Theoretical Principles of Deep Learning

Class III: Lazy training

Hédi Hadiji

Université Paris-Saclay - CentraleSupelec

hedi.hadiji@l2s.centralesupelec.fr

December 18th, 2023

Table of Contents

Plan

Last time: Approximation

- 2-layer nets with enough nodes are universal approximators
- Smooth functions (in the sense of Barron) can be well approximated by small(-ish) nets, even for high-dim inputs

Today: Optimization. Understanding the behavior of neural nets under gradient descent.

Reading Material:

- *On Lazy Training in Differentiable Programming*, Chizat, Oyallon and Bach '19

Table of Contents

ERM and SGD

Fix a set of hypotheses, parameterized by some $w \in \mathbb{R}^p$.

Given a dataset, find a model that minimizes the empirical loss

$$\arg\min_{w \in \mathbb{R}^p} F(w) := \frac{1}{n} \sum_{i=1}^{n} \ell(h(x_i; w), y_i) \,,$$

using (Stochastic) Gradient Descent.

This works well in practice for neural nets. **Why?**

Plan for the day

Bringing **partial** answer to why (S)GD works with neural nets

- Describe a typical cases we understand: linear models
- Identify the Lazy Regime (i.e. sets of dimensions, or parameters) in which neural nets can be seen as almost linear models

Table of Contents

Linear models

Linear models

Fix a feature map $\phi : \mathcal{X} \to \mathbb{R}^p$ for some $p \in \mathbb{N}$, and consider the corresponding set of linear models

$$\mathcal{H} = \left\{ h(\,\cdot\,; w) : x \mapsto w^\top \phi(x) \right\}.$$

Important Models are linear in w, not in x. In fact, this makes sense even if x is not an element of a vector space.

Convex losses

If the model is linear and the loss is convex then the ERM objective is a convex function of w,

$$F(w) = \frac{1}{n} \sum_{i=1}^{n} \ell\left(w^\top \phi(x_i), y_i\right)$$

Convex functions are easy to optimize

Gradient descent for convex optimization

Let $F : \mathbb{R}^p \to \mathbb{R}$ be a convex differentiable function. Gradients point in the direction where the function is locally growing the fastest. To minimize, move in opposite direction.

$$\nabla F(w) = \left(\frac{\partial F}{\partial w_i}(w)\right)$$

Gradient descent

Fix a learning rate (or step-size) $\eta > 0$, an initialization point w_0. Gradient descent generates the sequence of points defined by

$$w_{t+1} = w_t - \eta \nabla F(w_t)\,.$$

Stochastic Gradient Descent

Sometimes, only have access to noisy estimates of the gradient

$$w_{t+1} = w_t - \eta\, g_t \quad \text{where} \quad g_t \quad \text{is such that} \quad \mathbb{E}[g_t] = \nabla F(w_t)\,.$$

Example (SGD, sampling with replacement for ERM)

$$F(w) = \frac{1}{n}\sum_{i=1}^{n} \ell_i(w)$$

Pick I_t at random in $\{1, \dots, n\}$ and $g_t = \nabla \ell_{I_t}(w_t)$, then

$$\mathbb{E}[g_t] = \sum_{i=1}^{n} \nabla \ell_i(w_t)\mathbb{P}[I_t = i] = \frac{1}{n}\sum_{i=1}^{n} \nabla \ell_i(w_t) = \nabla\left(\frac{1}{n}\sum_{i=1}^{n} \ell_i\right)(w_t) = \nabla F(w_t)$$

Remark: SGD can also be interpreted as directly minimizing the true average loss $\mathbb{E}[\ell(h(X, w), Y)]$

GD: Typical behavior for convex and smooth objectives

If F is smooth and the learning rate small enough, GD will converge to a critical point $\nabla F(w^\star) = 0$. If function is convex, critical points are global optima.

