

Neural Networks and Deep Learning

Lecture 1: Deep learning for business: from linear models to the perceptron and the multilayer perceptron (MLP)

Prof. dr. Daniel Traian PELE

Bucharest University of Economic Studies (ASE), Department of Statistics and Econometrics
Romanian Academy, Institute for Economic Forecasting (IPE)
IDA Institute Digital Assets ▪ AI4EFin ▪ MSCA Digital Finance

Learning objectives

- Place deep learning among the tools of a business analyst: where it creates value and where a linear model is enough.
- Write linear and logistic regression as one-neuron networks, with their loss as a likelihood.
- Explain Rosenblatt's perceptron, its learning rule, its convergence theorem and its limit (the exclusive or (XOR) problem).
- Build a multilayer perceptron (MLP): forward pass, activation functions, number of parameters, universal approximation.
- Compare an MLP with logistic regression and gradient boosting for credit scoring on real data, with costs and uncertainty.

Abbreviations

AI	artificial intelligence
AUC	area under the receiver operating characteristic curve
CPU	central processing unit
DM	Deutsche Mark
ES	expected shortfall
GELU	Gaussian error linear unit
GNN	graph neural network
GPU	graphics processing unit
GraphSAGE	graph sample and aggregate (a GNN architecture)
HMEQ	home equity loans dataset
ICBE	International Conference on Business Excellence
IEEE-CIS	Institute of Electrical and Electronics Engineers, Computational Intelligence Society
LLM	large language model
MLP	multilayer perceptron
MSE	mean squared error
PD	probability of default
ReLU	rectified linear unit
ROC	receiver operating characteristic
SGD	stochastic gradient descent
SHAP	Shapley additive explanations
UCI	University of California, Irvine (Machine Learning Repository)
USD	US dollar
VaR	value at risk
VLM	vision-language model
XAI	explainable artificial intelligence
XGBoost	extreme gradient boosting
XOR	exclusive or

Outline

- Deep learning for business
- Linear models are one-neuron networks
- The perceptron
- The multilayer perceptron
- Case study: credit scoring with real loans
- Wrap-up

Deep learning learns the features that analysts used to build by hand

- **In one sentence:** deep learning, the branch of artificial intelligence (AI) behind today's image, text and speech systems, fits a chain of simple functions to data.

$$f(\mathbf{x};\boldsymbol{\theta}) = f^{(L)}\big(f^{(L-1)}(\cdots f^{(1)}(\mathbf{x}))\big), \qquad \hat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}} \frac{1}{n}\sum_{i=1}^{n} \ell\big(y_i, f(\mathbf{x}_i;\boldsymbol{\theta})\big)$$

- **where**
  - $\mathbf{x}$ the input (applicant, transaction, text), $f^{(l)}$ layer l of L, $\boldsymbol{\theta}$ all weights
  - ℓ the loss, n the sample size
- Why now: data, hardware and software arrived together
  - Large labelled datasets, graphics processing units (GPU) and automatic differentiation in PyTorch (Paszke et al., 2019).
- What changed: representation learning
  - The network builds its own features from raw inputs (LeCun, Bengio & Hinton, 2015).

- **In practice:** ask what the network would learn that a scorecard analyst cannot write down by hand.

(Bishop & Bishop, 2024, ch. 1, 6)

Where deep learning creates value in business and finance

Domain	Decision	Network	Course data
Credit	accept or reject a loan; price the risk	MLP (Lectures 1, 3)	German credit, home equity (HMEQ)
Payments	block or let a card transaction pass	MLP, autoencoder	card fraud (Lectures 2, 3, 8)
Markets	size a position; set a risk limit	recurrent, transformer	indices, Bucharest shares
Energy	bid on the day-ahead market	recurrent, foundation models	Romanian power prices
Text	read news, filings, central-bank speeches	embeddings, transformers	PhraseBank, tweets, speeches
Networks	flag money laundering in a payment graph	graph networks	Elliptic Bitcoin

- The course follows this table: Lectures 1–3 tabular data, 4 images, 5 and 13 time series, 6, 7 and 12 text, 9 graphs, 10 decisions, 11 explanations.

- **In practice:** start from the decision and its cost, then pick the network that fits the data type.

For small tabular data, a linear model or boosting is often enough

- **In one sentence:** deep learning wins on raw, high-dimensional inputs; on a few dozen engineered columns it must earn its place.

- Credit scoring benchmark: across 41 classifiers on 8 credit datasets, logistic regression stays a strong reference
  - Ensembles lead (Lessmann et al., 2015).
- Tabular benchmark: tree ensembles still beat deep networks on typical medium-sized tabular data (Grinsztajn, Oyallon & Varoquaux, 2022).
- Regulation: a credit decision must be explained to the applicant.

- Use a network when: inputs are images, text, sequences or graphs
  - Data are large
  - The relation is strongly nonlinear.
- Keep the benchmark when: a few thousand rows, engineered features, a need for a scorecard.
- This lecture tests both cases on real loans, with a fixed protocol.

- **In practice:** every network in this course is reported next to a linear or boosting benchmark.

The data: three real credit datasets

Dataset	Rows	Events	Source
German credit	1 000	300 bad risks	University of California, Irvine (UCI) Machine Learning Repository (Hofmann, 1994)
Home equity loans (HMEQ)	5 960	1 189 defaults	Baesens, Rösch & Scheule (2016)
California housing	20 640	(regression)	1990 census block groups (Pace & Barry, 1997)

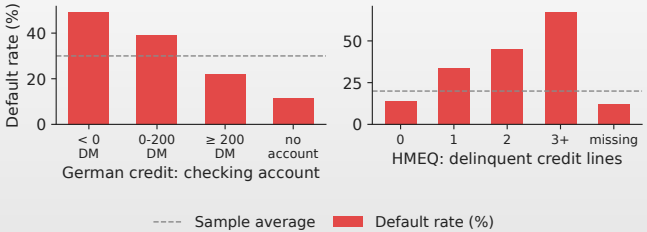
- German credit: 20 attributes, amounts in Deutsche Mark (DM); default rate 30.0 %.
- HMEQ: 12 variables, default rate 19.9 %; missing values are kept and modelled.
- California: median house value in 100,000 US dollars (USD), capped by the source at 5.

- **In practice:** check the source, the unit of observation and the event rate before any model.

Experiment yourself: notebook §1

Default rates differ strongly across simple applicant groups

- **In one sentence:** before any model, raw default rates already show which variables carry risk.



