COM S 578X: Optimization for Machine Learning

Lecture Note 9: Stochastic (Sub)Gradient Descent

Jia (Kevin) Liu

Assistant Professor
Department of Computer Science
Iowa State University, Ames, Iowa, USA

Fall 2019

Outline

In this lecture:

- Noisy unbiased subgradient
- Stochastic subgradient method
- Convergence results
- Online learning and adaptive signal processing
- Stochastic gradient descent for artificial neural networks

Noisy Unbiased Subgradient

- Random vector $\tilde{\mathbf{g}} \in \mathbb{R}^n$ is a noisy unbiased subgradient for a function $f : \mathbb{R}^n \to \mathbb{R}$ at $\mathbf{x}$ if for all $\mathbf{z}$

$$f(\mathbf{z}) \geq f(\mathbf{x}) + (\mathbb{E}\{\tilde{\mathbf{g}}\})^\top (\mathbf{z} - \mathbf{x})$$

  i.e., $\mathbb{E}\{\tilde{\mathbf{g}}\} \in \partial f(\mathbf{x})$

- Can be viewed as $\tilde{\mathbf{g}} = \mathbf{g} + \mathbf{n}$, where $\mathbf{g} \in \partial f$ and $\mathbb{E}\{\mathbf{n}\} = \mathbf{0}$

- $\mathbf{n}$ can be interpreted as error in computing $\mathbf{g}$, measurement noise, Monte Carlo sampling errors, etc.

- If $\mathbf{x}$ is also random, $\tilde{\mathbf{g}}$ is a noisy unbiased subgradient at $\mathbf{x}$ if

$$f(\mathbf{z}) \geq f(\mathbf{x}) + (\mathbb{E}\{\tilde{\mathbf{g}}|\mathbf{x}\})^\top (\mathbf{z} - \mathbf{x}), \quad \forall \mathbf{z}$$

  holds almost surely, i.e., $\mathbb{E}\{\tilde{\mathbf{g}}|\mathbf{x}\}) \in \partial f(\mathbf{x})$ w.p.1.

Stochastic Subgradient Method

- Consider $\min_{\mathbf{x} \in \mathbb{R}} f(\mathbf{x})$. Following standard GD or SGD, we should do:

$$\mathbf{x}_{k+1} = \mathbf{x}_k - s_k \mathbb{E}\{\mathbf{g}_k\}$$

- However, $\mathbb{E}\{\mathbf{g}_k\}$ is difficult to compute: Unknown distribution, too costly to sample at each iteration k, etc.

- Idea: Simply use a noisy unbiased subgradient to replace $\mathbb{E}\{\mathbf{g}_k\}$:

- The stochastic subgradient method works as follows:

$$\mathbf{x}_{k+1} = \mathbf{x}_k - s_k \tilde{\mathbf{g}}_k$$

  - $\mathbf{x}_k$ is the k-th iterate
  - $\tilde{\mathbf{g}}_k$ is any noisy subgradient of at $\mathbf{x}_k$, i.e., $\mathbb{E}\{\tilde{\mathbf{g}}_k | \mathbf{x}_k\} = \mathbf{g}_k \in \partial f(\mathbf{x}_k)$
  - s_k is the step size
  - Let $f_{\text{best}}^{(k)} = \min\{f(\mathbf{x}_1), \ldots, f(\mathbf{x}_k)\}$

Historical Perspective

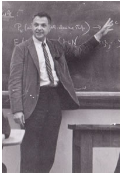
- Also referred to as stochastic approximation in the literature, first introduced by [Robbins, Monro '51] and [Keifer, Wolfowitz '52]

- The original work [Robbins, Monro '51] is motivated by finding a root of a continuous function:

$$f(\mathbf{x}) = \mathbb{E}\{F(\mathbf{x}, \theta)\} = 0,$$

where $F(\cdot, \cdot)$ is unknown and depends on a random variable θ. But the experimenter can take random samples (noisy measurements) of $F(\mathbf{x}, \theta)$

* CLT for dep. r.v. with Hoeffding.
* Lai-Robbins stochastic multi-armed bandits. (MAB). log(T)



UNC. Columbia Rutgers.

Herbert Robbins

BS MIT UNC. Lehigh.

Sutton Monro

Historical Perspective

- Robbins-Monro: $\mathbf{x}_{k+1} = \mathbf{x}_k + s_k Y(\mathbf{x}_k, \theta)$, where:
  - $\mathbb{E}\{Y(\mathbf{x}, \theta) | \mathbf{x} = \mathbf{x}_k\} = f(\mathbf{x}_k)$ is an unbiased estimator of $f(\mathbf{x}_k)$
  - Robbins-Monro originally showed convergence in L^2 and in probability
  - Blum later prove convergence is actually w.p.1. (almost surely)
  - Key idea: Diminishing step-size provides implicit averaging of the observations
- Robbins-Monro's scheme can also be used in stochastic optimization of the form $f(\mathbf{x}^*) = \min_{\mathbf{x}} \mathbb{E}\{F(\mathbf{x}, \theta)\}$ (equivalent to solving $\nabla f(\mathbf{x}^*) = 0$)
- Stochastic approximation (or more generally, stochastic (sub) gradient) has found applications in many areas
  - Adaptive signal processing
  - Dynamic network control and optimization
  - Statistical machine learning
  - Workhorse algorithm for deep neural networks (will see an example)

Assumptions and Step Size Rules

- $f^* = \inf_x f(\mathbf{x}_k) > -\infty$, with $f(\mathbf{x}^*) = f^*$
- $\mathbb{E}\{\|\tilde{\mathbf{g}}_k\|_2^2\} \leq G^2$, for all k
- $\mathbb{E}\{\|\mathbf{x}_0 - \mathbf{x}^*\|_2^2\} \leq R^2$

Commonly used step-size strategies:

- Constant step-size: $s_k = s$, $\forall k$
- Step-size is square summable, but not summable

$$s_k > 0,\ \forall k, \qquad \sum_{k=1}^{\infty} s_k^2 < \infty, \qquad \sum_{k=1}^{\infty} s_k = \infty$$

Note: This is stronger than needed, but just to simplify proof

Convergence Results

- Convergence in expectation:

$$\lim_{k\to\infty} \mathbb{E}\{f_{\text{best}}^{(k)}\} = f^*$$

- Convergence in probability: for any $\epsilon > 0$,

$$\lim_{k\to\infty} \Pr\{|f_{\text{best}}^{(k)} - f^*| > \epsilon\} = 0$$

- Almost sure convergence

$$\Pr\{\lim_{k\to\infty} f_{\text{best}}^{(k)} = f^*\} = 1$$

- See [Kushner, Yin '97] for a complete treatment on convergence analysis

Convergence in Expectation and Probability

Proof Sketch:

