ML & Optimization Lecture 18

- Noncux optimization w/ SGD
- Backpropagation for general FFNs, and MLPs
- Example of 2-layer NN for Fashion MNIST

FFN: L layers (0 input layer, L output layer)

n_l is the number of neurons on layer l $\quad W^l \in \mathbb{R}^{n_l \times n_{l-1}}$

σ activation function

$a^l \in \mathbb{R}^{n_l}$ preactivations, $\quad a^l = W^l o^{l-1} + b^l$

$o^l \in \mathbb{R}^{n_l}$ activation, $\quad o^l = \sigma(a^l)$

$\qquad\qquad = \sigma(W^l o^{l-1} + b^l)$

$b^l \in \mathbb{R}^{n_l}$

output of NN given an input $x \in \mathbb{R}^{n_0}$ is

$$o^L(x) = o^L(W^L \sigma(\cdots \sigma(W^1 x + b^1) \cdots) + b^L)$$

Last time: NNs are parametrized DAG of computation
Once the architecture is fixed, we learn the parameters of the NN via (for FFNs)

$$\mathcal{Z}_* = \{W_*^\ell, b_*^\ell\} = \underset{\substack{W^\ell \in \mathbb{R}^{n_\ell \times n_{\ell-1}} \\ b^\ell \in \mathbb{R}^{n_\ell}}}{\operatorname{argmin}} \quad f(\mathcal{Z})$$

where $f(\mathcal{Z}) = \frac{1}{n}\sum_{i=1}^{n} \ell(o^L(x_i), y_i) + \lambda \sum_{\ell=1}^{L} R_\ell(W^\ell, b^\ell)$

where R_ℓ is a layer-specific regularizer, e.g. a common choice is weight decay on W^ℓ:

$$R_\ell(W^\ell, b^\ell) = \|W^\ell\|_F^2 = \sum_{i \in [n_\ell]} \sum_{j \in [n_{\ell-1}]} (W^\ell)_{ij}^2$$

which encourages small weights

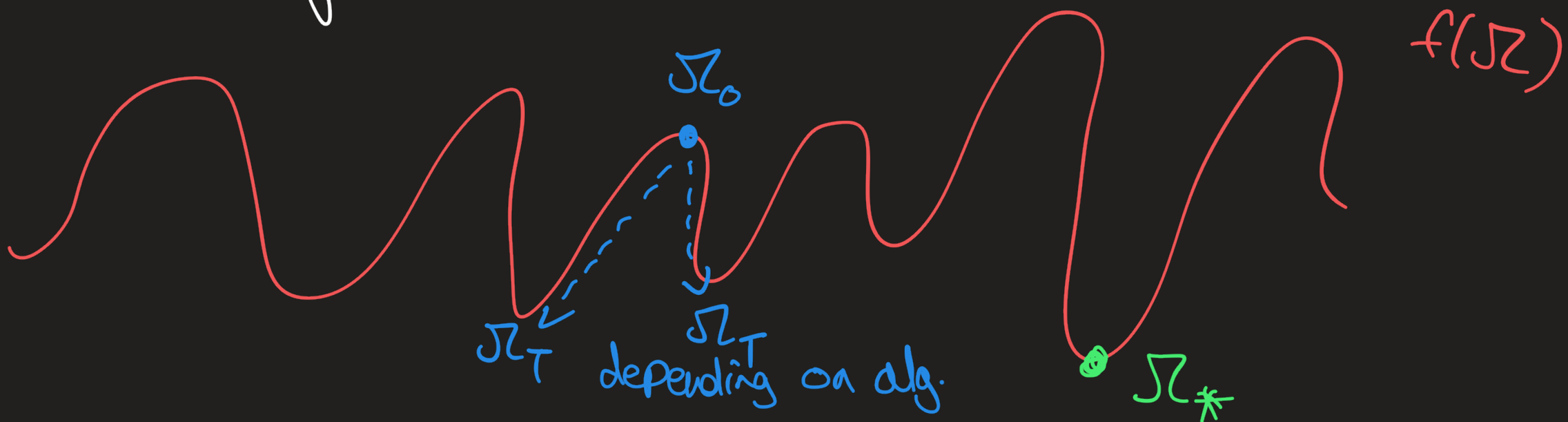
This is in general a non-convex optim problem, so guarantees of the forms:

(1) $f(\mathcal{Z}_T) - f^* \leq \varepsilon$ convergence of func values (ε-subopt.) to global min

(2) $\|\mathcal{Z}_T - \mathcal{Z}_*\|_F^2 \leq \varepsilon$ convergence of parameters to global min

$$\| \quad \| = \sum_{\ell \in [L]} \left[\|W^\ell - W_*^\ell\|_F^2 + \|b^\ell - b_*^\ell\|_2^2 \right]$$

do not hold, because in case (2) we can only hope to converge to local minimizers, and in case (1) there is no guarantee to a global minimum value.



In the case of general nonconvex optim, we guarantee convergence to approximate stationary points, i.e. parameters $\mathcal{Z}$ satisfying

$$\|\nabla f(\mathcal{Z})\|_F^2 = \sum_{\ell \in [L]} \left(\|\nabla_{\omega^\ell} f(\mathcal{Z})\|_F^2 + \|\nabla_{b^\ell} f(\mathcal{Z})\|_2^2 \right) \leq \varepsilon.$$

We call such an $\mathcal{Z}$ an ε-approximate stationary point. Intuitively, this suggests $\mathcal{Z}$ is near an extremum of f (could be max, min, or saddle pt). For certain classes of functions, we can quantify this intuition.

We use first-order algs, especially (adaptive) stochastic gradient descent, to train NNs. Luckily we can prove that first-order algs converge to ε-approx stat. pts on smooth non-convex functions.

Ex

If f is L-smooth, GD w/ stepsize $\alpha = \frac{1}{L}$ satisfy

$$\frac{1}{T}\sum_{i=1}^{T} \|\nabla f(x_{i-1})\|_2^2 \leq \frac{2LR}{T}$$

where $R = f(x_0) - f^*$

Note that this implies

$$\min_{i \in [T]} \|\nabla f(x_{i-1})\|_2^2 \leq \frac{1}{T}\sum_{i=1}^{T} \|\nabla f(x_{i-1})\|_2^2 \leq \varepsilon$$

for $T = O\left(\frac{1}{\varepsilon}\right)$, so with that many iterates we visit at least one ε-stat pt.

It also implies that if I randomly choose an iterate x from $x_0, \dots, x_{T-1}$

$$\mathbb{E}\|\nabla f(x)\|_2^2 \leq \varepsilon$$

so x is an ε-approx stat. point in expectation

Proof

$$f(x_t) \overset{\text{b/c of L-smoothness of } f}{\leq} f(x_{t-1}) + \langle \nabla f(x_{t-1}), x_t - x_{t-1} \rangle + \frac{L}{2}\|x_t - x_{t-1}\|_2^2$$

recall $x_t = x_{t-1} - \alpha \nabla f(x_{t-1})$, so

$$= f(x_{t-1}) - \alpha \|\nabla f(x_{t-1})\|_2^2 + \frac{\alpha^2 L}{2}\|\nabla f(x_{t-1})\|_2^2$$

$$= f(x_{t-1}) + \alpha\left(\frac{\alpha L}{2} - 1\right)\|\nabla f(x_{t-1})\|_2^2$$

since $\alpha = \frac{1}{L}$, we have

$$f(x_t) - f(x_{t-1}) \leq -\frac{1}{2L}\|\nabla f(x_{t-1})\|_2^2$$

or equivalently

$$\frac{1}{2L}\|\nabla f(x_{t-1})\|_2^2 \leq f(x_{t-1}) - f(x_t)$$

now sum these inequalities over $t = 1, \dots, T$:

$$\frac{1}{2L} \sum_{t=1}^{T} \| \nabla f(x_{t-1}) \|_2^2 \leq \sum_{t=1}^{T} \left[f(x_{t-1}) - f(x_t) \right]$$

$$= \left[f(x_0) - f(x_1) \right] + \left[f(x_1) - f(x_2) \right] + \cdots$$
$$+ \left[f(x_{T-1}) - f(x_T) \right]$$

$$= f(x_0) - f(x_T)$$

$$\leq f(x_0) - f^* = R$$

and now multiply both sides by $\frac{2L}{T}$, to get

$$\frac{1}{T} \sum_{t=1}^{T} \| \nabla f(x_{t-1}) \|_2^2 \leq \frac{2LR}{T}$$

What about SGD? We can imagine since g_t is only an estimate of $\nabla f(x_t)$, we will pay the price in the form of slower convergence.

Assume f is L-smooth, and

- $\mathbb{E}[g_t \mid x_t] = \nabla f(x_t)$
- $\mathbb{E}\|g_t - \nabla f(x_t)\|_2^2 \le \sigma^2$

and let R be as before, after T rounds of SGD

$$\begin{cases} \mathbb{E}\left(\frac{1}{T}\sum_{t=1}^{T}\|\nabla f(x_{t-1})\|_2^2\right) \le \frac{2LR}{T} + \sigma^2 & \text{if } \alpha = \frac{1}{2L} \\ \mathbb{E}\left(\frac{1}{T}\sum_{t=1}^{T}\|\nabla f(x_{t-1})\|_2^2\right) \le 4\sqrt{\frac{RL\sigma^2}{T}} & \text{if } \alpha = \sqrt{\frac{R}{TL\sigma^2}} \end{cases}$$

and $T \ge \frac{4RL}{\sigma^2}$

We now know SGD works in a specific quantifiable sense. In practice, we reach local minima and their quality is good.

The practical question left is: how to compute

$$\nabla f(\mathcal{Z}), \text{ where } \mathcal{Z} = \{W^\ell, b^\ell\}_{\ell=1}^{L} \text{ and}$$

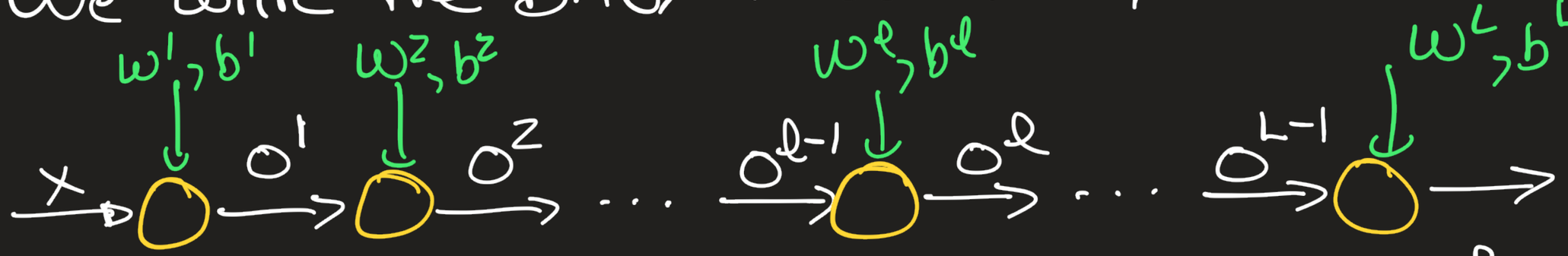
$$f(\mathcal{Z}) := \frac{1}{n}\sum_{i=1}^{n} \ell(o^L(x_i), y_i) + \lambda \sum_{\ell=1}^{L} R_\ell(W^\ell, b^\ell)$$

WLOG we will consider the case $n=1$, so

$$f(\mathcal{Z}) := \ell(o^L(x), y) + \lambda \sum_{\ell=1}^{L} R(W^\ell, b^\ell)$$