Source: UCI Statlog German credit; Baesens, Roesch and Scheule (2016), HMEQ. Data: 1,000 applicants (German credit); 5,960 loans (HMEQ).

- Probability of default (PD): 49.3 % with a negative checking balance, 11.7 % with no account
  - 14.0 % with no delinquent line, 67.3 % with three or more.

- **In practice:** a model must at least reproduce these group rates on the validation set.

(own computation)

NN_L1_default_rates

Learning as minimisation of the average training loss, evaluated on new data

- **In one sentence:** we fit on one sample and pay for the errors on the next applicants.

$$\underbrace{\hat{R}_n(\theta) = \frac{1}{n}\sum_{i=1}^{n} \ell\big(y_i, f(\mathbf{x}_i;\theta)\big)}_{\text{empirical risk (training)}} \qquad \text{vs.} \qquad \underbrace{R(\theta) = \mathbb{E}_{(\mathbf{x},y)\sim P}\, \ell\big(y, f(\mathbf{x};\theta)\big)}_{\text{risk on new applicants}}$$

- **where**
  - P the unknown distribution of future applicants, ℓ the loss, $\hat{R}_n$ is minimised, R is what the bank pays
- Three sets: train (fit θ), validation (choose the model), test (report once).
- HMEQ, stratified: 3 576 train, 1 192 validation, 1 192 test loans.
- No leakage: medians and scaling are fitted on the training part only.

- **In practice:** the test set is used once, at the end (validation design: Lecture 3).

(Bishop & Bishop, 2024, ch. 1)

Experiment yourself: notebook §2

Outline

- Deep learning for business
- Linear models are one-neuron networks
- The perceptron
- The multilayer perceptron
- Case study: credit scoring with real loans
- Wrap-up

Linear regression is one neuron with no activation

- **In one sentence:** a weighted sum of the inputs plus a bias; the squared loss is the Gaussian log-likelihood.

$$\hat{y} = \mathbf{w}^\top \mathbf{x} + b, \qquad \mathrm{MSE}(\mathbf{w}, b) = \frac{1}{n} \sum_{i=1}^{n} \left(y_i - \mathbf{w}^\top \mathbf{x}_i - b \right)^2$$

- **where**
  - $\mathbf{x} \in \mathbb{R}^d$ the inputs, $\mathbf{w}$ the weights, b the bias
  - Mean squared error (MSE) the loss

$$\varepsilon_i \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0, \sigma^2): \quad -\log L = \tfrac{n}{2} \log 2\pi\sigma^2 + \tfrac{n}{2\sigma^2} \mathrm{MSE}$$

- Maximum likelihood = least squares: the same $\hat{\mathbf{w}} = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y}$.
- A network has no closed form: it is trained by gradient descent (Lecture 2).
- California housing, latitude only: held-out MSE 0.896 for the line.

- **In practice:** a linear fit is the benchmark every regression network must beat out of sample.

(Bishop & Bishop, 2024, ch. 4)

Logistic regression is one neuron with a sigmoid

- **In one sentence:** the neuron outputs a probability, and the loss is the Bernoulli negative log-likelihood (cross-entropy).

$$p_i = \sigma(\mathbf{w}^\top \mathbf{x}_i + b) = \frac{1}{1 + e^{-z_i}}, \qquad \nabla_{\mathbf{w}} \mathcal{L} = \frac{1}{n} \sum_{i=1}^{n} (p_i - y_i)\, \mathbf{x}_i$$

$$\mathcal{L}(\mathbf{w}, b) = -\frac{1}{n} \sum_{i=1}^{n} \Big[y_i \log p_i + (1 - y_i) \log(1 - p_i) \Big]$$

- The gradient is error times input: the same form will appear in every layer of backpropagation.
- e^{w_j} is the odds ratio for a one-unit change of x_j.
- Convex loss: one global minimum.

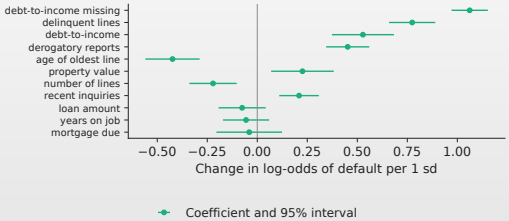
- **where**
  - z_i the logit (log-odds), σ the sigmoid, $y_i \in \{0, 1\}$ default
  - Derivation in the appendix

- **In practice:** the logistic score is the industry standard for PD models.

(Bishop & Bishop, 2024, ch. 5)

Delinquencies and a missing debt ratio raise the odds of default most

- **In one sentence:** HMEQ, standardised inputs: each coefficient is the change in log-odds per one standard deviation.



Source: Baesens, Roesch and Scheule (2016), HMEQ data. Data: 3,576 training loans.

- Odds ratios e^{w_j}: 2.89 for a missing debt ratio, 2.17 for delinquent lines, 0.65 for an older credit history (McFadden pseudo-R^2 0.374).

- **In practice:** a sign against credit sense (property value) is a reason to inspect the data.

(own computation) NN_L1_logit_coef

The forward pass of one real applicant, step by step

⊡ **In one sentence:** the score is a sum of contributions $w_j x_j$; the sigmoid turns it into a probability.

Input	x_j	w_j	$w_j x_j$
delinquent credit lines	1.46	0.774	+1.13
number of credit lines	2.81	−0.221	−0.62
debt-to-income missing	−0.51	1.062	−0.54
bias b			−1.99
8 other inputs			+0.31
score z			**−1.71**

- ⊡ Applicant from the test set who later defaulted: loan 15 400 USD, debt-to-income 36.4 %, 2 delinquent lines.
- ⊡ Output: $p = \sigma(z) = 15.3\,\%$, below the 50 % default cut-off.
- ⊡ Inputs x_j are standardised: $x_j = (\text{raw}_j - \bar{x}_j)/s_j$ with training means and standard deviations.

⊡ **In practice:** the cut-off is a business choice: Section 5 sets it from the cost of a missed default.

(own computation)

Experiment yourself: notebook §2

Missing values as predictors of default

- **In one sentence:** applicants who do not report their debt ratio default far more often: missingness is information.

$$\tilde{x}_{ij} = \begin{cases} x_{ij} & \text{observed} \\ \mathrm{med}_{\text{train}}(x_j) & \text{missing} \end{cases}, \qquad m_{ij} = \mathbb{1}\{x_{ij} \text{ missing}\}$$

- Debt-to-income is missing for 21.3 % of HMEQ loans.
- Default rate 62.0 % when it is missing against 8.6 % when it is reported.
- Each of the 10 numeric inputs gets a median fill and an indicator m_{ij}
  - With 8 dummies for reason and job, the network sees $d = 28$ inputs.