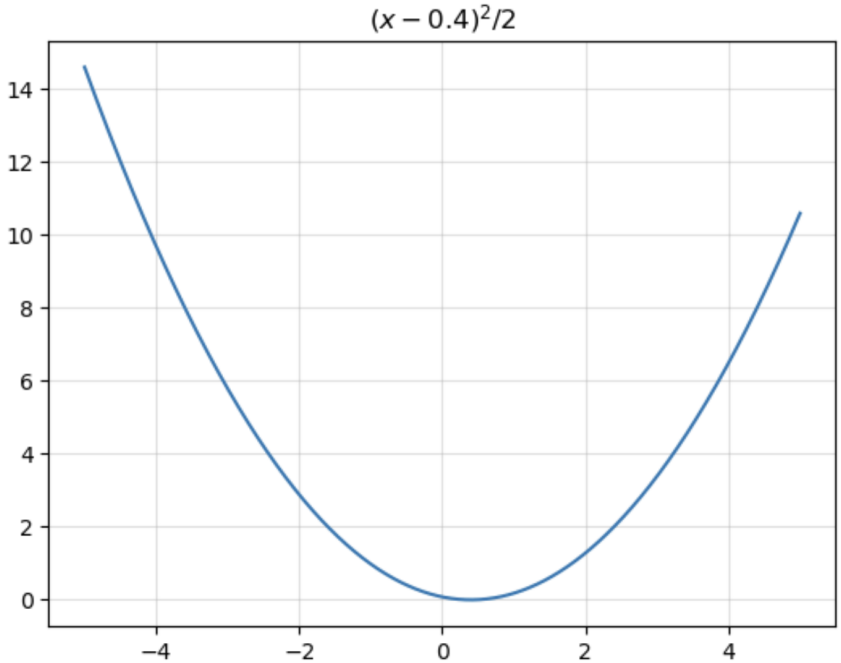
Regardless of initialization:

- reasonable η function value will decrease to the minimum
- too small η small steps, not moving much, slow convergence
- too large η big steps, instability, divergence or oscillations

Could do a (multiple) whole course on convex optimization but let us just look at important examples.

GD examples. Curvature and smoothness.

$$f(w) = \frac{1}{2}(w - w^\star)^2$$



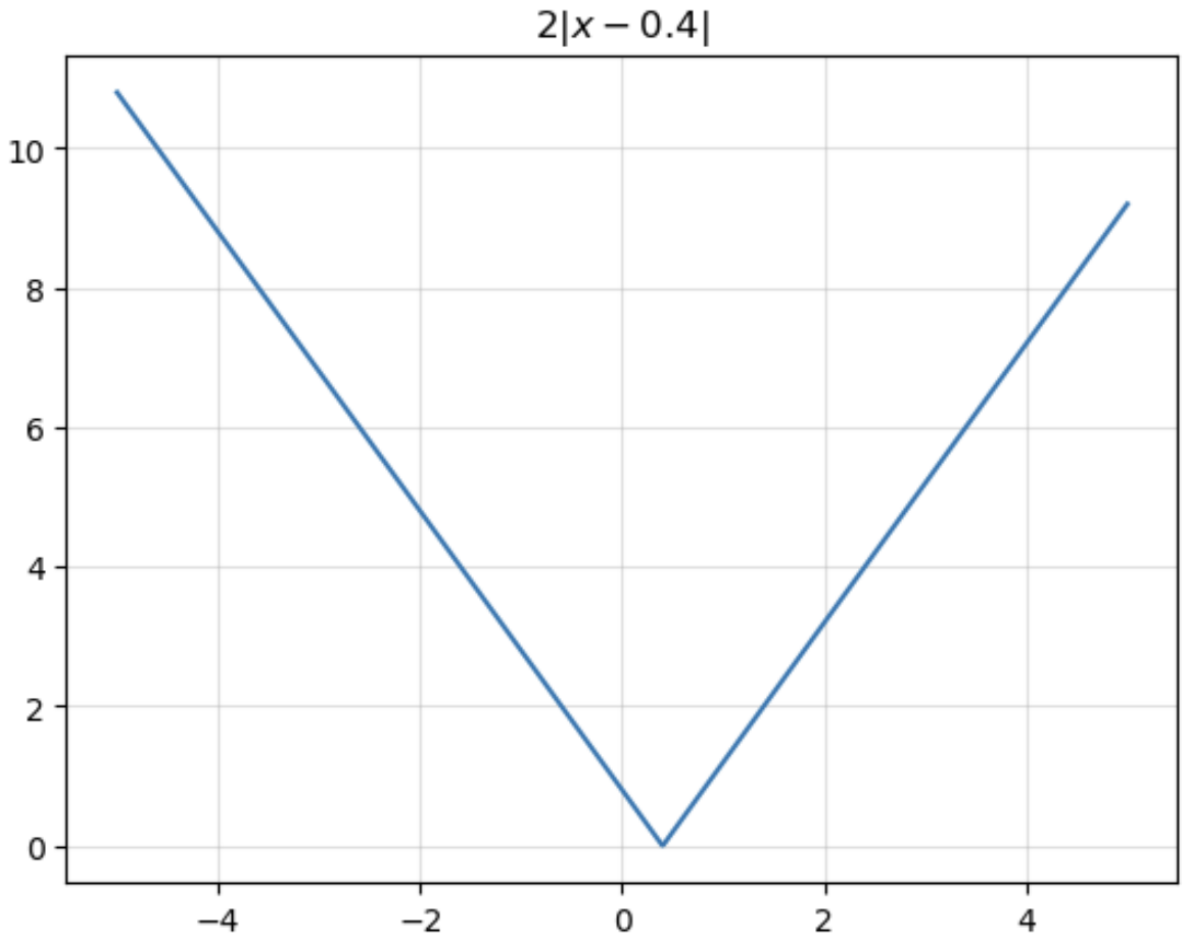
$$w_{t+1} = w_t - \eta(w_t - w_\star) \quad \text{so} \quad w_{t+1} - w_\star = (1-\eta)(w_t - w_\star)$$