- Key quantity: Expected squared Euclidean distance to the optimal set. Let $\mathbf{x}^*$ be any minimzer of f. We can show that

$$\mathbb{E}\{\|\mathbf{x}_{k+1} - \mathbf{x}^*\|_2^2 | \mathbf{x}_k\} \leq \|\mathbf{x}_k - \mathbf{x}^*\|_2^2 - 2s_k(f(\mathbf{x}_k) - f^*) + s_k^2 \mathbb{E}\{\|\tilde{\mathbf{g}}_k\|_2^2 | \mathbf{x}_k\}$$

- which can further lead to

$$\min_{i=1,\ldots,k} \left\{ \mathbb{E}\{f(\mathbf{x}_i)\} - f^* \right\} \leq \frac{R^2 + G^2 \|s\|^2}{2\sum_{i=1}^k s_i}$$

- The result $\min_{i=1,\ldots,k} \mathbb{E}\{f(\mathbf{x}_i)\} \to f^*$ simply follows from the divergent step-size series rule

Convergence in Expectation and Probability

- Jensen's inequality and concavity of minimum yields

$$\mathbb{E}\{f_{\text{best}}^{(k)}\} = \mathbb{E}\{\min_{i=1,\ldots,k} f(\mathbf{x}_i)\} \leq \min_{i=1,\ldots,k} \mathbb{E}\{f(\mathbf{x}_i)\}$$

  Therefore, $\mathbb{E}\{f_{\text{best}}^{(k)}\} \to f^*$ (convergence in expectation)

- Convergence in expectation also implies convergence in probability: By Markov's inequality, for any $\epsilon > 0$,

$$\Pr\{f_{\text{best}}^{(k)} - f^* \geq \epsilon\} \leq \frac{\mathbb{E}\{f_{\text{best}}^{(k)} - f^*\}}{\epsilon},$$

  i.e., RHS goes to 0, which proves convergence in probability. $\square$

Example: Piecewise Linear Minimization

$$\text{Minimize} \quad f(\mathbf{x}) = \min_{i=1,\ldots,m} \left\{ \mathbf{a}_i^\top \mathbf{x} + b_i \right\}$$

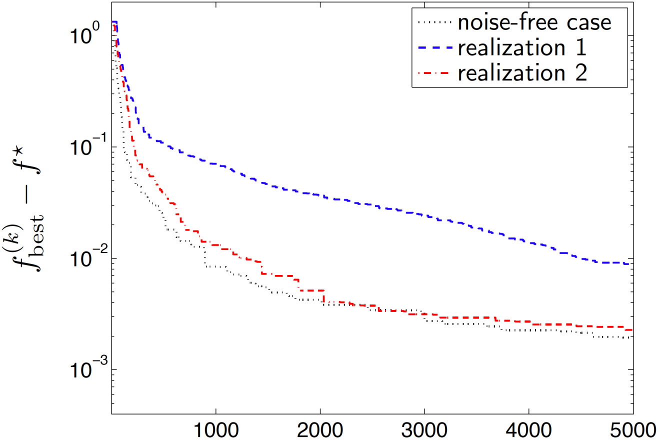
Using stochastic subgradient algorithm with noisy subgradient

$$\tilde{\mathbf{g}}_k = \mathbf{g}_k + \mathbf{n}_k, \qquad \mathbf{g}_k \in \partial f(\mathbf{x}_k),$$

where $\mathbf{n}_k$ is independent zero mean random variables

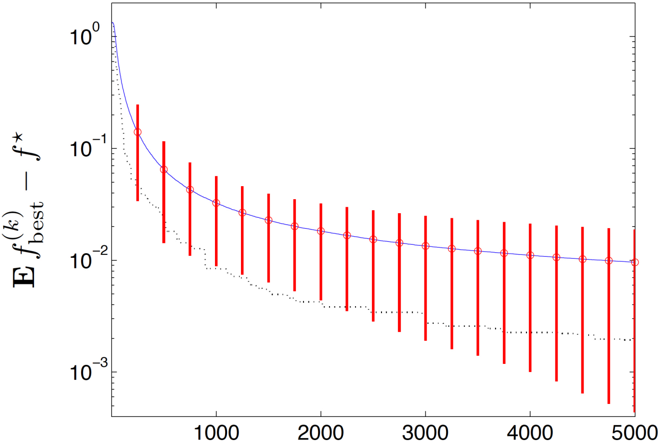
Example: Piecewise Linear Minimization

Problem instance: $n = 20$ variables, $m = 100$ terms, $f^* \approx 1.1$, $s_k = 1/k$, $\mathbf{n}_k$ are i.i.d. $\mathcal{N}(0, 0.05\mathbf{I})$ (25% noise since $\|\mathbf{g}\| \approx 4.5$)



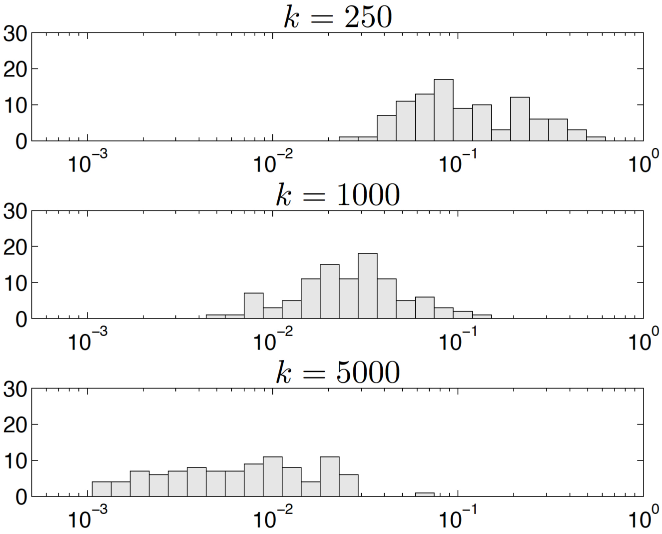
Example: Piecewise Linear Minimization

Average and standard deviation for $f_{\rm best}^{(k)} - f^*$ over 100 realizations



Example: Piecewise Linear Minimization

Empirical distributions of $f_{\text{best}}^{(k)} - f^*$ at $k = 250$, $k = 1000$, and $k = 5000$



