CS 480/680

2/24/26

# Lecture 12

SGD & Heavy Ball

[HW5 out W7
[HW6 out W8]
HW7 out W10
[HW8 out W11]

Q2: 3/5 or
3/10
Q3: W11

# Lecture Notes IV – Neural Networks, Part 2

Marina Meilă
mmp@uwaterloo.ca

With Thanks to Pascal Poupart & Gautam Kamath
Cheriton School of Computer Science
University of Waterloo

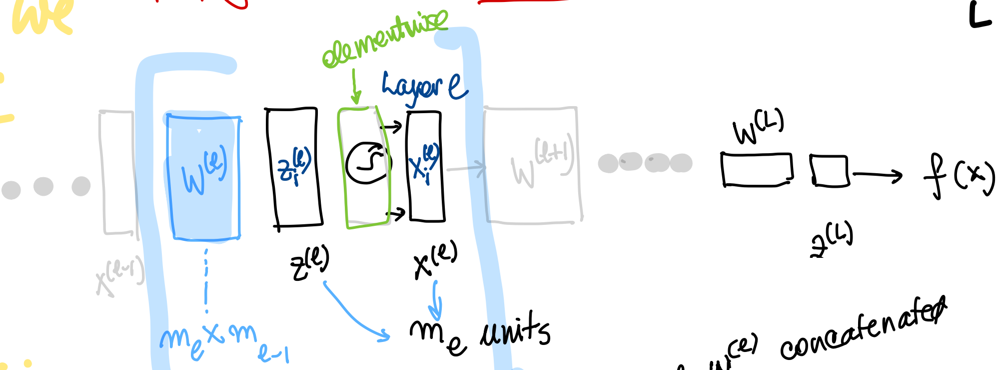
February 8, 2026

Where we left off before reading week...

# Backpropagation

## Training a multilayer network

L layers




**Parameters** $\mathbb{W} = \{ W^{(1)}, \dots W^{(L)} \}$ $\rightarrow$ $\mathbb{W} \in \mathbb{R}^P$ $\rightarrow$ all $w_{ij}^{(\ell)}$ concatenated

**Data** $\mathcal{D} = \{ (x^i, y^i), i = 1:n \}$

TRAIN: by G.Descent on loss $\mathcal{L}$

1. Init $\mathbb{W}$ with small random values $\quad w_{ij}^{(\ell)} \sim N(0, \sigma^2 \frac{1}{m_\ell})$ $\quad \sigma^2$ small

2. for $t = 1, 2, \dots$

$$\left[ \mathbb{W}^{t+1} \leftarrow \mathbb{W}^t - \underbrace{\eta}_{\text{step size}} \overbrace{\frac{\partial \mathcal{L}(\mathbb{W}^t)}{\partial \mathbb{W}}}^{g \,=\, \text{gradient}} \right] \Leftrightarrow \left[ w_j^{t+1} \leftarrow w_j^t - \eta \frac{\partial \mathcal{L}}{\partial w_j} \right]$$

$y^i, x^i$ ← training ex. i

$x^{(\ell)}$ ← layer $\ell$

$x_j$, $k$ } unit

$x_2^{(0)i}$ = input value layer 0, 2nd attribute i-th example

Training a single unit ✓

Training a 2-layer network ✓

Training a $L$-layer network ✓

Backpropagation in practice ←

Practical remedies to I, T, S, L, O

**Reading** HTF Ch.: 11.3 Neural networks, Murphy Ch.: (16.5 neural nets), Bach Ch.: –, Deep Learning Book (Goodfellow, Bengio, Courville) 6.1-4, ResNet 7.6, ConvNet 9., Autoencoders 14.1, Dive Into Deep Learning 4.1-4.3.

# Backpropagation – some issues

- I Computation – how many ops / iteration? $\sim np$
- T Convergence – how many iterations ($T$) ?
- S Saturation – $\phi'(x) \approx 0$ for large $|z|$
- L Local minima
- 0 Overfitting?