- Exponential convergence iff $0 < \eta < 2$
- Exponential divergence if $\eta > 2$

GD examples: Flat and not smooth

Absolute value: flat and not smooth at the optimum

$$f(w) = G|w - w^\star|$$



GD examples: Flat and not smooth II

$$w_t = \begin{cases} w_t - \eta G & \text{if} \quad w_t > w^\star \\ w_t + \eta G & \text{if} \quad w_t < w^\star \end{cases}$$

Move to a small neighborhood of $w^\star$, then oscillate around $w^\star$.

To find w such that $f(w) - f(w^\star) \leqslant \varepsilon$, pick $\eta = G/\varepsilon$ and wait for $T = |w_0 - w^\star| G/\varepsilon$ steps.

Exercise: think about Huber loss and local vs. global effects of curvature and smoothness

$$F(w) = \begin{cases} \dfrac{|w|^2}{2} & \text{if} \quad |w| \leqslant 1 \\ |w| - \dfrac{1}{2} & \text{if} \quad |w| > 1 \end{cases}$$

GD: Fundamental examples

Quadratic:

$$f(w) = \frac{1}{2}(w - w^\star)^\top M(w - w^\star), \quad M \text{ positive definite}$$

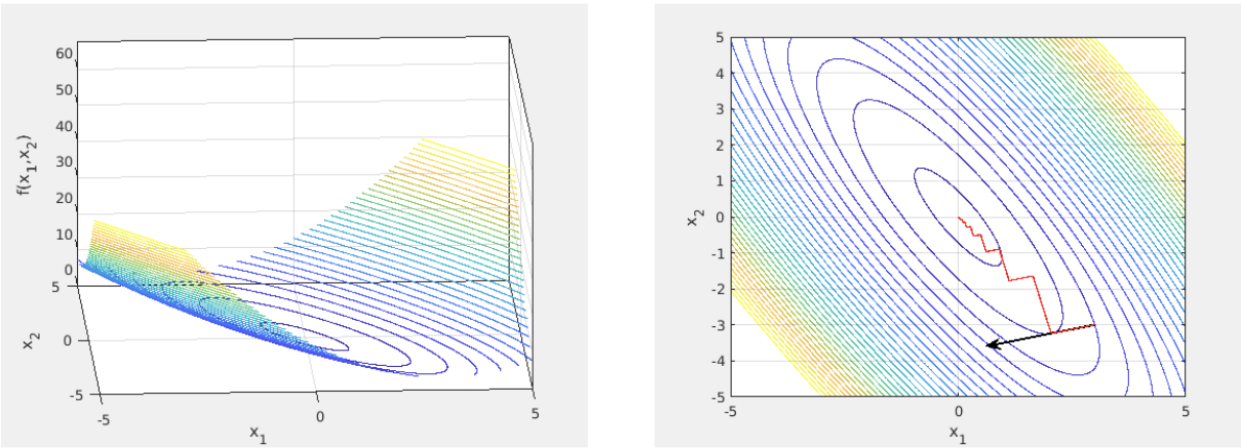


Figure 2. Gradient, level set, and behavior of GD.

Source: Francesco Orabona's blog

Speed of convergence is determined by the conditioning of the quadratic, i.e. how 'flat' the level sets are.