- **In practice:** never drop rows with missing values in credit data: you would drop the riskiest applicants.

(Baesens, Rösch & Scheule, 2016)

Experiment yourself: notebook §2

Outline

- Deep learning for business
- Linear models are one-neuron networks
- The perceptron
- The multilayer perceptron
- Case study: credit scoring with real loans
- Wrap-up

From the McCulloch–Pitts neuron to the perceptron

- **In one sentence:** McCulloch and Pitts wrote the neuron as a threshold unit; Rosenblatt made its weights learn from examples.

$$\hat{y} = \text{step}(\mathbf{w}^\top \mathbf{x} + b) = \begin{cases} +1 & \mathbf{w}^\top \mathbf{x} + b > 0 \\ -1 & \text{otherwise} \end{cases}$$

- **where**
  - $\mathbf{x}$ the inputs (dendrites), $\mathbf{w}$ the synaptic weights, $-b$ the firing threshold
  - $\hat{y}$ the class (axon)

- 1943: logical calculus of threshold neurons (McCulloch & Pitts, 1943).
- 1958: the perceptron learns $\mathbf{w}$ from labelled examples (Rosenblatt, 1958).
- 1969: a single layer cannot represent XOR (Minsky & Papert, 1969).
- 1986: backpropagation trains several layers (Rumelhart, Hinton & Williams, 1986).

- **In practice:** the perceptron is a linear classifier: its boundary $\mathbf{w}^\top \mathbf{x} + b = 0$ is a hyperplane.

(Bishop & Bishop, 2024, ch. 5)

The perceptron updates its weights only on misclassified examples

- **In one sentence:** when an example is misclassified, the weights move towards it; otherwise nothing changes.

$$\text{if } y_i(\mathbf{w}^\top \mathbf{x}_i + b) \le 0: \qquad \mathbf{w} \leftarrow \mathbf{w} + \eta\, y_i \mathbf{x}_i, \quad b \leftarrow b + \eta\, y_i$$

$$\mathcal{L}_P(\mathbf{w}, b) = -\sum_{i \in \mathcal{M}} y_i(\mathbf{w}^\top \mathbf{x}_i + b)$$

- **where**
  - $y_i \in \{-1, +1\}$, $\eta > 0$ the learning rate, $\mathcal{M}$ the set of misclassified examples
  - $\mathcal{L}_P$ the perceptron criterion

- The update is one step of stochastic gradient descent (SGD) on $\mathcal{L}_P$ for one misclassified example.
- Unlike logistic regression, correct examples give no gradient: the perceptron ignores how confident it is.
- No probability comes out: only a class.

- **In practice:** for credit, a class without a probability cannot be priced or set against a cost.

(Rosenblatt, 1958; Bishop & Bishop, 2024, ch. 5)

If the classes can be separated, the perceptron stops after finitely many mistakes

- **In one sentence:** a wide margin between the classes means few corrections are needed.

$$\exists \mathbf{w}^*,\ \|\mathbf{w}^*\| = 1 :\ y_i\, \mathbf{w}^{*\top} \mathbf{x}_i \geq \gamma > 0 \ \ \forall i, \quad \|\mathbf{x}_i\| \leq R \implies \#\text{mistakes} \leq \left(\frac{R}{\gamma}\right)^2$$

- **where**
  - γ the margin of the best separating hyperplane, R the radius of the data, the bias is absorbed in $\mathbf{x}$ (a constant 1)
- Novikoff's theorem (1962): the bound does not depend on the number of examples or on the dimension.
- Proof idea: $\mathbf{w}^\top \mathbf{w}^*$ grows by at least γ per mistake, $\|\mathbf{w}\|^2$ by at most R^2 (appendix).
- If no separating hyperplane exists, the theorem says nothing and the weights cycle for ever.

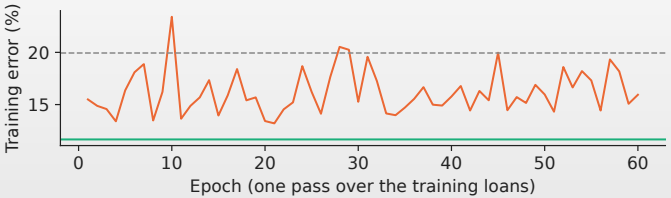
- **In practice:** real default data are never separable: applicants with the same file can repay or default.

(Novikoff, 1962)

Experiment yourself: notebook §3

The perceptron does not converge on non-separable credit data

- **In one sentence:** the classes overlap, so every epoch corrects some loans and breaks others.



Source: Baesens, Roesch and Scheule (2016), HMEQ data. Data: 3,576 training loans.

- Error after each of 60 epochs: 13.2–23.4 %, against 11.7 % for logistic regression and 19.9 % for 'always no default'.

- **In practice:** a smooth loss and a probability output fix this: networks use sigmoids, not steps.

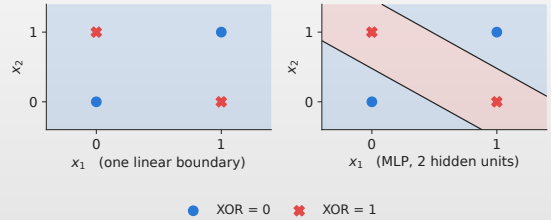
(own computation)

NN_L1_perceptron_errors

The XOR problem and the hidden layer

- **In one sentence:** XOR is true when exactly one input is on; the two positive points sit on a diagonal.

$$h_k = \tanh(\mathbf{w}_k^\top \mathbf{x} + b_k), \; k = 1, 2; \qquad p = \sigma(v_1 h_1 + v_2 h_2 + c)$$



Source: logical XOR table (no data). Data: the four XOR inputs.

- Accuracy on the four points: 50 % for any linear boundary, 100 % with two hidden units.
- **In practice:** interactions such as 'high debt and short history' are XOR-like.

