- 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 $\sigma: \mathbb{R} \to \mathbb{R}$ be an activation function, which is:

1) continuous
2) sigmoid, i.e. $\begin{cases} \lim\limits_{x \to -\infty} \sigma(x) = 0 \\ \lim\limits_{x \to +\infty} \sigma(x) = 1 \end{cases}$

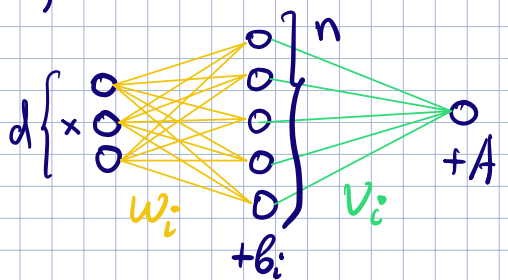
Fix the desired error level $\forall \varepsilon > 0$

Then, for any continuous $\forall f: [0,1]^d \to \mathbb{R}$, $f \in C([0,1]^d)$, there exists a 2-layer MLP, which uniformly approximates $f$:

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

$$g(x) = A + \sum_{i=1}^{n} v_i \sigma(w_i^T x + b_i)$$

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



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)

$\{(x_i, y_i)\}_{i=1}^{\ell}$ – 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:

$$\mathcal{L}^t(w) = \frac{1}{B} \sum_{i=1}^{B} L\left(f_w(x_i), y_i\right), \quad t_i \sim \mathcal{U}(\{1, \dots, \ell\})$$

SGD step:

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

$m_0 = 0$, $0 \le \mu \le 1$, usually $\mu = 0$

Adding $L_2$ regularization

$$\mathcal{L}^t(w) \to \mathcal{L}^t_\lambda(w) := \mathcal{L}^t(w) + \frac{\lambda}{2}\|w\|_2^2$$

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

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

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

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

$$= w_{t-1}(1 - \eta_t\lambda) - \eta_t \underbrace{(\mu \cdot m_{t-1} + \nabla_w \mathcal{L}^t(w_{t-1}))}_{m_t^{old}}$$

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

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

element wise square

$\beta_1, \beta_2$ to power $t$

division and sqrt are element wise

$m_0 = 0$, $v_0 = 0$

$\beta_1, \beta_2, \varepsilon$ – hyperparameters

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

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

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

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

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

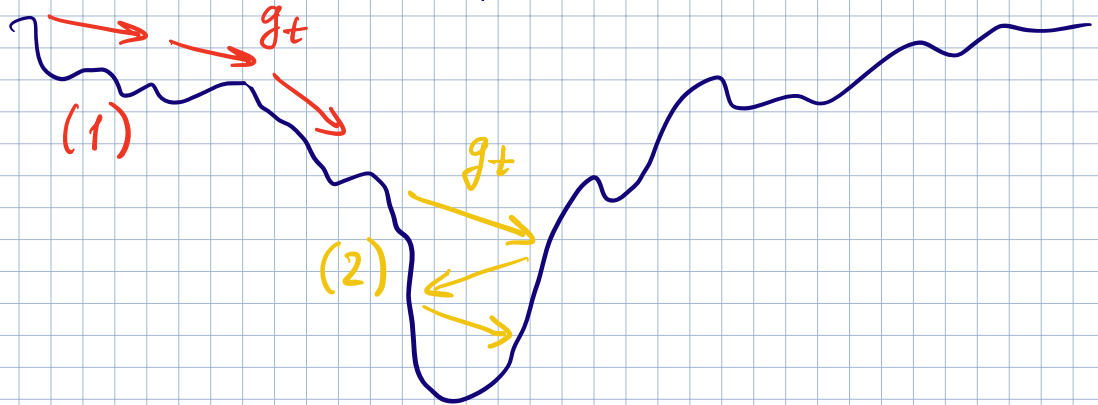
$$\hat{m}_t = \frac{m_t}{1-\beta_1^t} = \frac{1-\beta_1}{1-\beta_1^t} \sum_{i=0}^{t-1} \beta_1^i g_{t-i} = \frac{1}{\sum_{j=0}^{t-1} \beta_1^j} \sum_{i=0}^{t-1} \beta_1^i g_{t-i} =$$