Gradient descent

Exercise: Considering both $\|w - w^\star\|$ and $F(w) - F(w^\star)$. Describe the behavior of the iterates of gradient descent on the functions

a)

$$f(x, y) = \frac{x^2}{2}$$

b)

$$g(x, y) = \frac{x^2}{2} + 0.00001\frac{y^2}{2}$$

c)

$$h(x, y) = \frac{x^2}{2} + |y| .$$

Table of Contents

A closer look at least-squares regression

Least-squares regression

$\mathcal{Y} = \mathbb{R}$ and $\phi(x) \in \mathbb{R}^p$. Minimize the empirical square loss

$$\arg\min_{w \in \mathbb{R}^p} \frac{1}{2n} \sum_{i=1}^{n} \left(w^\top \phi(x_i) - y_i \right)^2$$

Useful notation

$$\mathbf{X} = \begin{pmatrix} \phi(x_1) & \dots & \phi(x_n) \end{pmatrix} \in \mathbb{R}^{p \times n} \quad \text{and} \quad \mathbf{Y} = \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix} \in \mathbb{R}^n$$

Least-squares can be written as (with euclidean norm in $\mathbb{R}^n$)

$$\arg\min_{w \in \mathbb{R}^p} \frac{1}{2n} \left\| \mathbf{X}^\top w - \mathbf{Y} \right\|^2$$

Overparameterization in linear regression

$$\arg\min_{w\in\mathbb{R}^p} \frac{1}{2n}\left\|\mathbf{X}^\top w - \mathbf{Y}\right\|^2$$

Consider the system of equations in $w \in \mathbb{R}^p$

$$\mathbf{X}^\top w = \mathbf{Y}\,, \qquad \mathbf{X}^\top \in \mathbb{R}^{n\times p}$$

Underparameterization. If $\mathrm{Rank}(\mathbf{X}^\top) < n$, then system can have no solutions (generically for $\mathbf{Y}$). Loss has unique minimizer, reached at the projection of $\mathbf{Y}$ on the linear span of the feature vectors.

Overparameterization. If $\mathrm{Rank}(\mathbf{X}^\top) = n$, then then system can have infinitely many solutions, and the minimal loss value is 0. The feature vectors have the capacity to fit perfectly the data in many ways.

GD in overparameterized least-squares

GD (with learning rate $n\eta$)

$$w_{t+1} = w_t - \eta \mathbf{X}(\mathbf{X}^\top w_t - \mathbf{Y}) = (I_p - \eta \mathbf{X}\mathbf{X}^\top) w_t + \eta \mathbf{X}\mathbf{Y}\,.$$

In overparameterized regime, denoting by λ_i the eigenvalues of $\mathbf{X}\mathbf{X}^\top$

$$I_p - \eta \mathbf{X}\mathbf{X}^\top \sim \begin{pmatrix} 1-\eta\lambda_1 & & & & \\ & \ddots & & \mathbf{0} & \\ & & 1-\eta\lambda_n & & \\ & & & 1 & \\ & \mathbf{0} & & & \ddots \\ & & & & 1 \end{pmatrix}$$

- GD converges exponentially fast to a minimizer of F, and the speed of convergence depends on eigenvalues of the feature matrix $\mathbf{X}\mathbf{X}^\top$.
- (w_t) only moves in an n-dimensional: the image of $\mathbf{X}$

(Essentially the same case as $f(x, y) = x^2/2$)

Table of Contents

Recall: Back to neural networks

Feedforward neural networks

For dimensions p, q, r, a **layer** is a function $\mathbb{R}^p \to \mathbb{R}^r$

$$\Phi_{\sigma,A,b} : x \mapsto \sigma\big(Ax + b\big)$$

where $\sigma : \mathbb{R}^q \to \mathbb{R}^r$ is a simple non-linear function, A is a $q \times p$ matrix and $b \in \mathbb{R}^q$ is a vector.

A **neural network** is a function of the form

$$h : x \mapsto \Phi_{\sigma_L,A_L,b_L} \circ \cdots \circ \Phi_{\sigma_0,A_0,b_0}(x)\,.$$

The trainable parameters are

$$w = (A_0, b_0, \ldots, A_L, b_L)\,.$$

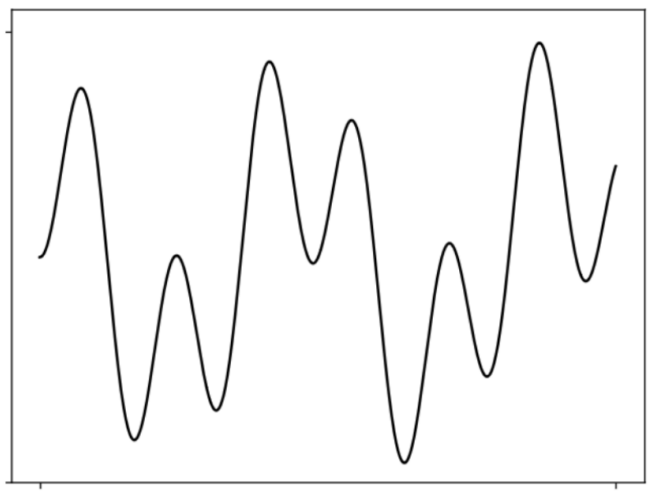
The (A_i) are the weights and the (b_i) the biases, but we often forget the biases and call w the weights.