(Minsky & Papert, 1969) **SIMULATED DATA (teaching illustration)** NN_L1_xor

Outline

- Deep learning for business
- Linear models are one-neuron networks
- The perceptron
- The multilayer perceptron
- Case study: credit scoring with real loans
- Wrap-up

An MLP alternates affine maps and nonlinear activations

- **In one sentence:** each layer computes new features from the previous ones; the last layer is a logistic regression on them.

$$\mathbf{h}^{(0)} = \mathbf{x}, \quad \mathbf{h}^{(l)} = a\big(\mathbf{W}^{(l)}\mathbf{h}^{(l-1)} + \mathbf{b}^{(l)}\big), \ l = 1, \dots, L-1, \quad p(\mathbf{x}) = \sigma\big(\mathbf{w}^{(L)\top}\mathbf{h}^{(L-1)} + b^{(L)}\big)$$

- **where**
  - $\mathbf{W}^{(l)} \in \mathbb{R}^{H_l \times H_{l-1}}$, $\mathbf{b}^{(l)}$ weights and biases of layer l, of width H_l ($H_0 = d$), a the activation, element by element
- Without a, the layers collapse: $\mathbf{W}^{(2)}\mathbf{W}^{(1)}\mathbf{x}$ is again linear (appendix).
- Our credit network: $d = 28$ inputs, two hidden layers of 64 rectified linear units (ReLU), one sigmoid output ('MLP 64-64').

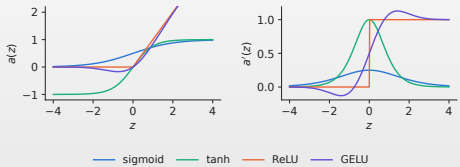
- **In practice:** width and depth are design choices
  - Section 5 chooses them on the validation set.

(Bishop & Bishop, 2024, ch. 6)

Experiment yourself: notebook §4

Activation functions and their properties

$$\sigma(z) = \frac{1}{1+e^{-z}}, \quad \tanh(z) = \frac{e^{z}-e^{-z}}{e^{z}+e^{-z}}, \quad \mathrm{ReLU}(z) = \max(0, z), \quad \mathrm{GELU}(z) = z\,\Phi(z)$$



Source: analytic functions (no data). Data: functions plotted on [-4, 4].

- Sigmoid and tanh saturate ($\sigma' \leq 0.25$)
  - The rectified linear unit (ReLU) passes gradient 1 for $z > 0$
  - The Gaussian error linear unit (GELU, Φ the normal distribution function) is used in transformers.

- **In practice:** use ReLU in hidden layers
  - The output activation follows the target.

(Bishop & Bishop, 2024, ch. 6) **SIMULATED DATA (teaching illustration)** NN_L1_activations

The output layer and the loss come from the type of the target

- **In one sentence:** each pair (output activation, loss) is a maximum-likelihood estimator for one distribution of y.

Target	Output	Loss	Example
real number	$\hat{y} = z$	MSE: $\frac{1}{n}\sum(y_i - \hat{y}_i)^2$	house value, a price
binary	$p = \sigma(z)$	cross-entropy: $-\frac{1}{n}\sum[y \log p + (1 - y)\log(1 - p)]$	default, fraud
K classes	$p_k = \frac{e^{z_k}}{\sum_j e^{z_j}}$	$-\frac{1}{n}\sum_i \log p_{i,y_i}$	rating grade, news sentiment
quantile τ	$\hat{q} = z$	pinball: $\frac{1}{n}\sum \rho_\tau(y_i - \hat{q}_i)$	value at risk (VaR), Lecture 2

- $\rho_\tau(u) = u(\tau - \mathbb{1}\{u < 0\})$; softmax with $K = 2$ reduces to the sigmoid.

- **In practice:** a PD model needs a sigmoid output and cross-entropy, so that its scores are probabilities.

(Bishop & Bishop, 2024, ch. 5–6)

A small MLP already has hundreds of times more parameters than the logistic score

- **In one sentence:** every connection is a weight, and every unit has a bias.

$$|\theta| = \sum_{l=1}^{L} (H_{l-1} + 1)\, H_l, \qquad H_0 = d,\ H_L = 1$$

- Logistic regression on HMEQ: $(d+1)\cdot 1 = 29$ parameters.
- MLP 64-64: $(d+1)\cdot 64 + 65\cdot 64 + 65\cdot 1 = 6\,081$ parameters for 3 576 training loans.
- More parameters than training defaults: without validation-based stopping the network would memorise them (Lecture 3).
- Here every MLP is trained with Adam (Lecture 2) and keeps the weights of its best validation epoch.

- **In practice:** count parameters before training: it tells you how much data and regularisation you will need.

Experiment yourself: notebook §4

One hidden layer can approximate any continuous function

- **In one sentence:** enough sigmoid 'bumps', added together, can trace any smooth curve on a bounded domain.

$$\forall\, g \in C(K),\ \forall\, \varepsilon > 0\ \ \exists\, H,\ v_k, \mathbf{w}_k, b_k :\quad \sup_{\mathbf{x}\in K} \left| g(\mathbf{x}) - \sum_{k=1}^{H} v_k\, \sigma\!\left(\mathbf{w}_k^\top \mathbf{x} + b_k\right) \right| < \varepsilon$$

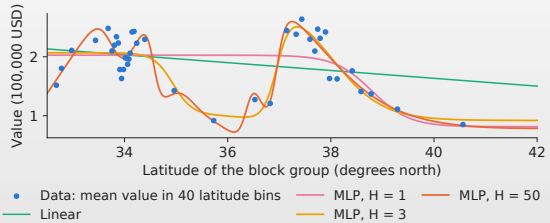
- **where**
  - $K \subset \mathbb{R}^d$ compact, g continuous, σ sigmoidal (Cybenko, 1989) or non-polynomial (Hornik, 1991)
- What it says: the MLP family is rich enough.
- What it does not say: how large H must be, whether gradient descent finds the weights, or whether they generalise.

- **In practice:** the theorem justifies trying an MLP
  - Only out-of-sample tests justify keeping it.

(Cybenko, 1989; Hornik, 1991)

More hidden units trace the price geography of California

- **In one sentence:** values peak near Los Angeles and San Francisco; $\hat{y}(x) = c + \sum_{k=1}^{H} v_k \sigma(w_k x + b_k)$ with $H = 1, 3, 50$.



Source: StatLib, California housing, 1990 census (Pace and Barry, 1997). Data: 19,648 block groups below the source cap.

- Held-out MSE: linear 0.896, $H = 1$ 0.855, $H = 3$ 0.757, $H = 50$ 0.731 (992 block groups at the 500,000 USD cap left out).

- **In practice:** location effects are nonlinear: a linear term in latitude cannot price them.

(own computation)

NN_L1_uat_california

Depth and width of a network

- **In one sentence:** each layer composes the features of the previous one, so the number of linear pieces can grow with depth.

