# Deep Learning (BEV033DLE)
# Lecture 3.

Czech Technical University in Prague

- Neural networks are universal approximators
- Loss functions for classification and regression
- Generalisation bounds and generalisation errors for neural classifiers
- Generalisation bounds for regression models

# Neural networks as universal approximators

**Question:** Can we approximate every function/mapping by a neural network?

**Boolean functions:** Every boolean function $f\colon \{\pm 1\}^n \to \{\pm 1\}$ can be represented by a binary neuron network with two layers. Write $f$ in conjunctive normal form and

- for every $w \in \{\pm 1\}^n$ s.t. $f(w) = 1$ introduce a neuron $\operatorname{sign}[w^T x - n + 1]$ in the first layer,
- output layer – one neuron with conjunction of all neurons in the first layer.

Notice: the number of neurons grows exponentially with $n$. Some function classes can be represented in a more efficient way.

**Theorem 1.** *Let $T\colon \mathbb{N} \to \mathbb{N}$ be a growing function and for every $n$ let $\mathcal{F}_n$ be the set of boolean functions implementable by a Turing machine with run time at most $T(n)$. Then there exist constants $a, b > 0$ such that for every $n$ there is a graph $(V, E)$ of size at most $aT^2(n) + b$ such that $\mathcal{H}_{V,E,\operatorname{sign}}$ contains $\mathcal{F}_n$.*

Illustrating example: the parity function on $\{\pm 1\}^n$ can be implemented by a network with $n \log n$ neurons and $\log n$ layers.

**Notation:** Let $(V, E)$ be a directed acyclic graph and let $f$ be a nonlinear activation function. $\mathcal{H}_{V,E,f}$ denotes the class of feed forward networks with architecture $(V, E)$ and activation function $f$.

# Neural networks as universal approximators

**Real valued functions:** consider real valued functions $f\colon [-1,1]^n \to [-1,1]$ that are Lipshitz continuous

$$|f(x) - f(x')| \leqslant \rho \|x - x'\| \quad \forall x, x' \in [-1,1]^n.$$

For every approximation bound $\epsilon > 0$ and every Lipshitz function $f\colon [-1,1]^n \to [-1,1]$, it is possible to construct a network with sigmoid units such that for every input $x \in [-1,1]^n$ it outputs a number between $f(x) - \epsilon$ and $f + \epsilon$. To prove this

- Partition : $[-1,1]^n$ in sufficiently small boxes.
- Approximate $f$ by a function that is piece-wise constant in the boxes
- Design a network that first decides which box the input vector belongs to and then predict the average value of $f$ at this box.

**Theorem 2.** *(Cybenko, 1989) Every smooth function on $[-1,1]^n$ can be approximated arbitrarily well by a network with sigmoid units and two layers. In other words, given a smooth function $f\colon [-1,1]^n \to [-1,1]$ and an $\epsilon > 0$, there is a sum*

$$G(x) = \sum_{j=1}^{N} \alpha_j \, \mathrm{S}(w_j^T x + b_j)$$

*s.t. $|f(x) - G(x)| \leqslant \epsilon$ for all $x \in [-1,1]^n$.*

# Loss functions for Classification

**Hinge loss (multiclass)** Denote the network outputs (class scores) by $z_k$, $k = 1, \ldots K$. The hinge loss for a training example $(x, y)$ is given by

$$\ell(y, h(x)) = \sum_{k \neq y} \max(0, z_k - z_y + 1)$$

**Conditional log-likelihood** If the last layer is softmax with outputs representing class probabilities $p_w(y = k \mid x)$, the objective function for the training set $\mathcal{T}^m = \{(x^j, y^j) \mid j = 1, \ldots, m\}$ is given by

$$L(w) = \frac{1}{m} \sum_{j=1}^{m} \log p_w(y^j \mid x) \to \max_w .$$

Log-likelihood maximisation can be seen as minimisation of the Kullback-Leibler divergence between distributions $q(y)$ and $p(y)$