Non-convexity

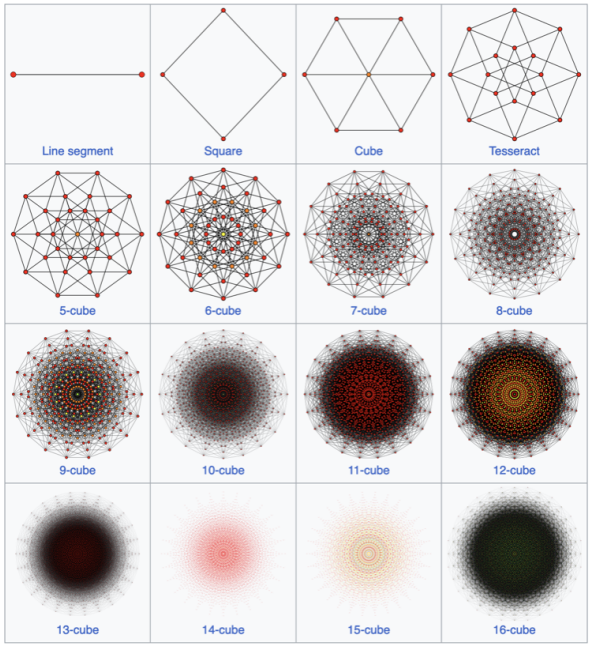
Even one hidden layer with square loss

$$\frac{1}{2}\big(h(x,w)-y\big)^2 = \frac{1}{2}\left(\sum_{k=1}^{m} c_k\,\sigma(a_k^\top x + b_k) - y\right)^2$$

Dependence on (c_k, a_k, b_k) looks complicated (and this is only one layer).



+



non-convex + high-dimension = trouble

Some empirical observations about neural nets

Neural net that are

- very large
- randomly initialized with an appropriate scaling (Le Cun initialization)

are such that

- training loss decreases very fast
- weights move very slowly
- the outputs of the net move in a similar fashion during training

This is called the **lazy regime** because the network learns by making small moves.

Table of Contents

Linearization

Lazy training is a very general phenomenon, that already happened in overparameterized linear models: the weights move to a close minimizer and the loss decreased fast.

What about non-linear models? Consider linearization:

$$h(x, w) = \underbrace{h(x, w_0) + Dh(x, w_0).(w - w_0)}_{\text{linear in } w} + o(\|w - w_0\|)$$

For convenience, omit the dependence on x.

Linearized model around w_0

$$h(w) = \underbrace{h(w_0) + Dh(w_0).(w - w_0)}_{:= \bar{h}(w)} + o(\|w - w_0\|)$$

(where the equality is between functions of x with an L_2-norm).

In a neighborhood of the the initial weights, the network behaves like a linear model with feature map $Dh(w_0)$

Lazy training: some heuristic computations

Lazy training

Lazy training occurs when the Taylor expansion of the model stays valid through the whole training procedure.

Hand-wavy, but important for intuition.

$$\underbrace{\frac{|F(w_1) - F(w_0)|}{F(w_0)}}_{\text{scale of loss variations}} \gg \underbrace{\frac{\|D^2 h(w_0)[w_1 - w_0, w_1 - w_0]\|}{\|Dh(w_0)[w_1 - w_0]\|}}_{\text{scale of variations of the linear approximations}}$$

i.e., using $w_1 - w_0 = \eta \nabla F(w_0)$ when

$$\frac{\|\nabla F(w_0)\|}{|F(w_0)|} \gg \frac{\|D^2 h(w_0)\|}{\|Dh(w_0)\|}.$$

Heuristic computations II: Square loss

Consider a square loss of the form

$$F(w) = \frac{\|h(w) - h^\star\|^2}{2} \left(:= \int \frac{(h(x,w) - y(x))^2}{2} \mathrm{d}x \right)$$

then $\|\nabla F(w_0)\| = \|Dh(w_0)^\top (h(w_0) - y)\| \approx \|Dh(w_0)\| \|h(w_0) - y\|$,
so

$$\frac{\|\nabla F(w_0)\|}{|F(w_0)|} \gg \frac{\|D^2 h(w_0)\|}{\|Dh(w_0)\|} \quad \text{iff} \quad \|h(w_0) - y\| \frac{\|D^2 h(w_0)\|}{\|Dh(w_0)\|^2} \ll 1 \,.$$