Example: Online Sequential Learning

- $(\mathbf{x}, y) \in \mathbb{R}^n \times \mathbb{R}$ have some joint distribution
- Find weight vector $\mathbf{w} \in \mathbb{R}^n$ such that $\mathbf{w}^\top \mathbf{x}$ is a good estimator of y
- Choose $\mathbf{w}$ to minimize expected value of a convex loss function L

$$J(\mathbf{w}) = \mathbb{E}\{L(\mathbf{w}^\top \mathbf{x} - y)\}$$

  - $L(u) = u^2$: mean-square error
  - $L(u) = |u|$: mean-absolute error
- At each step (i.e., each iteration), we are given a sample $(\mathbf{x}_k, y_k)$ drawn from the distribution

Example: Online Sequential Learning

- Noisy unbiased subgradient of J at $\mathbf{w}_k$, based on sample $(\mathbf{x}_{k+1}, y_{k+1})$:

$$\tilde{\mathbf{g}}_k = L'(\mathbf{w}_k^\top \mathbf{x}_{k+1} - y_{k+1})\mathbf{x}_{k+1},$$

  where L' denotes the derivative or a subgradient of L

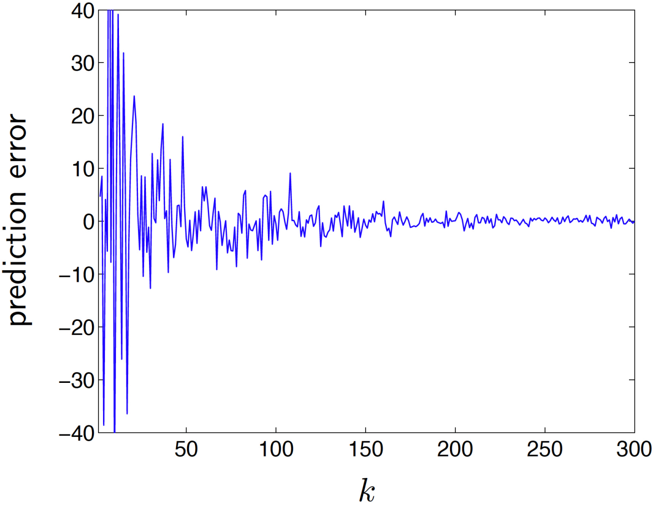
- Online algorithm:

$$\mathbf{w}_{k+1} = \mathbf{w}_k - s_k L'(\mathbf{w}_k^\top \mathbf{x}_{k+1} - y_{k+1})\mathbf{x}_{k+1}$$

  - For $L(u) = u^2$, gives the LMS (least mean-square) algorithm
  - For $L(u) = |u|$, gives the sign algorithm
  - $\mathbf{w}_k^\top \mathbf{x}_{k+1} - y_{k+1}$ is referred to as the predictor error

Example: Mean Square Error Minimization

Problem instance: $n = 10$, $(\mathbf{x}, y) \sim \mathcal{N}(0, \mathbf{\Sigma})$, and $s_k = 1/k$



One More Example

Artificial Neural Networks ...

Convergence of R.V.

1. Convergence in distr. (weak convergence):

A seq. of (real-valued) r.v. $\{X_n\}$ converges in distr. to X if $\lim_{n\to\infty} F_n(X_n) = F(X)$, where F_n and F are cdf of X_n and X, resp. Denote as: $X_n \xrightarrow{D} X$.

2. Convergence in prob. to a r.v.:

$\{X_n\}$ converges in prob. to a r.v. X if $\forall \varepsilon > 0$, $\lim_{n\to\infty} \Pr\{|X_n - X| > \varepsilon\} = 0$. Denote as: $X_n \xrightarrow{P} X$.

3. Almost sure convergence (pt.-wise convergence in real analysis):

$\{X_n\}$ converges a.s. (a.e., or w.p.1, or strongly) to X if $\Pr\{\lim_{n\to\infty} X_n = X\} = 1$, Denote as: $X_n \xrightarrow{a.s.} X$

4. Convergence in expectation: Given $r \geq 1$, $\{X_n\}$ converges in r-th mean to r.v. X if r-th absolute moments $\mathbb{E}\{|X_n|^r\}$ and $\mathbb{E}\{|X|^r\}$ exist and

$$\lim_{n\to\infty} \mathbb{E}\{|X_n - X|^r\} = 0.$$

Denote as: $X_n \xrightarrow{L^r} X$.

* $r=1$: X_n converges in mean to X.
* $r=2$: —— — — . mean square to X.

strongest

$$\xrightarrow{a.s.} \;\Rightarrow\; \xrightarrow{P} \;\Rightarrow\; \xrightarrow{D}$$

$$\Uparrow$$

Markov ineq.

$$\xrightarrow{L^s} \;\underset{s>r\geq 1}{\Rightarrow}\; \xrightarrow{L^r}$$

Thm: If $\mathbb{E}\{\|\tilde{g}_k\|_2\} \le G$, $\forall k$, $\mathbb{E}\{\|\underline{x}_1 - \underline{x}^*\|_2\} \le R$, and step-sizes $\{s_k\}_{k=1}^{\infty}$ satisfy: $s_k > 0$, $\forall k$, $\sum_{k=1}^{\infty} s_k^2 = B < \infty$, $\sum_{k=1}^{\infty} s_k = \infty$, then we have:

$$\lim_{k\to\infty} \mathbb{E}\{f_{best}^{(k)}\} = f^* \quad \text{and} \quad \lim_{k\to\infty} \Pr\{|f_{best}^{(k)} - f^*| > \varepsilon\} = 0, \ \forall \varepsilon > 0.$$

Proof. Consider the conditional expected square Eclidean dist:

$$\mathbb{E}\{\|\underline{x}_{k+1} - \underline{x}^*\|_2^2 \mid \underline{x}_k\} = \mathbb{E}\{\|\underline{x}_k - s_k \tilde{g}_k - \underline{x}^*\|_2^2 \mid \underline{x}_k\}$$

$$= \mathbb{E}\{\|\underline{x}_k - \underline{x}^*\|_2^2 + s_k^2 \|g_k\|_2^2 - 2 s_k \tilde{g}_k^T (\underline{x}_k - \underline{x}^*) \mid \underline{x}_k\}.$$

$$= \|\underline{x}_k - \underline{x}^*\|_2^2 + s_k^2 \mathbb{E}\{\|g_k\|_2^2 \mid \underline{x}_k\} - 2 s_k \mathbb{E}\{\tilde{g}_k \mid \underline{x}_k\}^T (\underline{x}_k - \underline{x}^*)$$

$$\le \|\underline{x}_k - \underline{x}^*\|_2^2 + s_k^2 \mathbb{E}\{\|g_k\|_2^2 \mid \underline{x}_k\} \underbrace{- 2 s_k (f(\underline{x}_k) - f^*)}_{(*)}$$

where $(*)$ follows from $\mathbb{E}\{\tilde{g}_k \mid \underline{x}_k\} = g_k \in \partial f(\underline{x}_k)$