# Ops/ iteration

- Number parameters $p$
  - $\mathbb{W} = \{W^{(l)} \in \mathbb{R}^{m_l \times m_{l-1}},\ l = 1 : L\}$
  - Hence $p \sim \sum_{l=1}^{L} m_{l-1} m_l$
- Forward pass: each $W_{ij}^{(l)}$ is used once to propagate from $x_j^{(l-1)}$ to $x_i^{(l)}$, for each data point
- Backward pass: each $W_{ij}^{(l)}$ is used once to propagate from $x_i^{(l)}$ to $x_j^{(l-1)}$, for each data point
- Update: each weight is updated once.
- Hence number operations / iteration is $\mathcal{O}(np)$
- This can become very large with deep networks, large data, and large input dimensions $d\,(= m_0)$

# T Convergence – how many iterations?

- **Theory** – Gradient Descent (GD) is "order 1", a slower method (asymptotically) compared to Newton.
  - 1-st order optimization method: uses only 1-st derivative info
  - 2-nd order optimization method: uses 1-st and 2-nd derivative info
  - 0-th order optimization method: uses no derivative info (only $\mathcal{L}$ values)
- In the case of neural networks, as for any $\mathcal{L}$ with local minima, the practical issues below are also relevant.
- Progress of training is slow when the gradient $\frac{\partial \mathcal{L}}{\partial W_{ij}^{(l)}}$ is $\approx 0$ away from the optimum
- When does this happen? $\mathcal{L}$ has valleys (in some direction $\frac{\partial \mathcal{L}}{\partial \mathbb{W}} \approx 0$) or plateaus, or saddles
- Saturation

Newton & variations :)

GD is slow

SGD very slow !

classical

opt

for ML

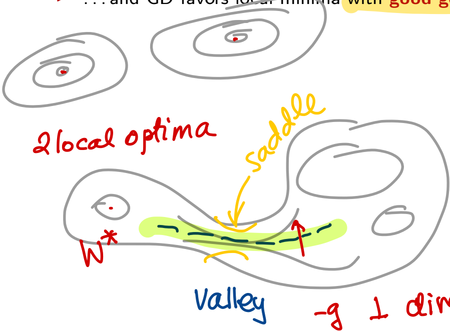
SGD ✓ :)

GD slow

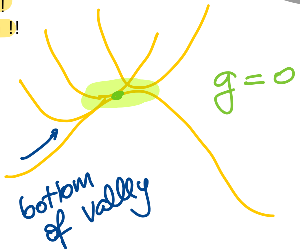
{ Heavy Ball

# L Local optima

- $\mathcal{L}$ has **many local optima** and valleys (often between local optima) [and plateaus]
- It is not possible or expected to find the global optimum in $\mathbb{W}$ space (note also there are multiple equal optima)
- The goal of training is
  - to avoid the really bad local optima
  - to avoid stopping at saddles or plateaus
- Modern (very large) nn
  - have extremely many local minima
  - have extremely many good local minima
  - GD (and SGD) converge very close to the starting point!!
  - . . . and GD favors local minima with **good generalization** !!


-g ⊥ direction to W*



# Stochastic gradient descent – I,T,L

SGD

1. At each iteration $t$, select a **batch** $\mathcal{B} \subset \mathcal{D}$ of size $n' \ll n$ from the data
2. Calculate gradient and update only with respect to the samples in $\mathcal{B}$

SGD

$$\mathbb{W}^{t+1} \leftarrow \mathbb{W}^t - \eta \frac{1}{n'} \sum_{i \in \mathcal{B}} \frac{\partial \mathcal{L}(y^i, f(x^i))}{\partial \mathbb{W}} (\mathbb{W}^t) \tag{26}$$

GD

$$\mathbb{W}^{t+1} \leftarrow \mathbb{W}^t - \eta \underbrace{\frac{1}{n} \sum_{i \in \mathcal{D}} \frac{\partial \mathcal{L}}{\partial \mathbb{W}}(y^i, f(x^i, \mathbb{W}^t))}_{g^t}$$

$i$ data point

$g_i^t$ contribution of point $i$

SGD $g^t \leftarrow \operatorname{avg}_{i \in \mathcal{B}} g_i^t = g_\mathcal{B}^t$

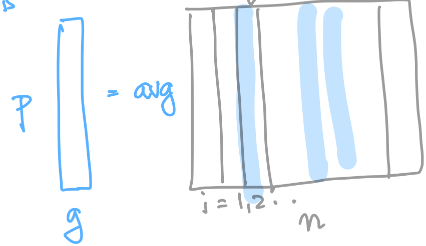
$g^t = \operatorname{avg}_{i \in \mathcal{D}} g_i^t$

$g_\mathcal{B}^t$ is random!

