitting.
The journal of machine learning research, 15(1):1929–1958, 2014.

Daniel Soudry, Elad Hoffer, Mor Shpigel Nacson, Suriya Gunasekar, and Nathan Srebro.
The implicit bias of gradient descent on separable data.
The Journal of Machine Learning Research, 19(1):2822–2878, 2018.