Hence, $f(\underline{x}^*) \ge f(\underline{x}_k) + \mathbb{E}\{\tilde{g}_k \mid \underline{x}_k\}^T (\underline{x}^* - \underline{x}_k)$

$$\Rightarrow -\mathbb{E}\{\tilde{g}_k \mid \underline{x}_k\}^T (\underline{x}_k - \underline{x}^*) \le -(f(\underline{x}_k) - f^*)$$

Note : $\underline{x}_{k+1}$ only depends on $\underline{x}_k$ and cond. indep. of $\underline{x}_{k-1}, \cdots, \underline{x}_1$.

$$\mathbb{E}\left\{\|\underline{x}_{k+1}-\underline{x}^*\|_2^2 \,\middle|\, x_k\right\} = \mathbb{E}\left\{\|\underline{x}_{k+1}-\underline{x}^*\|_2^2 \,\middle|\, x_k, \cdots, x_1\right\}$$

Take expectation over joint distr. of $\{x_k, \cdots, x_1\}$ yields:

$$\mathbb{E}\left\{\|\underline{x}_{k+1}-\underline{x}^*\|_2^2\right\} \le \mathbb{E}\left\{\|\underline{x}_k-\underline{x}^*\|_2^2\right\} - 2s_k\left[\mathbb{E}\{f(\underline{x}_k) - f^*\}\right] + s_k^2\, \mathbb{E}\left\{\|\tilde{g}_k\|_2^2\right\}.$$

Apply this process recursively, noting $\mathbb{E}\{\|\tilde{g}_k\|_2^2\} \le G^2$:

$$\mathbb{E}\left\{\|\underline{x}_{k+1}-\underline{x}^*\|_2^2\right\} \le \mathbb{E}\left\{\|\underline{x}_1-\underline{x}^*\|_2^2\right\} - 2\sum_{i=1}^{k} s_i \underbrace{\left(\mathbb{E}\{f(\underline{x}_i)\} - f^*\right)}_{\ge \min_{i=1,\cdots,k} \mathbb{E}\{f(\underline{x}_i)\}.} + G^2 \sum_{k=1}^{\infty} s_k^2$$

$$\Rightarrow \min_{i=1,\cdots,k} \left\{\mathbb{E}\{f(\underline{x}_k)\} - f^*\right\} \le \frac{R^2 + G^2 B}{2\underbrace{\sum_{i=1}^{k} s_i}_{\infty}} \to 0 \text{ as } k \to \infty.$$

So, we can conclude that $\min_{i=1,\cdots,k} \mathbb{E}\{f(\underline{x}_i)\} \to f^*$.

in HW2.

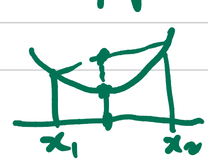
Claim: The fn $g(y) \triangleq \min_{i=1,\cdots,k} \{y_i\}$ is concave, $\forall y \in \mathbb{R}^k$

Therefore, by Jensen's ineq.

Jensen's ineq: If f is convex, for $\mu \in [0,1]$: $f(\mu \underline{x}_1 + (1-\mu)\underline{x}_2) \le \mu f(\underline{x}_1) + (1-\mu) f(\underline{x}_2)$

In prob:

* If f convex: $f(\mathbb{E}X) \le \mathbb{E}f(X)$
* —— concave $\ge$



$$\mathbb{E} f_{best}^{(k)} = \mathbb{E}\Big\{ \underbrace{\min_{i=1,\cdots,k} f(x_i)}_{\text{concave}} \Big\} \underset{\uparrow \text{ Jensen}}{\leq} \min_{i=1,\cdots,k} \mathbb{E}\{f(x_i)\} \longrightarrow f^*,$$

i.e., convergence in expectation.

Use Makov's Ineq. (If X is non-neg. r.v.
$$\Pr\{X \geq \varepsilon\} \leq \frac{\mathbb{E}X}{\varepsilon} \Big).$$

$$\Pr\{ f_{best}^{(k)} - f^* \geq \varepsilon \} \leq \frac{\overbrace{\mathbb{E}\{f_{best}^{(k)} - f^*\}}^{\to 0 \text{ as } k \to \infty.}}{\varepsilon} \longrightarrow 0,$$

so, we get convergence in prob. ∎

Neural Networks

- History:

* Rosenblatt : 1958 "Mark-1 perceptron" (linear classifier).

* Widrow-Hoff : 1960's "Adaline/Madaline" (Multi-layer perceptron)
Nonlinear activation, however backprop.

* Rumelhart - Hinton - Williams : 1986 BP (Nature). 4-page.

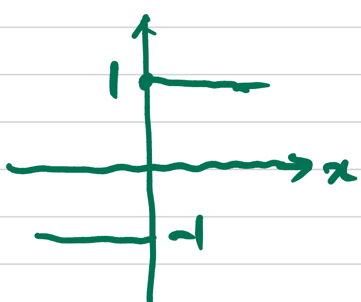
* Yan LeCun 1989 : CNN.

* Hinton - Salakhutdinov 2006 : Deep learning,
restricted Boltzman machine

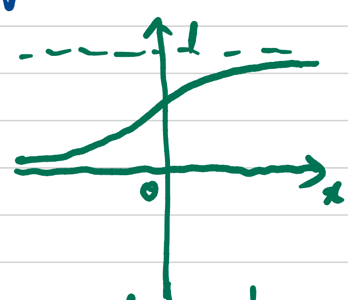
* Krizevsky, Sutskever, Hinton 2012 : (~25%).
"AlexNet": Deep CNN. ImageNet 12%   (5%) 2015, ResNet.
1st GPU-based CNN.