$\mathcal{B} \subset \mathcal{D}$

$|\mathcal{B}| = n' \ll n$ } selected unif. at random

$|\mathcal{D}| = n$



# Stochastic gradient descent – **I,T,L**

**SGD**

1. At each iteration $t$, select a **batch** $\mathcal{B} \subset \mathcal{D}$ of size $n' \ll n$ from the data
2. Calculate gradient and update only with respect to the samples in $\mathcal{B}$

$$\mathbb{W}^{t+1} \leftarrow \mathbb{W}^t - \eta \frac{1}{n'} \sum_{i \in \mathcal{B}} \frac{\partial \mathcal{L}(y^i, f(x^i))}{\partial \mathbb{W}}(\mathbb{W}^t) \tag{26}$$

noise

$g_{j\mathcal{B}} = g_j + \varepsilon_j$

weight $j$

Ex: $g = \text{avg}\{g_i\}$

$\sigma^2 = \text{Var}\{g_{i,j}\}$

weight $j$

$\text{avg } g_{\mathcal{B}} = g \iff \text{avg } \varepsilon_j = 0$

$\text{Var } \varepsilon_j = \frac{\sigma^2}{n'}$

$\text{std } \varepsilon_j = \frac{\sigma}{\sqrt{n'}}$

$n' \ll n$

$\boxed{n' = 1 \text{ possible}}$

Benefits: SGD GD

1) I : $n'p$ vs $np$ ✓

2) T : $\boxed{T' n' p}$ vs $\boxed{T n p}$ ← fewer epochs !!

total computation

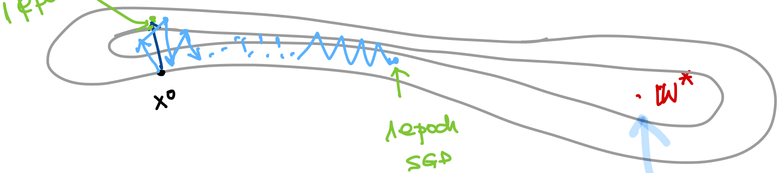
shallow

3) L : avoids saddles, ~~bad~~ local minima ✓

# Stochastic gradient descent – **I,T,L**

SGD
1. At each iteration $t$, select a **batch** $\mathcal{B} \subset \mathcal{D}$ of size $n' \ll n$ from the data
2. Calculate gradient and update only with respect to the samples in $\mathcal{B}$

$$\mathbb{W}^{t+1} \leftarrow \mathbb{W}^t - \eta \frac{1}{n'} \sum_{i \in \mathcal{B}} \frac{\partial \mathcal{L}(y^i, f(x^i))}{\partial \mathbb{W}} (\mathbb{W}^t) \tag{26}$$




SGD does NOT CONVERGE

$n = 10^6$

$n' = 1000$

$t' = \frac{n}{n'} = 1000 = \#\ \mathcal{B}'s$ to $\underbrace{\text{use all } \mathcal{D}}_{\text{1 epoch}}$

Fix – 1) compare $\frac{\|\mathbb{W}^t - \mathbb{W}^{t+t'}\|}{\|\mathbb{W}\|} < tol$ ← average $g_{\mathcal{B}}^t$ for $t \geq T_0$

$\tilde{g}^t = \text{avg}\{g_{\mathcal{B}}^t, g_{\mathcal{B}}^{t-1}, \ldots g_{\mathcal{B}}^{T_0}\}$

# Stochastic gradient descent – **I,T,L**

SGD
1. At each iteration $t$, select a **batch** $\mathcal{B} \subset \mathcal{D}$ of size $n' \ll n$ from the data
2. Calculate gradient and update only with respect to the samples in $\mathcal{B}$

$$\mathbb{W}^{t+1} \leftarrow \mathbb{W}^t - \eta \frac{1}{n'} \sum_{i \in \mathcal{B}} \frac{\partial \mathcal{L}(y^i, f(x^i))}{\partial \mathbb{W}} (\mathbb{W}^t) \tag{26}$$

Fix 1) compare $\frac{\|W^t - W^{t+t'}\|}{\|W\|} < tol$ ← average $g_{\mathcal{B}}^t$ for $t \geq T_0$

$\tilde{g}^t = \text{avg}\{g_{\mathcal{B}}^t, g_{\mathcal{B}}^{t-1}, \ldots g_{\mathcal{B}}^{T_0}\}$