$$D_{KL}(q \parallel p) = \sum_y q(y) \log \frac{q(y)}{p(y)}$$

here: $q$ - data distribution and $p$ - distribution by modelled by the network

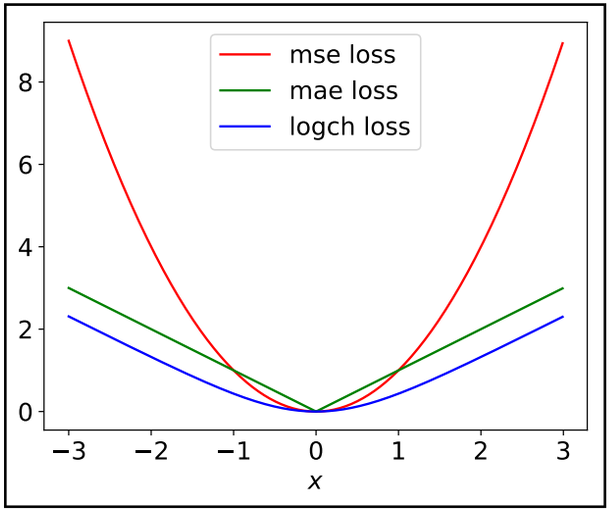
# Loss functions for Regression

The regression function $y = f(x, w)$is modelled by a network with parameters $w$.

**MSE:** $\frac{1}{m}\sum_{j=1}^{m}(y^j - f(x^j, w))^2 \to \min_w$

**MAE:** $\frac{1}{m}\sum_{j=1}^{m}|y^j - f(x^j, w)| \to \min_w$

**logch:** $\frac{1}{m}\sum_{j=1}^{m}\log\cosh(y^j - f(x^j, w)) \to \min_w$




Empirical risk minimisation on predictor classes $\mathcal{H}$ represented by neural networks is, as a rule, hard.

**Theorem 3.** *Let $\mathcal{H}_{V,E,\text{sign}}$ be the set of feed forward networks with $n$ inputs, $k \geqslant 3$ neurons in the first layer and one neuron in the second (output) layer. ERM for this class $\mathcal{H}_{V,E,\text{sign}}$ is NP-hard.*

Methods like back-propagation with stochastic gradient descent (SGD) solve ERM approximately.

# Generalisation bounds for classification

Estimating the VC dimension of classifier networks

**Theorem 4.** *$\mathcal{H}_{V,E,\mathrm{sign}}$ has VC-dimension bounded by $\mathcal{O}(|E|\log|E|)$.*

**Theorem 5.** *$\mathcal{H}_{V,E,\mathrm{S}}$ has VC -dimension $d$ bounded by*

$$\mathcal{O}(|E|^2) < d < \mathcal{O}(|V|^2|E|^2).$$

The proofs of these and similar theorems rely on the concept of **growth function** $\tau(m)$ for a class $\mathcal{H}$ of binary predictors

$$\tau(m) = \max_{C\subset\mathbb{R}^n:\ |C|=m} |\mathcal{H}_C|$$

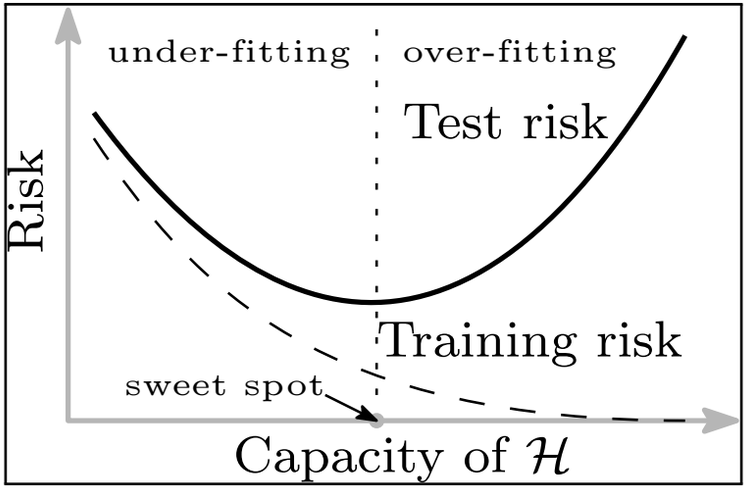
and the bound $|\mathcal{H}_C| \leqslant |\{B \subset C \mid \mathcal{H} \text{ shatters } B\}|$ provided by Sauer's lemma.

**In practice:** we consider networks with weights represented by $\mathcal{O}(1)$ bits (floating point numbers). Hence, their VC-dimension is $\mathcal{O}(|E|)$.