Recall this is loss for a general FFN

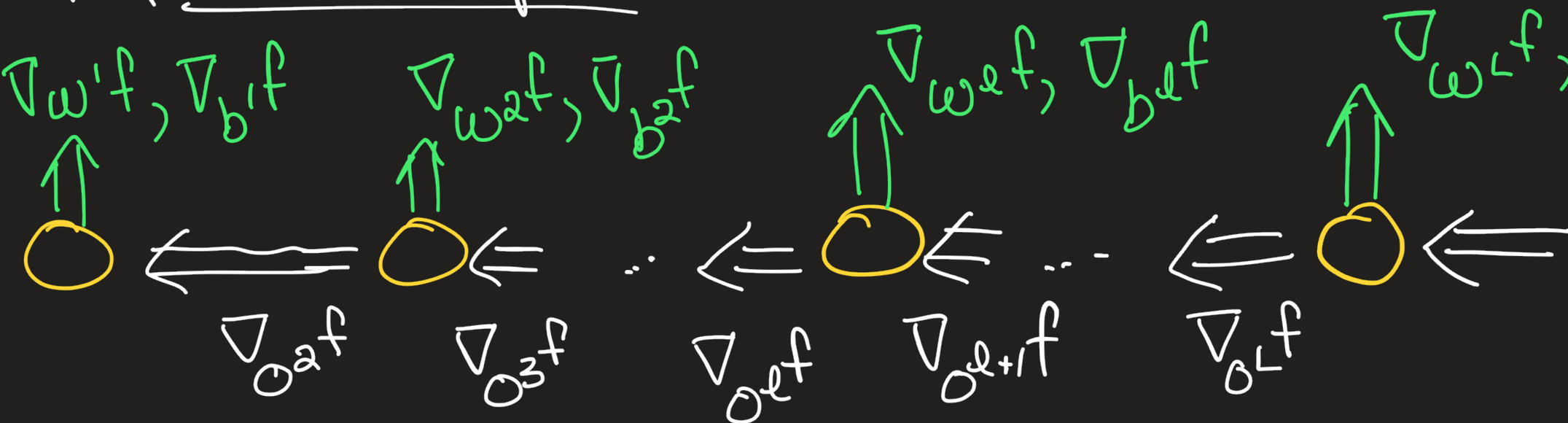
We write the DAG associated w/ f:



$$f = l(o^L, y) + \lambda R(\mathcal{W})$$

We see that computing the value of each layer is done in a <u>forward pass</u> : $o^\ell = o^\ell(o^{\ell-1})$

Similarly, gradients w.r.t. parameters can be computed in a <u>backward pass</u> (chain rule):



Justification: Chain Rule

Consider $h: \mathbb{R}^n \to \mathbb{R}^p$ and $r: \mathbb{R}^p \to \mathbb{R}$

Define $\phi(z) = r(h(z))$

[Think of $h: \mathbb{R}^{n_{l-1}} \to \mathbb{R}^{n_l}$ as one hidden layer $\Rightarrow h = o^l$
and $r: \mathbb{R}^{n_{l+1}} \to \mathbb{R}$ as $l(o^L(o^{L-1}(\cdots(o^{l+1}(z))\cdots)$]

We claim that

$$\nabla_\phi(z) = J_h(z)^T \nabla_r(h(z))$$

where $$J_h(z) = \left[\frac{\partial h_i}{\partial z_j}\right]_{\substack{i=1,\cdots,p \\ j=1,\cdots,n}} \in \mathbb{R}^{p \times n}$$

is the Jacobian of h at z.

Notice $J_h(z) = \begin{bmatrix} \nabla h_1(z)^T \\ \vdots \\ \nabla h_p(z)^T \end{bmatrix}$.