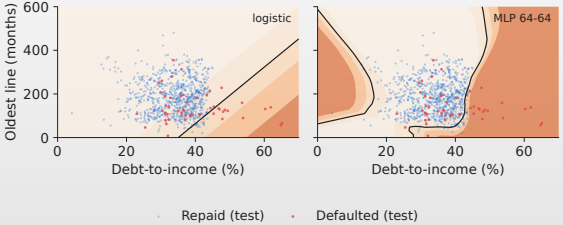
$$\text{ReLU MLP: } f(\mathbf{x}) \text{ is piecewise linear; number of linear regions up to } O\left(H^{dL}\right)$$

- **where**
  - H the width of each layer, L the number of hidden layers, d the input dimension (bound of Montúfar et al., 2014)
- For some functions a shallow net needs exponentially more units than a deep one.
- On tabular credit data, two or three hidden layers are usually enough
  - Depth pays off on images, text and sequences.
- **In practice:** start with 2 hidden layers on tabular data
  - Go deeper only if validation loss keeps falling.

(Bishop & Bishop, 2024, ch. 6)

Nonlinear decision boundary of an MLP with two inputs

- **In one sentence:** predicted PD for two HMEQ variables: straight bands for the logistic model, curved regions for the MLP.



Source: Baesens, Roesch and Scheule (2016), HMEQ data. Data: 2,708 training and 871 test loans with both variables.

- Area under the receiver operating characteristic (ROC) curve (AUC) $= \Pr(p_{\text{default}} > p_{\text{good}})$
  - With these two inputs: logistic 0.744, MLP 64-64 0.736.

- **In practice:** a flexible boundary pays only if the inputs carry the signal.

(own computation)

NN_L1_boundary_hmeq

Outline

- Deep learning for business
- Linear models are one-neuron networks
- The perceptron
- The multilayer perceptron
- Case study: credit scoring with real loans
- Wrap-up

Experimental protocol: a common split and common inputs for three models

- **In one sentence:** a fair comparison changes only the model; data, split and metric stay fixed before anything is trained.

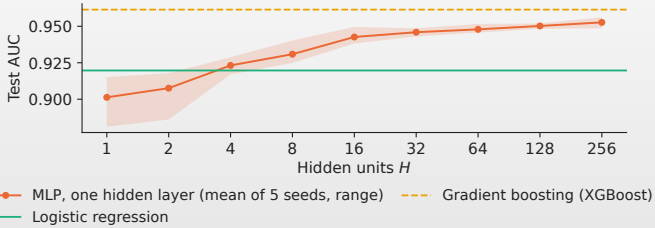
- Logistic regression: maximum likelihood, no penalty on HMEQ
  - A mild ridge penalty ($C = 1$) on German credit.
- MLP: ReLU layers, Adam with learning rate 10^{-3}, batches of 128 (64 on German credit), best validation epoch, 5 seeds.
- Extreme gradient boosting (XGBoost): depth 4, learning rate 0.03, early stopping on validation (Chen & Guestrin, 2016).
- HMEQ: one stratified 60/20/20 split
  - AUC, ROC curve, calibration, bootstrap interval of the AUC gain.
- German credit: 5 times repeated 5-fold cross-validation (25 test folds), corrected t-test.
- Seed 20261108 (+ run index)
  - Trained on the central processing unit (CPU), under a minute per model family.

- **In practice:** write the protocol down before training: it is what a model validator will ask for first.

Experiment yourself: notebook §5

On HMEQ, a wider hidden layer raises the test AUC above the logistic score

⊡ **In one sentence:** one hidden layer of width H; each point is the mean of 5 seeds, the band their range.



Source: Baesens, Roesch and Scheule (2016), HMEQ data. Data: 3,576 training, 1,192 validation, 1,192 test loans.

⊡ AUC 0.901 at $H = 1$, 0.953 at $H = 256$
- Logistic 0.920, XGBoost 0.961
- The gain flattens after about 16 units.

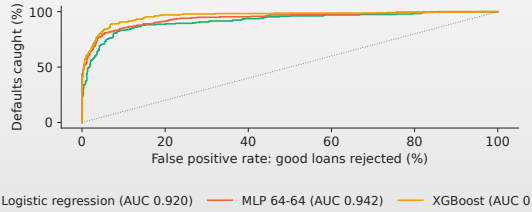
⊡ **In practice:** interactions matter here: the network beats the linear score, boosting beats both.

(own computation)

NN_L1_width_auc

At a 10 % false-alarm rate, the MLP catches more defaults than the logistic score

- **In one sentence:** the ROC curve plots $\Pr(p \geq c \mid \text{default})$ against $\Pr(p \geq c \mid \text{good})$ for every cut-off c.



Source: Baesens, Roesch and Scheule (2016), HMEQ data. Data: 1,192 test loans (238 defaults).

- Defaults caught when 10 % of good loans are rejected: logistic 83.2 %, MLP 85.3 %, XGBoost 90.8 % (238 test defaults).

- **In practice:** read the curve at the rejection rate the business accepts, not only its area.

(own computation) NN_L1_roc_hmeq

The MLP gain on HMEQ is larger than seed noise and test-set noise

- **In one sentence:** one run proves nothing: we repeat the training and resample the test set.

$$\Delta = \mathrm{AUC}_{\mathrm{MLP}} - \mathrm{AUC}_{\mathrm{logit}}, \qquad \Delta_b^* = \mathrm{AUC}_{\mathrm{MLP}}(\mathcal{T}_b^*) - \mathrm{AUC}_{\mathrm{logit}}(\mathcal{T}_b^*), \; b = 1, \ldots, 2000$$

- **where**
  - $\mathcal{T}_b^*$ a bootstrap resample of the 1 192 test loans, the same resample for both models (paired)
- Seeds: MLP 64-64 test AUC 0.945 on average over 5 seeds (standard deviation 0.004; range 0.942–0.951).
- Test noise: $\Delta = 0.023$, 95 % bootstrap interval [0.008; 0.037]: zero is excluded.
- The seed spread of the logistic model is zero: its loss is convex.

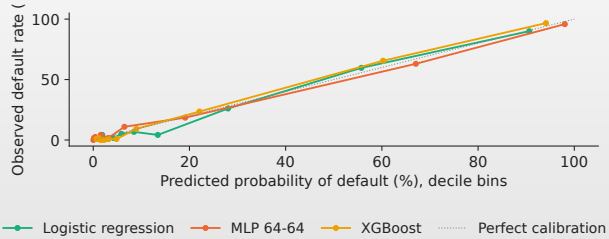
- **In practice:** report a model gain with an interval
  - A gain inside the seed spread is not a gain.