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

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

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

Now, consider 2 possible situations



1. Far from optimum

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

$$w_t = w_{t-1} - \eta_t \frac{\hat{m}_t}{\sqrt{\hat{v}_t + \varepsilon}}$$ - effective step size $\approx \eta_t$

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

($\mathrm{Var}(g) \approx 0$)

2. Close to optimum

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

$$\frac{\mathbb{E} g}{\sqrt{\mathrm{Var}(g) + (\mathbb{E} g)^2 + \varepsilon}} < 1$$

effective step size becomes smaller, $< \eta_t$

($\mathrm{Var}(g) \gg 0$)

Now, let's track the weight decay factor:

$$w_t = w_{t-1} - \eta_t \frac{\hat{m}_t}{\sqrt{\hat{v}_t + \varepsilon}} = w_{t-1} - \eta_t \frac{m_t}{(1-\beta_1^t)\sqrt{\hat{v}_t + \varepsilon}} =$$

$$= w_{t-1} - \eta_t \frac{\beta_1 \cdot m_{t-1} + \nabla_w L^t(w_{t-1}) + \lambda w_{t-1}}{(1-\beta_1^t)\sqrt{\hat{v}_t + \varepsilon}} =$$

$$= w_{t-1}\left(1 - \frac{\eta_t \lambda}{(1-\beta_1^t)\sqrt{\hat{v}_t + \varepsilon}}\right) - \dots$$

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

## AdamW

Processes weight decay differently to vanilla Adam:

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

## Learning rate schedules

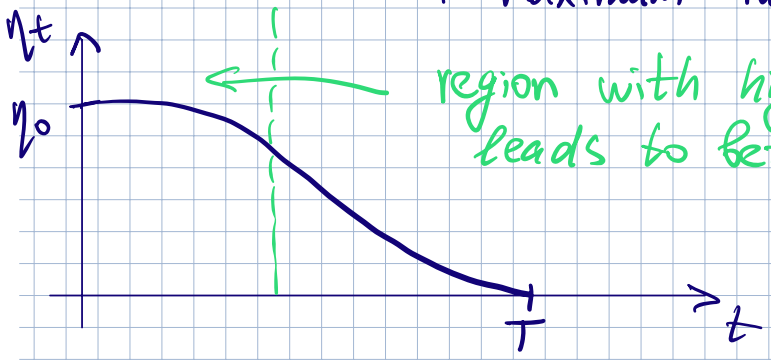
How to choose learning rate (LR) $\eta_t$ - ?

LR is often update once per training epoch

- Constant $\eta_t = \eta_0$
  Often gives suboptimal quality
- StepLR $\eta_t = \begin{cases} \eta_0, & 1 \le t < t_0 \\ \eta_1, & t_0 \le t < t_1 \\ \dots \end{cases}$
  $\eta_0 > \eta_1 > \dots$
- Exponential LR $\eta_t = \gamma \eta_{t-1}$, $\gamma < 1$

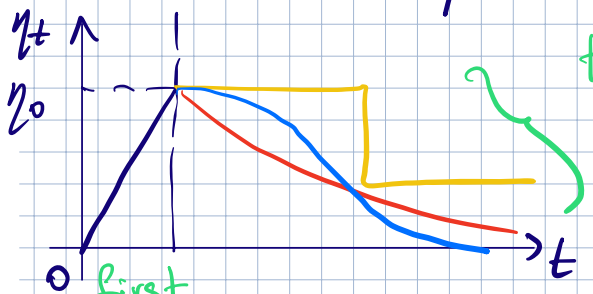
- **Cosine LR** $\eta_t = \eta_0 \cdot \frac{1}{2}\left(1 + \cos\left(\frac{\pi t}{T}\right)\right)$

$T$ – maximum number of epochs




region with high learning rate leads to better generalization

- **Linear warmup LR**




first few epochs

then apply any desired schedule

- **Reduce LR on plateau**

Validation set is required

Strategy: decrease LR when validation loss stagnates