Also it is a fact that for any function g, the gradient is the <u>only</u> vector that satisfies

$$g(x+\Delta) = g(x) + \langle \nabla g(x), \Delta \rangle + h.o.t.$$

So to compute $\nabla \phi(z)$, we expand

$$\phi(z+\Delta) = r(h(z+\Delta)) = r\left(\begin{bmatrix} h_1(z+\Delta) \\ \vdots \\ h_p(z+\Delta) \end{bmatrix}\right)$$

$$= r\left(\begin{bmatrix} h_1(z) + \langle \nabla h_1(z), \Delta \rangle + h.o.t. \\ \vdots \\ h_p(z) + \langle \nabla h_p(z), \Delta \rangle + h.o.t. \end{bmatrix}\right)$$

$$= r\left(h(z) + \begin{bmatrix} \nabla h_1(z)^T \\ \vdots \\ \nabla h_p(z)^T \end{bmatrix} \Delta + h.o.t.\right)$$

$$= r\left(h(z) + J_h(z) \circ \Delta + h.o.t.\right)$$

$$= r(h(z)) + \langle \nabla r(h(z)), J_h(z) \circ \Delta + h.o.t. \rangle + h.o.t.$$

$$= r(h(z)) + \langle \nabla r(h(z)), J_h(z)\Delta \rangle + h.o.t.$$

$$= r(h(z)) + \langle J_h(z)^T \nabla r(h(z)), \Delta \rangle + h.o.t.$$

$$= \phi(z) + \langle J_h(z)^T \nabla r(h(z)), \Delta \rangle + h.o.t.$$

so by uniqueness of gradients,

$$\nabla_z \phi(z) = J_h(z)^T \nabla r(h(z))$$

Returning to backward pass

First key observation: if we can compute $\Leftarrow$

$$\nabla_{o^\ell} f = \left[\frac{\partial f}{\partial (o^\ell)_i} \right]_{i=1}^{n_\ell}$$

then the chain rule allows us to compute

$$\nabla_{w^\ell} f = J_{o^\ell}(w^\ell)^T \nabla_{o^\ell} f + \lambda \nabla_{w^\ell} R_\ell$$

$$(*) \quad \nabla_{b^\ell} f = J_{o^\ell}(b^\ell)^T \nabla_{o^\ell} f + \lambda \nabla_{b^\ell} R_\ell$$

e.g. take $h = o^\ell$ which takes b^ℓ as input

and $r = l \circ o^L \circ o^{L-1} \circ \cdots \circ o^{\ell+1}$ which takes o^ℓ as input

then $(*)$ follows from the chain rule

Second key observation

We can compute ∇ recursively. Since f depends on o^l only through o^{l+1}, we have by the chain rule that

$$(*) \quad \nabla_{o^l} f = J_{o^{l+1}}(o^l)^T \nabla_{o^{l+1}} f$$

here $h = o^{l+1}$ takes o^l as input

$r = l \circ o^L \circ o^{L-1} \circ \cdots \circ o^{l+2}$ takes o^{l+1} as input

so

$$J_h(o^l)^T = J_{o^{l+1}}(o^l)^T$$

and

$$\nabla_{o^{l+1}} r = \nabla_{o^{l+1}} f$$

so

(*) follows from the chain rule

Putting these pieces together we see that we can recursively compute $\nabla_{w^\ell} f$, $\nabla_{b^\ell} f$ in a single backward pass as follows:

Backprop for FFNs

$$\text{temp} \leftarrow \nabla_{o^L} f$$

$$\nabla_{b^L} f = J_{o^L}(b^L)^T \text{temp} + \lambda \nabla_{b^L} R_L$$

$$\nabla_{w^L} f = J_{o^L}(w^L)^T \text{temp} + \lambda \nabla_{w^L} R_L$$

for $\ell = L-1, \dots, 1$:

$$\text{temp} \leftarrow J_{o^{\ell+1}}(o^\ell)^T \text{temp}$$

$$\nabla_{b^\ell} f \leftarrow J_{o^\ell}(b^\ell)^T \text{temp} + \lambda \nabla_{b^\ell} R_\ell$$

$$\nabla_{w^\ell} f \leftarrow J_{o^\ell}(w^\ell)^T \text{temp} + \lambda \nabla_{w^\ell} R_\ell$$