(own computation)

Experiment yourself: notebook §5

The scores are close to calibrated probabilities

- **In one sentence:** in each decile of predicted PD, the observed default rate should equal the predicted one.



Source: Baesens, Roesch and Scheule (2016), HMEQ data. Data: 1,192 test loans.

- Brier score $\frac{1}{n}\sum(p_i - y_i)^2$: logistic 0.0742, MLP 0.0625, XGBoost 0.0549
  - Constant rate 0.1598.

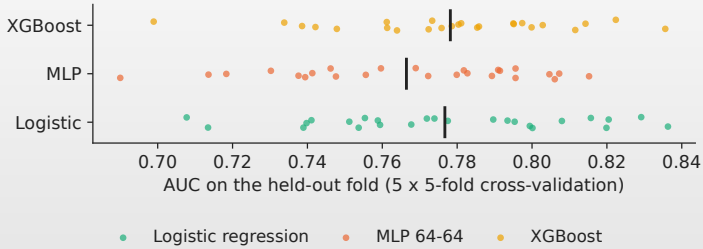
- **In practice:** provisions and prices use the PD itself: check calibration, not only ranking.

(own computation)

NN_L1_calibration

On 1,000 German applicants, the MLP does not beat logistic regression

⊡ **In one sentence:** each dot is the AUC of one held-out fold; the black bar is the mean of the 25 folds.



Source: UCI Machine Learning Repository, Statlog German credit. Data: 1,000 applicants, 25 held-out folds.

⊡ Mean AUC: logistic 0.777, MLP 0.766, XGBoost 0.778
- MLP minus logistic: $t = -0.93$, $p = 0.362$ (next frame).

⊡ **In practice:** with 1,000 rows, keep the logistic scorecard: the network adds risk, not accuracy.

(own computation)

NN_L1_german_cv

Why the usual t-test overstates differences in cross-validation

- **In one sentence:** the training sets of different folds overlap, so the fold scores are positively correlated.

$$t = \frac{\bar{d}}{\sqrt{\left(\frac{1}{J} + \frac{n_{\text{test}}}{n_{\text{train}}}\right) s_d^2}} \sim t_{J-1}, \qquad d_j = \text{AUC}_j^{\text{MLP}} - \text{AUC}_j^{\text{logit}}$$

- **where**
  - $J = 25$ folds, s_d^2 the sample variance of d_j, $n_{\text{test}}/n_{\text{train}} = 1/4$ for 5 folds
- The term $n_{\text{test}}/n_{\text{train}}$ is the Nadeau–Bengio correction: without it, t is far too large.
- MLP vs logistic: $t = -0.93$ ($p = 0.362$)
  - XGBoost vs logistic: $t = 0.12$ ($p = 0.909$).

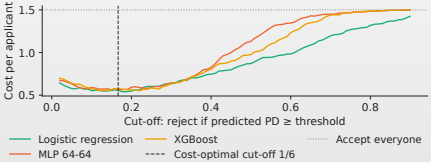
- **In practice:** a model change in a bank needs evidence that survives this correction.

(Nadeau & Bengio, 2003)

Experiment yourself: notebook §6

Cost-sensitive cut-off of 1/6 when a missed default costs five times a rejected good applicant

$$\text{reject if } \; c_{\mathrm{FN}}\, p \;\geq\; c_{\mathrm{FP}}(1-p) \iff p \;\geq\; \frac{c_{\mathrm{FP}}}{c_{\mathrm{FP}} + c_{\mathrm{FN}}} = \frac{1}{6}$$



Source: UCI Machine Learning Repository, Statlog German credit. Data: 1,000 applicants, out-of-fold predictions.

- Cost per applicant (UCI cost matrix 5:1): at 1/6 logistic 0.554, MLP 0.568
  - At 1/2 0.870 and 1.130
  - Accept all 1.50.

- **In practice:** here the choice of cut-off matters more than the choice of model.

(own computation)

NN_L1_cost_threshold

A minimal MLP implementation in PyTorch

```
import torch, torch.nn as nn
model = nn.Sequential(
    nn.Linear(d, 64), nn.ReLU(),
    nn.Linear(64, 64), nn.ReLU(),
    nn.Linear(64, 1))            # logit z
loss_fn = nn.BCEWithLogitsLoss()  # sigmoid + cross-entropy
opt = torch.optim.Adam(model.parameters(), lr=1e-3)
for epoch in range(300):
    for xb, yb in train_batches:
        opt.zero_grad()
        loss = loss_fn(model(xb), yb)
        loss.backward()           # backpropagation
        opt.step()                # Adam update
    # evaluate on validation, keep the best weights
p = torch.sigmoid(model(x_test))   # predicted PD
```

- `BCEWithLogitsLoss` combines the sigmoid and the cross-entropy in one numerically stable function.
- `backward()` computes all gradients by backpropagation
  - `step()` applies the optimiser: both are Lecture 2.
- The notebook adds the data pipeline, the validation stopping and the seeds.

- **In practice:** the same fifteen lines train every tabular network of this course
  - Only data and layers change.

(Paszke et al., 2019)

Experiment yourself: notebook §7

When to use an MLP for tabular business data

Evidence from this lecture

- HMEQ, 3 576 loans: MLP AUC 0.945 vs logistic 0.920
  - XGBoost 0.961.
- German credit, 1,000 applicants: no significant difference ($p = 0.362$).
- Both: the cut-off from the cost matrix matters more than the model.

Rule of thumb

- Few rows, engineered features, explanation required: logistic scorecard.
- Thousands of rows, interactions: test boosting and an MLP against it.
- Raw text, images, sequences, graphs, or a network shared across tasks: deep learning.

- **In practice:** choose the simplest model whose out-of-sample gain survives seeds, resampling and the cost matrix.

Outline

- Deep learning for business
- Linear models are one-neuron networks
- The perceptron
- The multilayer perceptron
- Case study: credit scoring with real loans
- Wrap-up

Key takeaways

- Linear and logistic regression are one-neuron networks
  - Their losses are likelihoods, their gradients are error times input.
- The perceptron converges only on separable data
  - Real credit data are not separable, and XOR shows the limit of one layer.
- An MLP stacks affine maps and activations
  - One hidden layer approximates any continuous function, depth makes it efficient.
- On real loans: the MLP beats logistic regression on HMEQ (AUC 0.945 vs 0.920), not on German credit
  - Boosting is the bar to clear.
- Evaluate with seeds, intervals, calibration and the cost of errors, never with one accuracy number.

