

- Dropout - regularization, makes convergence slower, may improve test quality
- Batch Normalization - implicit regularization, improves convergence ensuring that neurons are normalized

But what are the theoretic bounds of expressivity of fully-connected networks?

Universal Approximation Theorem (UAT, Cybenko, 1989)

Let $\mathcal{E}: \mathbb{R} \rightarrow \mathbb{R}$ be an activation function, which is:

1) continuous

2) sigmoid, i.e.

$$\left\{ \begin{array}{l} \lim_{x \rightarrow -\infty} \mathcal{E}(x) = 0 \\ \lim_{x \rightarrow +\infty} \mathcal{E}(x) = 1 \end{array} \right.$$

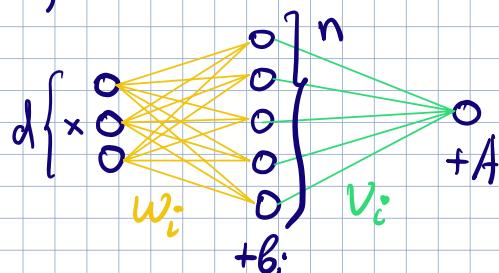
Fix the desired error level $H \epsilon > 0$
 Then, for any continuous $f: [0,1]^d \rightarrow \mathbb{R}$, $f \in C([0,1]^d)$, there exists a 2-layer MLP, which uniformly approximates f :

$\exists n \in \mathbb{N}, \{w_i\}_{i=1}^n, w_i \in \mathbb{R}^d, \{b_i\}_{i=1}^n, b_i \in \mathbb{R},$

$\{v_i\}_{i=1}^n, v_i \in \mathbb{R}, A \in \mathbb{R} :$

$$g(x) = A + \sum_{i=1}^n v_i \mathcal{E}(w_i^\top x + b_i)$$

$$\sup_{x \in [0,1]^d} |f(x) - g(x)| \leq \epsilon$$



Thus, MLPs are universal approximators
(in theory)

But in practice we often have:

- 1) n (hidden layer size) may be extremely large
- 2) It is difficult to find a global optimum using gradient optimization
- 3) Overfitting to train set

Optimization

SGD (Stochastic Gradient Descent)

$\{(x_i, y_i)\}_{i=1}^e$ - train set

B - batch size

$L(\cdot, \cdot)$ - loss function

w - all trainable parameters of the model

$f_w(\cdot)$ - neural network

At step t we have:

$$\hat{L}^t(w) = \frac{1}{B} \sum_{i=1}^B L(f_w(x_i), y_i), b_i \sim U\{1, \dots, e\}$$

SGD step:

$$\begin{cases} g_t = \nabla_w \hat{L}^t(w_{t-1}) \\ m_t = \mu \cdot m_{t-1} + g_t \quad - \text{momentum} \\ w_t = w_{t-1} - \eta_t \cdot m_t \end{cases}$$

$m_0 = 0, 0 \leq \mu \leq 1, \text{ usually } \mu = 0.9$

Adding L_2 regularization

$$L^t(w) \rightarrow L_\lambda^t(w) := L^t(w) + \frac{\lambda}{2} \|w\|_2^2$$

$$\nabla_w \|w\|_2^2 = 2w$$

$$\begin{cases} g_t = \nabla_w h^t(w_{t-1}) + \lambda w_{t-1} \\ m_t = \mu \cdot m_{t-1} + g_t \\ w_t = w_{t-1} - \eta_t \cdot m_t \end{cases}$$

$$(1) w_{t-1} - \eta_t (\mu \cdot m_{t-1} + g_t) =$$

$$= w_{t-1} - \eta_t (\mu \cdot m_{t-1} + \nabla_w h^t(w_{t-1}) + \lambda w_{t-1}) =$$

$$= w_{t-1} \underbrace{(1 - \eta_t \lambda)}_{\text{weight decay}} - \eta_t \underbrace{(\mu \cdot m_{t-1} + \nabla_w h^t(w_{t-1}))}_{m_t^{\text{old}}}$$

weight decay

Main idea: decaying the weights by a factor less than 1 before the step is equivalent to L_2 regularization of weights