2) $n' \nearrow n$

3) $\eta \searrow 0$

$\eta \leftarrow \eta^t \quad \eta < 1$

$\sum \eta^t = \frac{1}{1-\eta}$ NO

$\eta \sim \frac{1}{t} \quad \sum_t \eta^{(t)} = \infty$

$\eta^{(t)} \to 0$

# Stochastic gradient descent – **I,T,L**

SGD
1. At each iteration $t$, select a **batch** $\mathcal{B} \subset \mathcal{D}$ of size $n' \ll n$ from the data
2. Calculate gradient and update only with respect to the samples in $\mathcal{B}$

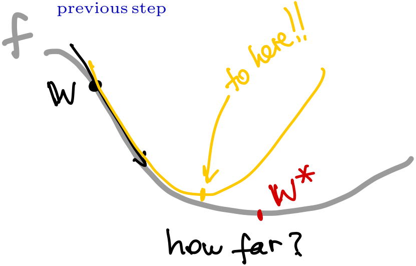
$$\mathbb{W}^{t+1} \leftarrow \mathbb{W}^t - \eta \frac{1}{n'} \sum_{i \in \mathcal{B}} \frac{\partial \mathcal{L}(y^i, f(x^i))}{\partial \mathbb{W}}(\mathbb{W}^t) \tag{26}$$

- What is achieved?
  - Faster gradient calculation! $\sim n'p$ instead of $\sim np$
  - $g_{\mathcal{B}} \approx g_{\mathcal{D}}$ where $g_{\mathcal{B},\mathcal{D}}$ represent the gradients on the batch, respectively entire $\mathcal{D}$
  - If $\mathcal{B}$ is a random subset, then $g_{\mathcal{B}}$ is a random variable with mean $g_{\mathcal{D}}$.
- Variations and refinements
  - use $n' = 1$ possible (update weights after every data point)
  - start with $n'$ small (to advance fast) then increase it (to reduce "noise")
  - naturally adapts to on-line learning (use $n' \geq 1$ data points, then discard them)
- Randomness in gradient can help avoid very poor local minimima, saddles, etc (not plateaus)
- Impact on **I**: faster iteration, **T**: arrrive faster near the local minimumum, **L**: as above
- Terminology: GD also called batch GD, SGD with $n' > 1$ also called minibatch SGD, some people understand by SGD the SGD with $n' = 1$ (typical in algorithm analysis); epoch a pass through the entire $\mathcal{D}$, e.g. $n/n'$ iterations of SGD.

# Can we mimic 2-nd order methods "cheaply"? (T,L)

- "2-nd derivative" is Hessian $\frac{\partial^2 \mathcal{L}}{\partial \mathbb{W}^2}$ a $p \times p$ matrix
- We want to get the benefits of 2-nd order in $\mathcal{O}(p)$ time.[4]
- Let $g \in \mathbb{R}^p$ denote a gradient or stochastic gradient ($p$ is still the number of parameters we are training)
- **Momentum (Heavy ball method)**

$$\mathbb{W}^{t+1} \leftarrow \mathbb{W}^t - \gamma\eta g^t + (1-\gamma)\underbrace{(\mathbb{W}^t - \mathbb{W}^{t-1})}_{\text{previous step}} \tag{27}$$




- (many variations exist, e.g. Nesterov method)
- Adaptive learning rates (coming next)

GD = 1st order (Taylor)

Newton: 2nd order

Hessian $\left[\frac{\partial^2 \mathcal{L}}{\partial w_j \partial w_{j'}}\right]_{j,j'=1:p}$ $p \times p$

[4] The factor $n$ or $n'$ is ignored, because it does not affect what we do.

# Heavy Ball / Momentum

$\gamma \in (0,1)$

GD

+ Mom

$$w^{t+1} \leftarrow w^t - \gamma\eta g^t + (1-\gamma)(w^t - w^{t-1})$$

$$w^t - w^{t-1} = -\gamma\eta g^{t-1} + (1-\gamma)(w^{t-1} - w^{t-2})$$

$$w^{t-1} - w^{t-2} = -\gamma\eta g^{t-2} + (1-\gamma)(\;\cdots\;)$$

.....

$$w^{t+1} \leftarrow w^t - \underbrace{\gamma\eta}_{\tilde{\eta}}\left\{g^t + (1-\gamma)g^{t-1} + (1-\gamma)^2 g^{t-2} + (1-\gamma)^3 g^{t-3} + \dots\right\}$$

weighted SUM of $g^{t, t-1, \dots}$