# Generalisation errors

**Question:** are these worst case generalisation bounds useful in deep learning?

We would expect




For large networks with $|E| > 10^6$ parameters the generalisation bound

$$\mathbb{P}\Big(\sup_{h\in\mathcal{H}}|R(h) - R_{\mathcal{T}^m}(h)| > \varepsilon\Big) < 4\Big(\frac{2em}{d}\Big)^d e^{\frac{-m\varepsilon^2}{8}}$$

would require billions of training examples.

These bounds become "vacuous" for realistic training data sizes. The operating point is far to the right in the plot.

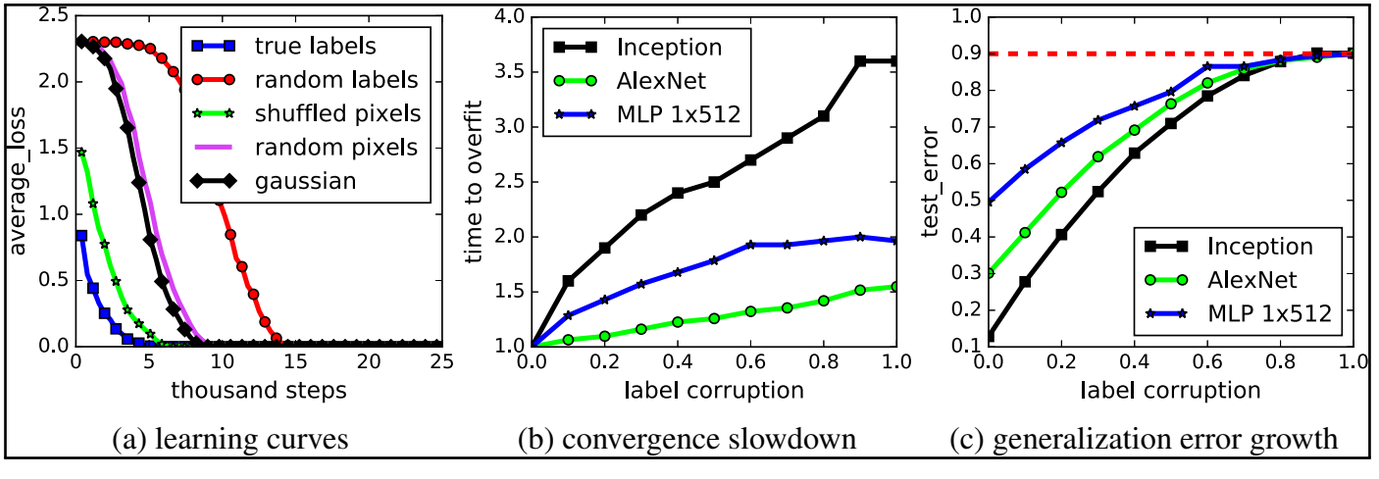
# Generalisation errors

Moreover, often deep networks are so large that they can shatter the training set!

Typical examples:

- image classification on CIFAR (10 classes, $\sim 5 \cdot 10^4$ training examples, tackled by networks with $\sim 10^5$ parameters. Networks learned by SGD and additional regularisers (e.g. data augmentation, droupout, etc.) Achieved accuracy $> 95\%$, generalisation error $< 5\%$.
- image classification on ImageNet (1000 classes, $\sim 10^6$ training examples, tackled by networks with $> 10^6$ parameters. Networks learned by SGD and additional regularisers. Achieved accuracy $\sim 90\%$, generalisation error $< 10\%$.

Training sets can be shattered! Learning with random labels:

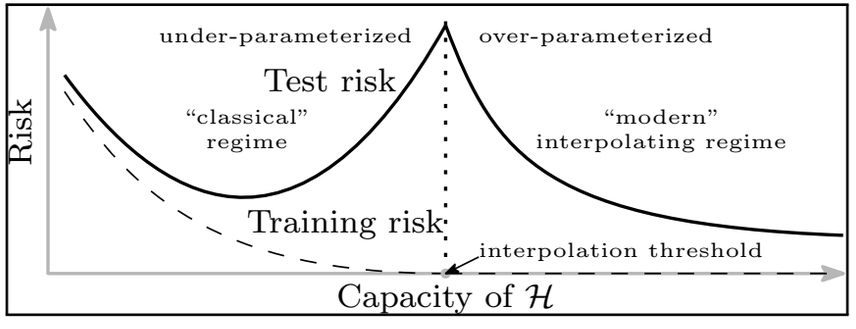


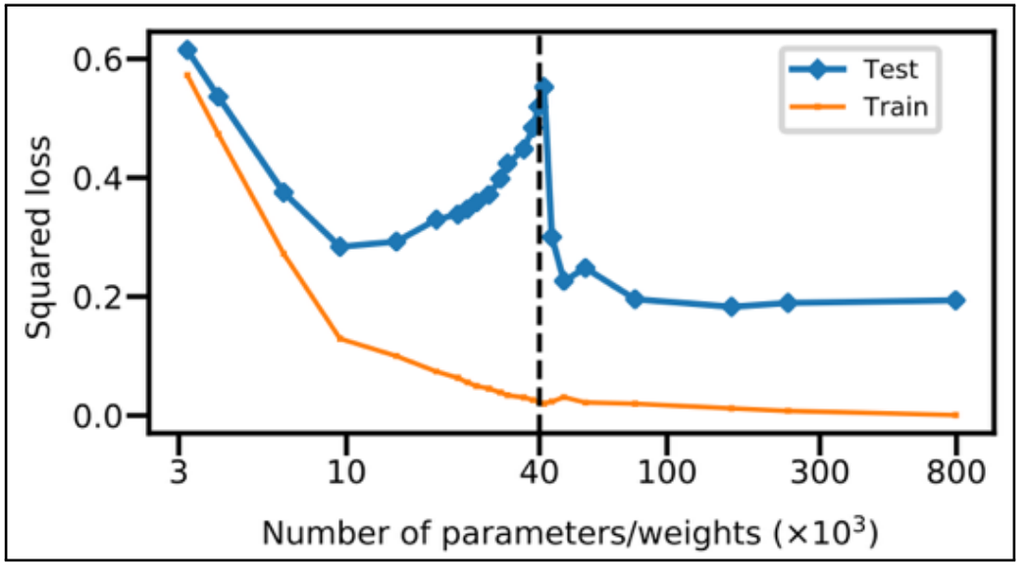

(a) learning curves (b) convergence slowdown (c) generalization error growth

Zhang et al., 2018

# Generalisation errors

Current ongoing research seems to indicate that SGD, when used for training over parametrised networks, is choosing smooth predictors with small norm. For some networks this leads to the following unexpected behaviour.



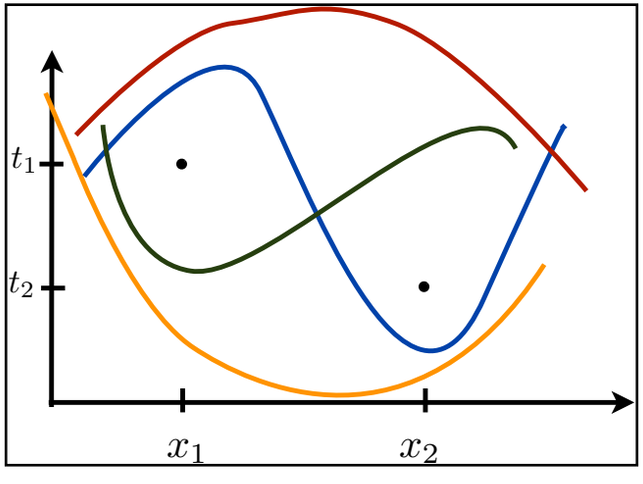




Belkin et al., 2019 right display: network with with a single hidden layer learned on MNIST

# Generalisation bounds for regression

Concepts of shattering, growth function and VC-dimension must be generalised for regressions

- given a finite set $C = \{x_1, \ldots, x_m) \subset \mathcal{X}$ of inputs, consider all tuples $\{(f(x_1), \ldots, f(x_m)) \mid f \in \mathcal{H}\}$. Measure complexity of this set by minimal number of $\epsilon$-balls needed to cover it.
- bound the generalisation error in terms of this "covering function"
- bound the covering function itself by **fat shattering dimension**




Functions in $\mathcal{H}$ shatter the two points $(x_1, t_1)$ and $(x_2, t_2)$