Definition (Lazy regime criterion for the square loss)

$$\kappa(w_0) := \|h(w_0) - y\| \frac{\|D^2 h(w_0)\|}{\|Dh(w_0)\|^2}$$

Morally, if $\kappa(w_0)$ is small, then the model stays close to its linear approximation during the whole training.

Lazy training

Let us describe two cases of lazy training:

- Scaling
- Large width + random initialization

Lazy training by scaling

Simple way to ensure lazy training: multiply the model outputs by a scale factor α, while maintaining $h(w_0) = 0$. Then

$$\kappa_{\alpha h(w_0)}(w_0) = \frac{1}{\alpha}\|y\| \frac{\|D^2 h(w_0)\|}{\|Dh(w_0)\|^2} \xrightarrow[\alpha \to \infty]{} 0$$

Consider the linearized model around w_0,

$$\bar{h}(w) = h(w_0) + Dh(w_0).(w - w_0)$$

Theorem (Chizat, Oyallon, Bach '19)

For the square loss, if $h(w_0) = 0$, the trajectories of gradient descent following αh and $\alpha \bar{h}$ stay within $O(1/\alpha^2)$ of each other.

Lazy training in ∞-width two layer nets

Proposition (Lazy training in infinite-width)

Consider a one-layer hidden layer neural network with m hidden nodes with Le Cun initialization, then

$$\kappa(W_0) \to 0 \quad as \quad m \to \infty\,.$$

Le Cun initialization: all weights are randomly initialized with variance of inputs of every node that sum to 1.

$$h(x, w_0) = \sum_{j=1}^{m} c_i \sigma(x^\top a_i)$$

$$a_j \sim \mathcal{N}(0,1) \qquad c_j \sim \mathcal{N}\left(0, \frac{1}{m}\right)$$

Equivalently:

$$h(x, w_0) = \alpha(m) \sum_{j=1}^{m} c_i \sigma(x^\top a_i)$$

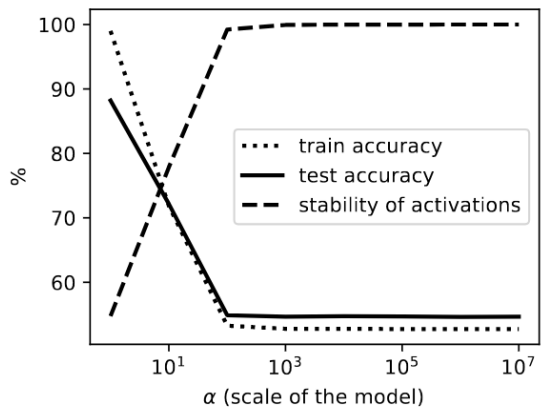
with $a_j, c_j \sim \mathcal{N}(0,1)$.

Lazy training: so what?

Benefits of the lazy training regime

- Neural nets are easy to train in the lazy training
- Great for theory: we can also identify the point of convergence and study the statistical properties of that point.

BUT limited for practice. Linearized nets behave worse than real life nets.



(a)

Model	Train acc.	Test acc.
ResNet wide, linearized	55.0	56.7
VGG-11 wide, linearized	61.0	61.7
Prior features [25]	-	82.3
Random features [28]	-	84.2
VGG-11 wide, standard	99.9	89.7
ResNet wide, standard	99.4	91.0

(b)

Figure 3: (a) Accuracies on CIFAR10 as a function of the scaling α. The stability of activations suggest a linearized regime when high. (b) Accuracies on CIFAR10 obtained for $\alpha = 1$ (standard, non-linear) and $\alpha = 10^7$ (linearized) compared to those reported for some linear methods without data augmentation: random features and prior features based on the scattering transform.

source the paper

Conclusion and Next time

Today

- Optimization in linear models
- The Lazy regime

Next time: A closer look at the linearized model for infinite-width networks **Neural Tangent Kernel**

Conclusion and Next time

Thanks !!
Break