Key equations

$$p = \sigma(\mathbf{w}^\top \mathbf{x} + b)$$

$$\mathcal{L} = -\tfrac{1}{n} \sum [y \log p + (1-y) \log(1-p)]$$

$$\nabla_{\mathbf{w}} \mathcal{L} = \tfrac{1}{n} \sum (p_i - y_i) \mathbf{x}_i$$

$$\mathbf{w} \leftarrow \mathbf{w} + \eta\, y_i \mathbf{x}_i \quad \text{if misclassified}$$

$$\mathbf{h}^{(l)} = a\big(\mathbf{W}^{(l)} \mathbf{h}^{(l-1)} + \mathbf{b}^{(l)}\big)$$

$$|\boldsymbol{\theta}| = \textstyle\sum_l (H_{l-1} + 1) H_l$$

$$\#\text{mistakes} \le (R/\gamma)^2$$

$$\text{reject if } p \ge \frac{c_{\mathrm{FP}}}{c_{\mathrm{FP}} + c_{\mathrm{FN}}}$$

Research from our group: deep learning for business decisions

- **Explainable fraud detection** (Diaconu & Pele, 2026): compared XGBoost and CatBoost with a graph neural network (GNN), graph sample and aggregate (GraphSAGE), on the Institute of Electrical and Electronics Engineers, Computational Intelligence Society (IEEE-CIS) fraud data
  - Boosting had the higher AUC, the graph model captured relational fraud patterns
  - A multi-level explainable artificial intelligence (XAI) framework joins Shapley additive explanations (SHAP) with node and edge ablation — replicated in Case study 1 (course site).
- **Early warning of crypto 'zombies'** (Będowska-Sójka, Wójcik & Pele, 2026): machine-learning early warning of crypto-assets that become 'zombie' assets — Lecture 3.
- **Risk measures from language models** (Pele et al., 2026): value at risk (VaR) and expected shortfall (ES) obtained by prompting large language models (LLM) — Lecture 12.
- **Foundation models for tail risk** (Jheng, Tak, Găman, Mazurencu-Marinescu-Pele & Pele, 2026): zero-shot VaR and ES with conformal calibration in Central and Eastern European markets — Lecture 13.
- **Seeing market anomalies** (Jheng, Pele, Tak & Găman, 2026; best paper award at the International Conference on Business Excellence (ICBE) 2026): market indicators drawn as images and classified by a vision-language model (VLM), with natural-language rationales, tested out of sample on the S&P 500 — Lecture 14.
- Findings as stated in the papers; numbers on the other slides come from our own computations.

Explore it visually

- **TensorFlow Playground**
  - Choose the circle data set and keep only the inputs x_1, x_2. With no hidden layer the boundary stays a straight line, as in logistic regression; add one hidden layer of four neurons and the boundary closes around the circle.
  - Keep the hidden layer and set the activation to Linear: the boundary is straight again, because a composition of linear maps is linear.
- **3Blue1Brown, chapter 1**: what the weights, biases and layers of a network compute, on handwritten digits.

Further reading and resources

- **Bishop & Bishop (2024)**, chapters 4–6: linear models, classification, deep networks: bishopbook.com.
- **Dive into Deep Learning**, chapters 3–5, with PyTorch code: d2l.ai.
- **Goodfellow, Bengio & Courville**, chapter 6: deeplearningbook.org.
- **PyTorch tutorials**: pytorch.org/tutorials.
- **This course**: code, data, notebooks, Quantlets — github.com/danpele/NEURAL_BIZ.
- **Credit scoring in depth**: Baesens, Rösch & Scheule (2016).

Quiz 1: logistic regression as a neuron

- Which loss makes a one-neuron network with a sigmoid output equal to logistic regression?

Answer

- The cross-entropy, i.e. the Bernoulli negative log-likelihood
  - Its gradient is $\frac{1}{n}\sum(p_i - y_i)\mathbf{x}_i$.

Quiz 2: the perceptron on real data

- Why does the training error of the perceptron on HMEQ jump between 13.2 % and 23.4 %?

Answer

- The classes are not linearly separable, so Novikoff's bound does not apply: each correction of one loan breaks others.

Quiz 3: counting parameters

- How many parameters has an MLP on the HMEQ inputs ($d = 28$) with one hidden layer of 32 units and one output?

Answer

- $(d + 1) \cdot 32 + (32 + 1) \cdot 1 = 961$.

Quiz 4: the cost-optimal cut-off

- A missed default costs 5, a rejected good applicant costs 1.
  - Above which PD should the bank reject?

Answer

- $p \geq c_{\mathrm{FP}}/(c_{\mathrm{FP}} + c_{\mathrm{FN}}) = 1/6$
  - On German credit this cut-off lowers the logistic cost per applicant from 0.870 (cut-off $1/2$) to 0.554.

Quiz 5: when is an MLP worth it?

- On German credit the MLP has mean AUC 0.766 and logistic regression 0.777.
  - What do you recommend?

Answer

- Keep the logistic scorecard: the difference is not significant ($p = 0.362$) and the simpler model is easier to explain and validate.

References (1/2)

- Bishop, C. M. & Bishop, H. (2024). *Deep Learning: Foundations and Concepts*. Springer. doi:10.1007/978-3-031-45468-4
- Baesens, B., Rösch, D. & Scheule, H. (2016). *Credit Risk Analytics*. Wiley. doi:10.1002/9781119449560
- Będowska-Sójka, B., Wójcik, P. & Pele, D. T. (2026). Early warning systems for cryptocurrency markets: Predicting 'zombie' assets using machine learning. *The North American Journal of Economics and Finance*, 81, 102543. doi:10.1016/j.najef.2025.102543
- Chen, T. & Guestrin, C. (2016). XGBoost: A scalable tree boosting system. *Proceedings of KDD 2016*, 785–794. doi:10.1145/2939672.2939785
- Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems*, 2(4), 303–314. doi:10.1007/BF02551274
- Diaconu, D. & Pele, D. T. (2026). Explainable AI in financial fraud detection: Evidence from the IEEE-CIS fraud dataset. *Proceedings of the International Conference on Business Excellence*, 20(1), 118–132. doi:10.2478/picbe-2026-0015
- Grinsztajn, L., Oyallon, E. & Varoquaux, G. (2022). Why do tree-based models still outperform deep learning on typical tabular data? *Advances in Neural Information Processing Systems*, 35. arXiv:2207.08815
- Hofmann, H. (1994). Statlog (German Credit Data). UCI Machine Learning Repository. doi:10.24432/C5NC77
- Hornik, K. (1991). Approximation capabilities of multilayer feedforward networks. *Neural Networks*, 4(2), 251–257. doi:10.1016/0893-6080(91)90009-T
- Jheng, S.-L., Pele, D. T., Tak, R. & Găman, Ș. (2026). A multimodal vision-language framework for financial anomaly detection. *Proceedings of the International Conference on Business Excellence*, 20(1), 191–201. doi:10.2478/picbe-2026-0020
- Jheng, S.-L., Tak, R., Găman, Ș., Mazurencu-Marinescu-Pele, M. & Pele, D. T. (2026). Can foundation models manage risk? Zero-shot VaR and ES forecasting with conformal calibration in CEE markets. *Economic Computation and Economic Cybernetics Studies and Research*, 60(2), 267–290. doi:10.24818/18423264/60.2.26.14
- LeCun, Y., Bengio, Y. & Hinton, G. (2015). Deep learning. *Nature*, 521, 436–444. doi:10.1038/nature14539