Adam

1. Adaptive learning rate (as in AdaGrad, RMSProp)
2. Momentum weight decay

$$\begin{cases} g_t = \nabla_w h^t(w_{t-1}) + \lambda w_{t-1} \\ m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t \\ v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2 \\ \hat{m}_t = \frac{m_t}{1 - \beta_1^t} \quad \hat{v}_t = \frac{v_t}{1 - \beta_2^t} \\ w_t = w_{t-1} - \eta_t \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon} \end{cases}$$

element wise square
 β_1, β_2 to power t
 division and sqrt are elementwise

$$m_0 = 0, v_0 = 0$$

$\beta_1, \beta_2, \varepsilon$ - hyperparameters
 $\beta_1 = 0.9, \beta_2 = 0.999, \varepsilon = 10^{-8}$

$$m_t = (1-\beta_1)g_t + \beta_1 m_{t-1} =$$

$$= (1-\beta_1)g_t + \beta_1(1-\beta_1)g_{t-1} + \beta_1^2 m_{t-2} =$$

$$= (1-\beta_1)g_t + \beta_1(1-\beta_1)g_{t-1} + \dots + \beta_1^{t-1}(1-\beta_1)g_1 + \cancel{\beta_1^t m_0} =$$

$$= \sum_{i=0}^{t-1} \beta_1^i (1-\beta_1)g_{t-i}$$

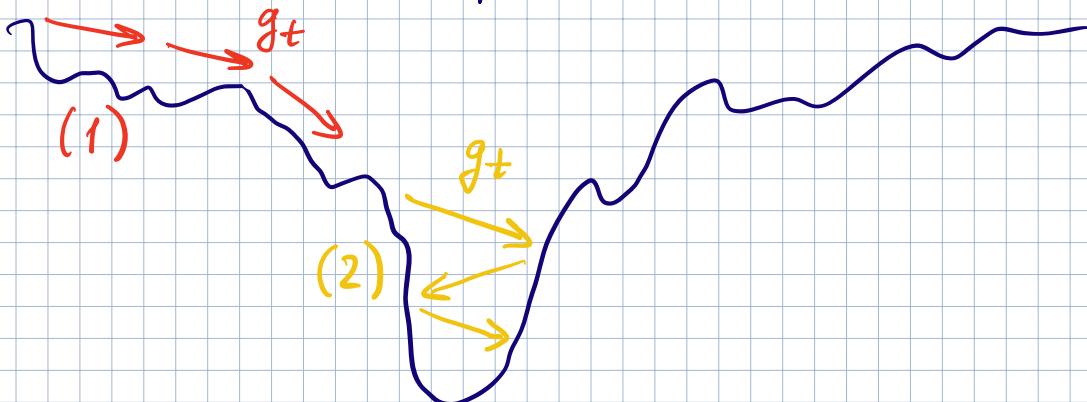
$$\hat{m}_t = \frac{m_t}{1-\beta_1^t} = \frac{1-\beta_1}{1-\beta_1^t} \sum_{i=0}^{t-1} \beta_1^i g_{t-i} = \frac{1}{\sum_{j=0}^{t-1} \beta_1^j} \sum_{i=0}^{t-1} \beta_1^i g_{t-i} =$$

$$= \left\{ \alpha_i := \frac{\beta_1^i}{\sum_{j=0}^{t-1} \beta_1^j}, \alpha_i > 0, \sum_{i=0}^{t-1} \alpha_i = 1 \right\} =$$

$$= \sum_{i=0}^{t-1} \alpha_i g_{t-i} = \mathbb{E}g$$

$$\text{Similarly, } \hat{v}_t = \mathbb{E}g^2 = \text{Var}(g) + (\mathbb{E}g)^2$$

Now, consider 2 possible situations



1. Far from optimum

Steps in the "same" direction $\Rightarrow \text{Var } g \approx 0$

$$w_t = w_{t-1} - \eta_t \frac{\hat{m}_t}{\sqrt{\hat{v}_t + \epsilon}} \quad \begin{array}{l} \text{- effective step size} \\ \approx \eta_t \end{array}$$

$$\frac{\mathbb{E} g}{\sqrt{\text{Var}(g) + (\mathbb{E} g)^2 + \epsilon}} \approx \frac{\mathbb{E} g}{\sqrt{(\mathbb{E} g)^2}} \approx 1$$

≈ 0

2. Close to optimum

Direction of gradient varies $\Rightarrow \text{Var } g \gg 0$

$$\frac{\mathbb{E} g}{\sqrt{\text{Var}(g) + (\mathbb{E} g)^2 + \epsilon}} < 1 \quad \begin{array}{l} \text{effective step size} \\ \text{becomes smaller, } < \eta_t \end{array}$$