- Neuron : A nonlinear fn : $\sigma : \mathbb{R} \to \mathbb{R}$ $x \to \sigma \to \sigma(x)$

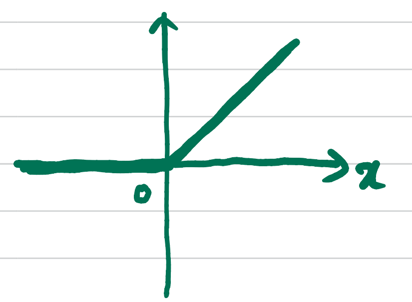
Ex:



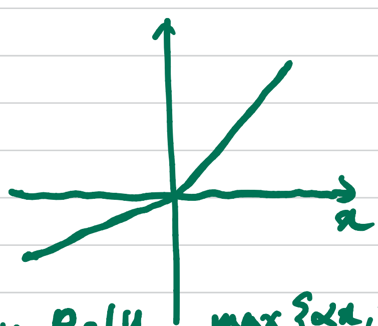
sgn(x)



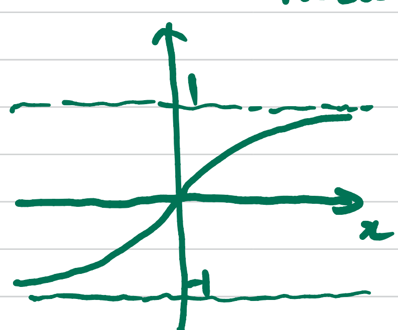
sigmoid : $\frac{1}{1+e^{-x}}$



ReLU : $\max(0, x)$

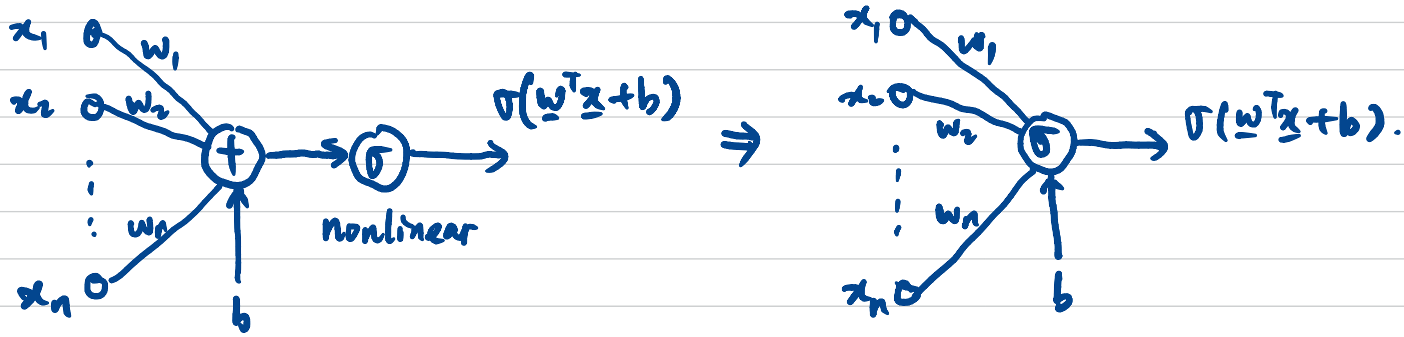


Leaky ReLU : $\max\{\alpha x, x\}$
$\alpha \in [0,1)$.



$\tanh(x) = -i \tan(ix)$
"hyperbolic tangent".

* Neuron structure :

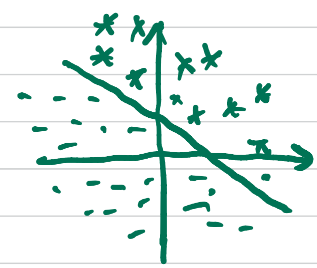


1°. Summation always assumed.

2°. Bias b is important: e.g., Let σ be sign$(\cdot)$. then:

$$\begin{cases} +1, & \text{if } \underline{w}^T\underline{x} \geq -b \\ -1, & \text{------} < -b \end{cases}$$

3°. $\underline{w}^T\underline{x}+b$: is hyperplane. A single neuron divides input space in 2 parts.



Universal Approx. Theorem (UAT).

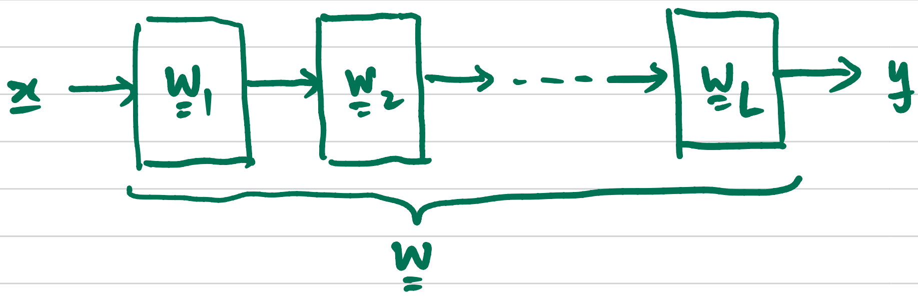
I_n : n-dim unit cube $[0,1]^n$ $\quad f(\underline{x}) \in C(I_n)$ to be approximated.

$C(I_n)$: Space of cont. fns on I_n.

Thm (Cybenko '89): Let σ be any cont. nonlinear fn, the finite sum of the form: $G(\underline{x}) = \sum_{i=1}^{N} \alpha_i \sigma(\underline{w}_i^T\underline{x}+b_i)$ is dense in $C(I_n)$, i.e., given any $\varepsilon>0$, there must $\exists$ a $G(\underline{x})$ of the above form, s.t. $|G(\underline{x})-f(\underline{x})| < \varepsilon$, $\forall \underline{x} \in I_n$

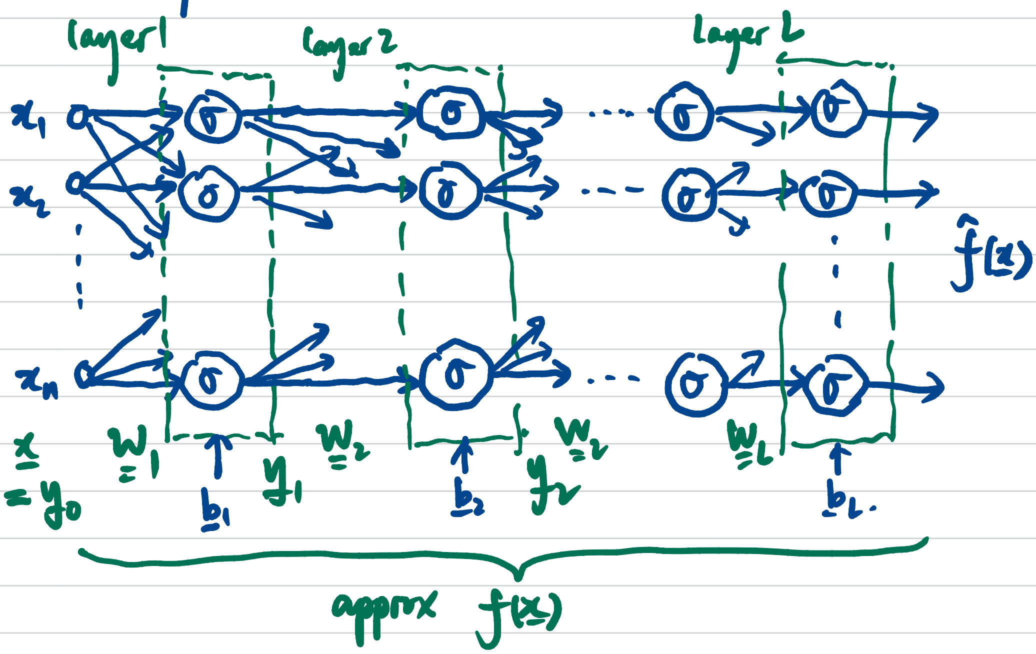
* Not specific choice of activation fn, but rather the architecture of NN that gives the potential of being a universal learning machine.