References (2/2)

- Lessmann, S., Baesens, B., Seow, H.-V. & Thomas, L. C. (2015). Benchmarking state-of-the-art classification algorithms for credit scoring: An update of research. *European Journal of Operational Research*, 247(1), 124–136. doi:10.1016/j.ejor.2015.05.030
- McCulloch, W. S. & Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *The Bulletin of Mathematical Biophysics*, 5(4), 115–133. doi:10.1007/BF02478259
- Minsky, M. & Papert, S. (1969). *Perceptrons: An Introduction to Computational Geometry*. MIT Press.
- Nadeau, C. & Bengio, Y. (2003). Inference for the generalization error. *Machine Learning*, 52(3), 239–281. doi:10.1023/A:1024068626366
- Novikoff, A. B. J. (1962). On convergence proofs on perceptrons. *Proceedings of the Symposium on the Mathematical Theory of Automata*, 12, 615–622. Polytechnic Institute of Brooklyn.
- Pace, R. K. & Barry, R. (1997). Sparse spatial autoregressions. *Statistics & Probability Letters*, 33(3), 291–297. doi:10.1016/S0167-7152(96)00140-X
- Montúfar, G., Pascanu, R., Cho, K. & Bengio, Y. (2014). On the number of linear regions of deep neural networks. *Advances in Neural Information Processing Systems*, 27. arXiv:1402.1869
- Paszke, A. et al. (2019). PyTorch: An imperative style, high-performance deep learning library. *Advances in Neural Information Processing Systems*, 32. arXiv:1912.01703
- Pele, D. T., Bolovăneanu, V., Lin, M.-B., Ren, R., Ginavar, A. T., Spilak, B., Andrei, A.-V., Toma, F.-M., Lessmann, S. & Härdle, W. K. (2026). In the beginning was the Word: LLM-VaR and LLM-ES. *Expert Systems with Applications*, 295, 128676. doi:10.1016/j.eswa.2025.128676
- Rosenblatt, F. (1958). The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6), 386–408. doi:10.1037/h0042519
- Rumelhart, D. E., Hinton, G. E. & Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, 323, 533–536. doi:10.1038/323533a0

Appendix: the gradient of the logistic loss

$$\sigma'(z) = \sigma(z)\big(1-\sigma(z)\big), \qquad \ell_i = -y_i \log p_i - (1-y_i)\log(1-p_i), \quad p_i = \sigma(z_i)$$

$$\frac{\partial \ell_i}{\partial z_i} = -\frac{y_i}{p_i} p_i(1-p_i) + \frac{1-y_i}{1-p_i} p_i(1-p_i) = p_i - y_i$$

$$\frac{\partial \ell_i}{\partial \mathbf{w}} = \frac{\partial \ell_i}{\partial z_i}\,\frac{\partial z_i}{\partial \mathbf{w}} = (p_i - y_i)\,\mathbf{x}_i, \qquad \frac{\partial^2 \mathcal{L}}{\partial \mathbf{w}\,\partial \mathbf{w}^\top} = \frac{1}{n}\sum_i p_i(1-p_i)\mathbf{x}_i\mathbf{x}_i^\top \succeq 0$$

- The Hessian is positive semi-definite: the loss is convex, so gradient descent reaches the global minimum.
- The same chain rule, applied layer by layer, is backpropagation (Lecture 2).

Appendix: proof of the perceptron convergence theorem

- Start at $\mathbf{w}_0 = \mathbf{0}$ and let $\mathbf{w}_k$ be the weights after the k-th mistake, on example $(\mathbf{x}, y)$: $\mathbf{w}_k = \mathbf{w}_{k-1} + y\mathbf{x}$ ($\eta = 1$).
- Progress: $\mathbf{w}_k^\top \mathbf{w}^* = \mathbf{w}_{k-1}^\top \mathbf{w}^* + y\mathbf{x}^\top \mathbf{w}^* \geq \mathbf{w}_{k-1}^\top \mathbf{w}^* + \gamma$, so $\mathbf{w}_k^\top \mathbf{w}^* \geq k\gamma$.
- Growth: $\|\mathbf{w}_k\|^2 = \|\mathbf{w}_{k-1}\|^2 + 2y\mathbf{w}_{k-1}^\top \mathbf{x} + \|\mathbf{x}\|^2 \leq \|\mathbf{w}_{k-1}\|^2 + R^2$, since the example was misclassified
  - So $\|\mathbf{w}_k\|^2 \leq kR^2$.
- Cauchy–Schwarz: $k\gamma \leq \mathbf{w}_k^\top \mathbf{w}^* \leq \|\mathbf{w}_k\| \leq \sqrt{k}\, R$, hence $k \leq (R/\gamma)^2$.

(Novikoff, 1962)

Appendix: stacked linear layers collapse to one

$$\mathbf{W}^{(2)}\left(\mathbf{W}^{(1)}\mathbf{x}+\mathbf{b}^{(1)}\right)+\mathbf{b}^{(2)}=\underbrace{\mathbf{W}^{(2)}\mathbf{W}^{(1)}}_{\mathbf{W}}\mathbf{x}+\underbrace{\mathbf{W}^{(2)}\mathbf{b}^{(1)}+\mathbf{b}^{(2)}}_{\mathbf{b}}$$

- Any depth of linear layers is a single linear map: depth without activation adds parameters, not expressiveness.
- With a ReLU between them, f becomes piecewise linear with a number of pieces that grows with depth.