Now, let's track the weight decay factor:

$$\begin{aligned} w_t &= w_{t-1} - \eta_t \frac{\hat{m}_t}{\sqrt{\hat{v}_t + \epsilon}} = w_{t-1} - \eta_t \frac{m_t}{(1-\beta_1^t)\sqrt{\hat{v}_t + \epsilon}} = \\ &= w_{t-1} - \eta_t \frac{\beta_1 \cdot m_{t-1} + D_w L^t(w_{t-1}) + \lambda w_{t-1}}{(1-\beta_1^t)\sqrt{\hat{v}_t + \epsilon}} = \\ &= w_{t-1} \left(1 - \frac{\eta_t \lambda}{(1-\beta_1^t)\sqrt{\hat{v}_t + \epsilon}} \right) - \dots \end{aligned}$$

All weights are decayed with different coefficients,
this is not what we wanted originally

Adam W

Processes weight decay differently to vanilla Adam:

$$\begin{cases} g_t = \nabla_w L^+(w_{t-1}) \\ m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t \\ v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^2 \\ \hat{m}_t = \frac{m_t}{1-\beta_1^t} \quad \hat{v}_t = \frac{v_t}{1-\beta_2^t} \\ w_t = w_{t-1} (1 - \eta_t \lambda) - \eta_t \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon} \end{cases}$$

Learning rate schedules

How to choose learning rate (LR) $\eta_t - ?$

LR is often update once per training epoch

- Constant $\eta_t = \eta_0$
Often gives suboptimal quality
- StepLR $\eta_t = \begin{cases} \eta_0, & 1 \leq t < t_0 \\ \eta_1, & t_0 \leq t < t_1 \\ \dots \end{cases}$

$$\eta_0 > \eta_1 > \dots$$

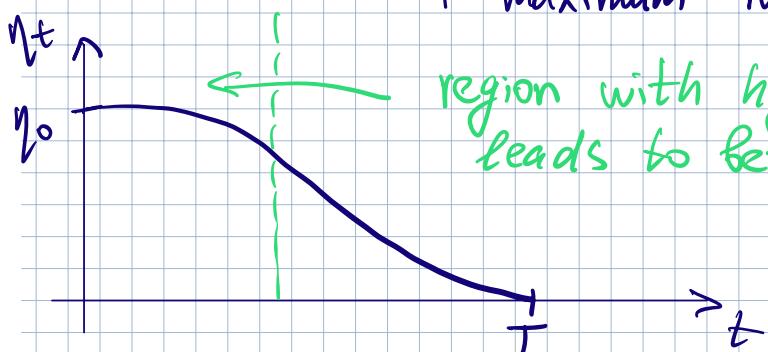
- Exponential LR $\eta_t = \gamma \eta_{t-1}, \gamma < 1$



• Cosine LR

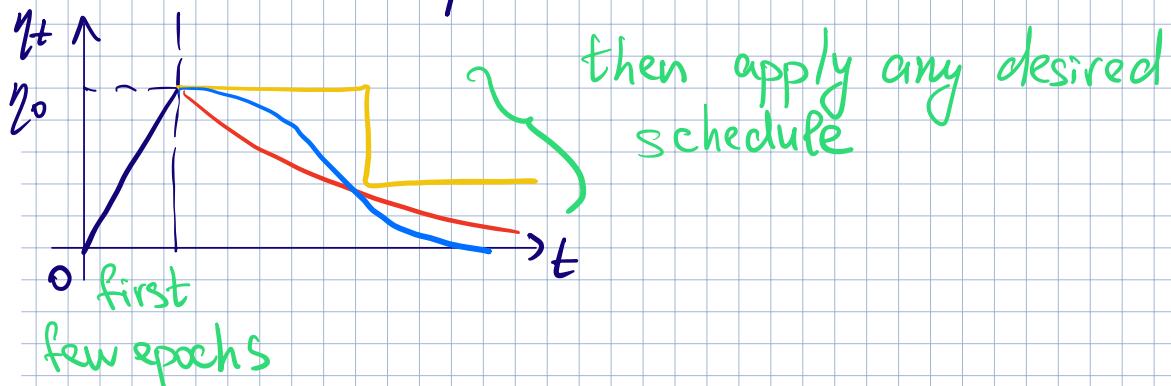
$$\eta_t = \eta_0 \cdot \frac{1}{2} (1 + \cos(\frac{\pi t}{T}))$$

T - maximum number of epochs



region with high learning rate
leads to better generalization

• Linear warmup LR



• Reduce LR on plateau

Validation set is required

Strategy: decrease LR when validation loss stagnates