* σ : cont. nonlin. otherwise, no richness.



* Caveat: UAT is only a existence result, doesn't say how many neurons are needed, also doesn't say how to construct $G(\underline{x})$.

- Multi-layer NN: Allow dividing high-dim space in more complicated ways.



- # of neurons per layer could be different.

- Goal: To choose weights so that NN's output $\hat{f}(\underline{x})$ is "close" to $f(\underline{x})$ for some unknown f

i.e., $\hat{f}(\underline{x}) \approx f(\underline{x})$, $\forall \underline{x}$.

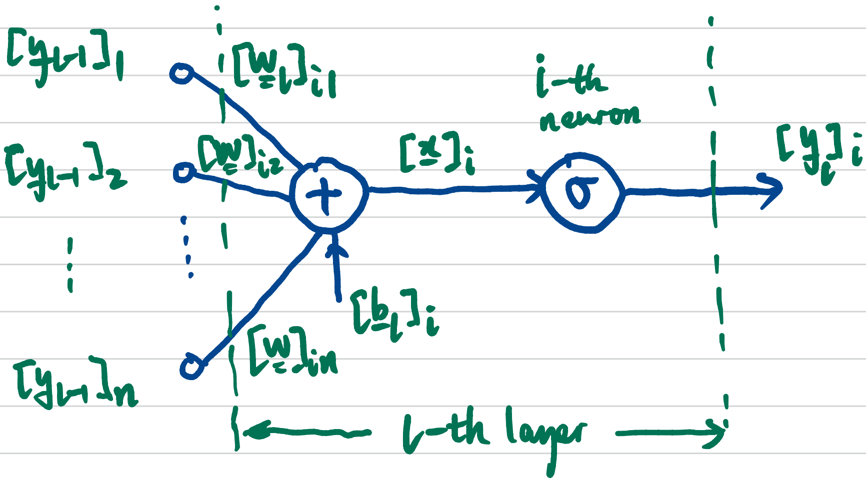
- structure of $\hat{f}$: Let y_i be vector output after layer i.

$\underline{y}_1 = \sigma(\underline{x}_1) = \sigma(\underline{\underline{W}}_1 \underset{\leftarrow \underline{x}}{\underline{y}_0} + \underline{b}_1)$ // $\sigma(\cdot)$ element-wise

Similarly, $\underline{y}_2 = \sigma(\underline{x}_2) = \sigma(\underline{\underline{W}}_2 \underline{y}_1 + \underline{b}_2)$

$\vdots$

$\underline{y}_L = \sigma(\underline{x}_L) = \sigma(\underline{\underline{W}}_L \underline{y}_{L-1} + \underline{b}_L)$



* $[\underline{\underline{W}}_l]_{ij}$: weight from input $[\underline{y}_{l-1}]_j$ to neuron i in l-th layer.

* If f is scalar-valued fn : last layer has single neuron.

* Let's define $\underline{y}_0 = \underline{x}$, then $\underline{y}_k = \sigma(\underline{x}_k)$, $\underline{x}_k = \underline{\underline{W}}_k \underline{y}_{k-1} + \underline{b}_k$

* <u>Goal of training:</u> Find "good" weight matrices $\underline{\underline{W}}_1, \cdots, \underline{\underline{W}}_L$ and bias vectors $\underline{b}_1 \cdots \underline{b}_L$ through "training" and sample data $(\underline{x}^{(1)}, f(\underline{x}^{(1)})), \cdots\cdots (\underline{x}^{(N)}, f(\underline{x}^{(N)}))$ to minimize some empirical loss fn. empirical risk minimization (ERM).

$$J = \frac{1}{N}\sum_{n=1}^{N} \tilde{L}(\underbrace{f(\underline{x}^{(n)})}_{\text{ground truth}}, \underbrace{y_L^{(n)}}_{\text{model output.}})$$

* $\tilde{L}$: often convex. For example:

square loss: $J = \frac{1}{N}\sum_{n=1}^{N} \frac{1}{2}\|f(x) - y_L^{(n)}\|^2$

logistic regression: $J = \frac{1}{N}\sum_{n=1}^{N} \log \Pr(y_i | x_i, \theta)$,

$$\Pr(\underline{Y}=1|\underline{X};\theta) = \frac{1}{1+e^{-\theta^T \underline{X}}} = h_{\underline{\theta}}(\underline{X})$$

$$\Pr(\underline{Y}=0|\underline{X};\theta) = 1 - h_{\underline{\theta}}(\underline{X}).$$

- <u>Traing</u>: Optimization to solve ERM.

* GD: $\underline{\underline{W}}_l[t+1] = \underline{\underline{W}}_l[t] - s_t \nabla_{\underline{\underline{W}}_l} J[t]$,

$$\underline{b}_l[t+1] = \underline{b}_l[t] - s_t \nabla_{\underline{b}_l} J[t].$$

where $\nabla_{\underline{\underline{W}}_l} J[t]$ is matrix of partial der. w.r.t. weights.

the entry at i-th row j-th col being $\frac{1}{N}\sum_{n=1}^{N} \frac{\partial \tilde{L}^{(n)}[t]}{\partial [\underline{\underline{W}}]_{ij}}$

$\nabla_{\underline{b}_l} J[t]$ is vector of par. der. w.r.t. biases.

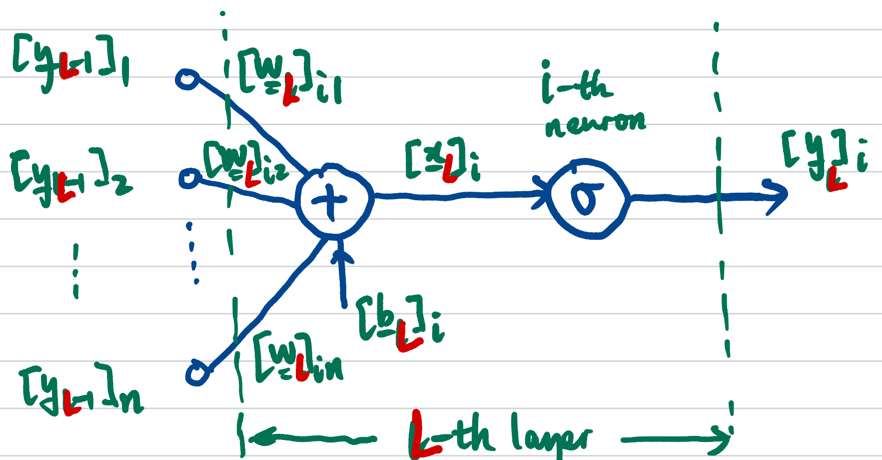
the i-th component: $\frac{1}{N}\sum_{n=1}^{N} \frac{\partial \tilde{L}^{(n)}[t]}{\partial [\underline{b}_l]_i}$

in the form of summation.

To calculate $\frac{\partial \tilde{L}^{(n)}[t]}{\partial [\underline{W}]_{ij}}$ and $\frac{\partial \tilde{L}^{(n)}[t]}{\partial [\underline{b}_l]_i}$: Drop "(n)" and "[t]".

* At layer L (last output layer)

$$\frac{\partial \tilde{L}}{\partial [\underline{W}_L]_{ij}} = \frac{\partial \tilde{L}}{\partial [\underline{y}_L]_i} \frac{\partial [\underline{y}_L]_i}{\partial [\underline{W}_L]_{ij}}$$



$$= \underbrace{\frac{\partial \tilde{L}}{\partial [\underline{y}_L]_i}}_{\substack{\text{depends}\\ \text{on } \tilde{L}\\ (1)}} \cdot \underbrace{\frac{\partial [\underline{y}_L]_i}{\partial [\underline{x}_L]_i}}_{\substack{\text{local}\\ \text{grad of}\\ \sigma(\cdot).\ (2)}} \cdot \underbrace{\frac{\partial [\underline{x}_L]_i}{\partial [\underline{W}_L]_{ij}}}_{\substack{\text{local}\\ \text{grad of}\\ \text{weights } (3).}}$$

From computational graph :

$$\frac{\partial [\underline{x}_L]_i}{\partial [\underline{W}_L]_{ij}} = \frac{\partial \left(\sum_{j=1}^{N} [\underline{W}_L]_{ij} [\underline{y}_{L-1}]_j + [\underline{b}]_i \right)}{\partial [\underline{W}_{ij}]_{ij}} = [\underline{y}_{L-1}]_j$$

$$\frac{\partial [\underline{y}_L]_i}{\partial [\underline{x}_L]_i} = \frac{\partial [\sigma([\underline{x}_L]_i)]}{\partial [\underline{x}_L]_i} = \sigma'([\underline{x}_L]_i)$$

} computable using local info at L-th layer.

For $\frac{\partial \tilde{L}}{\partial [\underline{y}_L]_i}$, consider, e.g., square loss. Then

$$\frac{\partial \tilde{L}}{\partial [\underline{y}_L]_i} = \frac{\partial \left(\frac{1}{2} \| f(\underline{x}) - \underline{y}_L \|^2 \right)}{\partial [\underline{y}_L]_i} = [\underline{y}_L - f(\underline{x})]_i$$

Just the prediction error if square loss is used.

Thus, $\frac{\partial \tilde{L}}{\partial [\underline{W}_L]_{ij}}$ can be computed using (1)–(3).

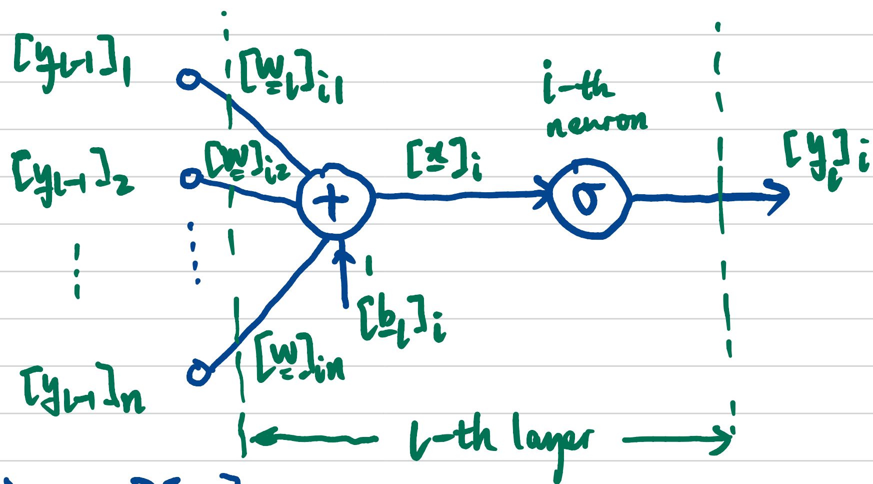
Similarly, for bias, we have:

$$\frac{\partial \tilde{L}}{\partial [\underline{b}_L]_i} = \underbrace{\frac{\partial \tilde{L}}{\partial [\underline{y}_L]_i}}_{\text{same as (1)}} \cdot \underbrace{\frac{\partial [\underline{y}_L]_i}{\partial [\underline{x}_L]_i}}_{\text{same as (2)}} \cdot \underbrace{\frac{\partial [\underline{x}_L]_i}{\partial [\underline{b}_L]_i}}_{=1 \text{ (from the computational graph)}}.$$

Hence, $\frac{\partial \tilde{L}}{\partial [\underline{\underline{W}}_L]_{ij}}$ and $\frac{\partial \tilde{L}}{\partial [\underline{b}_L]_i}$ can all be calculated.

* At layer $1 \le l < L$.

Following the decomposition approach by chain rule:



$$\frac{\partial \tilde{L}}{\partial [\underline{\underline{W}}_l]_{ij}} = \underbrace{\frac{\partial \tilde{L}}{\partial [\underline{y}_l]_i}}_{\substack{\text{non-local grad.} \\ (4)}} \cdot \underbrace{\underbrace{\frac{\partial [\underline{y}_l]_i}{\partial [\underline{x}_l]_i}}_{\substack{\text{local grad} \\ =\sigma'([\underline{x}_l]_i)}} \cdot \underbrace{\frac{\partial [\underline{x}_l]_i}{\partial [\underline{\underline{W}}_l]_{ij}}}_{\substack{\text{local grad} \\ =[\underline{y}_{l-1}]_i}}}_{\text{same derivation as in layer } L.}$$

Consider $\frac{\partial \tilde{L}}{\partial [y_l]_i}$ in (4): Again, by chain rule:

$$\frac{\partial \tilde{L}}{\partial [y_l]_i} = \sum_{k=1}^{|l+1|} \frac{\partial \tilde{L}}{\partial [y_{l+1}]_k} \cdot \frac{\partial [y_{l+1}]_k}{\partial [y_l]_i}$$

$$= \sum_{k=1}^{|l+1|} \underbrace{\frac{\partial \tilde{L}}{\partial [y_{l+1}]_k}}_{\substack{\text{computed previously,}\\ \text{so available after finishing}\\ \text{layer } l+1}} \cdot \underbrace{\underbrace{\frac{\partial [y_{l+1}]_k}{\partial [\underline{x}_{l+1}]_k}}_{\substack{\text{local grad at layer}\\ l+1:\\ =\sigma'([\underline{x}_{l+1}]_k)}} \cdot \underbrace{\frac{\partial [\underline{x}_{l+1}]_k}{\partial [y_l]_i}}_{\substack{\text{local grad at layer}\\ l+1:\\ =[\underline{W}_{l+1}]_{ki}}}}_{\text{available after processing layer } l+1.}$$

Similarly, for bias, we have:

$$\frac{\partial \tilde{L}}{\partial [\underline{b}_l]_i} = \underbrace{\frac{\partial \tilde{L}}{\partial [y_l]_i}}_{\substack{\text{same as}\\ (4)}} \cdot \underbrace{\frac{\partial [y_l]_i}{\partial [\underline{x}_l]_i}}_{\substack{\text{local grad}\\ =\sigma'([\underline{x}_l]_i)}} \cdot \underbrace{\frac{\partial [\underline{x}_l]_i}{\partial [\underline{b}_l]_i}}_{=1 \text{ (from comp. graph).}}$$

Finally, combining all discussions, we have the "Backprop" algorithm as follows:

Backpropogation: (recursively using chain rule).

(1) Compute $\dfrac{\partial \tilde{L}^{(n)}}{\partial [y_L]_i}$, $\forall i = 1, \cdots, |L|$, $n = 1, \cdots, m$ ← Just the avg training error if square loss is used.

(2) for ($l = L$ down to 1) {

$$\frac{\partial \tilde{L}^{(n)}}{\partial [\underline{W}_l]_{ij}} = \frac{\partial \tilde{L}^{(n)}}{\partial [y_l]_i} \cdot \sigma'([x_l]_i) \cdot [y_{l-1}]_j, \quad \forall i = 1, \cdots, |L|, n = 1, \cdots, m \quad (1)$$

$$\frac{\partial \tilde{L}^{(n)}}{\partial [b_l]_i} = \frac{\partial \tilde{L}^{(n)}}{\partial [y_l]_i} \cdot \sigma'([x_l]_i), \quad \forall \cdots 1, \cdots m \quad (2).$$

"local grad"!

Compute average of (1), (2).

$l = l-1$

$$\frac{\partial \tilde{L}^{(n)}}{\partial [y_{l-1}]_i} = \sum_{k=1}^{|L|} \underbrace{\frac{\partial \tilde{L}^{(n)}}{\partial [y_l]_k}}_{\text{computed previously}} \cdot \sigma'([x_l]_k) \cdot [\underline{W}]_{ki}, \quad \forall, \cdots = 1, \cdots, m$$

"upstream grad".

This is the backprop part

}

Remarks:

1. To compute full grad for GD, we need to compute

$$\frac{1}{N} \sum_{n=1}^{N} \frac{\partial \tilde{L}^{(n)}[t]}{\partial [\underline{W}]_{ij}} \quad \text{and} \quad \frac{1}{N} \sum_{n=1}^{N} \frac{\partial \tilde{L}^{(n)}[t]}{\partial [b_l]_i},$$

N is large typically.

More practical: Use a mini-batch of size m (usually $m \ll N$) to compute a stochastic grad:

$$\frac{1}{m}\sum_{n=1}^{m} \frac{\partial \tilde{L}^{(n)}[t]}{\partial [\underline{\underline{W}}_l]_{ij}} \quad \text{and} \quad \frac{1}{m}\sum_{n=1}^{m} \frac{\partial \tilde{L}^{(n)}(t)}{\partial [\underline{b}_l]_i}$$

GD → SGD.

2°. Even though loss fn is convex, the training of NN is NOT a convex opt. prob! The obj fn is:

$$F(\underline{x}) = \underbrace{L\left[\sigma_L\left(\underline{\underline{W}}_L\left(\sigma_{L-1}\left(\underline{\underline{W}}_{L-1}\cdots\cdots\sigma_1\left(\underline{\underline{W}}_1\underline{x}+\underline{b}_1\right)+\cdots+\underline{b}_L\right)-f(\underline{x})\right]\right.\right.\right.}_{\text{High-Dimensional Non-convex Optimization.}}$$

even w/o activation, $F = L\left(\underline{\underline{W}}_L \cdot \underline{\underline{W}}_{L-1} \cdots \left(\underline{\underline{W}}_1\underline{x}+\underline{b}_1\right)+\cdots+\underline{b}_L - f(\underline{x})\right)$ is still non-convex (poly prog.).

Active research & still many open problems.

* SGD can at best converge to stationary pt., which can either local min, saddle pt. Can we escape from saddle pt.? If yes, how & how fast. $O(\mathrm{poly}\log^4(d)/\varepsilon^2)$ 
* "Landscape": Many theories to characterize "landscape" of $F(\underline{x})$. e.g., ? $\exists$ spurious local min (i.e., local $=$ or $\neq$ global min in NN.)
* "Overparameterized Regime".

$\underline{y} = \underline{\underline{W}}\,\underline{x} + \underline{b}$ $\quad$ $\underline{W} \in \mathbb{R}^{m \times n}$. $\quad$ $n \gg m$. large null space.

large.

Can every global min generalization SGD.

* Optimal choice of arch ?