



Neural Networks and Deep Learning

Lecture 3: Regularisation and generalisation



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

- Diagnose overfitting from training and validation curves, and explain the bias-variance trade-off and double descent.
- Write weight decay, L1 penalties, dropout, early stopping and batch normalisation with their formulas, and say what each one controls.
- Design a validation scheme without leakage: hold-out, cross-validation, time-ordered splits, and resampling only inside the training set.
- Evaluate a fraud detector on imbalanced data with precision, recall, the area under the receiver operating characteristic (ROC) curve (AUC), the area under the precision-recall curve (PR-AUC) and the cost of errors.

Abbreviations

AdamW	Adam with decoupled weight decay
AI	artificial intelligence
AUC	area under the receiver operating characteristic curve
BN	batch normalisation
EUR	euro
GNN	graph neural network
GraphSAGE	graph sample and aggregate (a GNN architecture)
IEEE-CIS	Institute of Electrical and Electronics Engineers, Computational Intelligence Society
MAP	maximum a posteriori
MLP	multilayer perceptron
MSE	mean squared error
OpenML	Open Machine Learning platform
PCA	principal component analysis
PR-AUC	area under the precision-recall curve
ReLU	rectified linear unit
ROC	receiver operating characteristic
SGD	stochastic gradient descent
SHAP	Shapley additive explanations
UCI	University of California, Irvine (Machine Learning Repository)
ULB	Université libre de Bruxelles
XAI	explainable artificial intelligence
XGBoost	extreme gradient boosting

Outline

- Generalisation and overfitting
- Penalties on the weights
- Dropout, early stopping and batch normalisation
- Validation design
- Case study: card fraud detection
- Wrap-up

A model is judged on applicants it has never seen

- **In one sentence:** the training error is optimistic by construction; the generalisation gap is what regularisation tries to close.

$$\underbrace{R(\hat{f}) - \hat{R}_n(\hat{f})}_{\text{generalisation gap}}, \qquad \hat{f} = \arg\min_{f \in \mathcal{F}} \hat{R}_n(f) + \lambda\, \Omega(f)$$

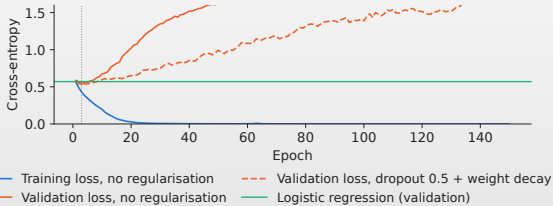
- **where**
  - R the expected loss on new data, $\hat{R}_n$ the training loss, $\mathcal{F}$ the model class
  - Ω a penalty on complexity, $\lambda \geq 0$ its weight
- Regularisation = anything that lowers R at the price of a higher $\hat{R}_n$: penalties, noise, early stopping, more data.
- Data: German credit from the University of California, Irvine (UCI) repository, California housing, and 28 924 card payments of the Université libre de Bruxelles (ULB) with 492 frauds
  - Models: multilayer perceptrons (MLP).

- **In practice:** always report the validation loss next to the training loss.

(Bishop & Bishop, 2024, ch. 9)

Without regularisation, a large MLP memorises 600 applicants in a few epochs

- **In one sentence:** MLP 256-256 (78 593 parameters) on German credit; dotted line: best validation epoch.



Source: UCI Machine Learning Repository, Statlog German credit. Data: 600 training and 200 validation applicants; MLP 256-256.

- Validation loss 0.536 at epoch 3, 2.187 after 150
  - With dropout and decay 1.767
  - Logistic 0.570.

- **In practice:** a training loss near zero on credit data is a warning, not a success.

(own computation)

NN_L3_overfit_german

The test error splits into bias, variance and noise

- **In one sentence:** a rigid model misses the pattern (bias), a flexible one chases the noise of its sample (variance).

$$\mathbb{E}_{\mathcal{D},\varepsilon}\left[\left(y-\hat{f}_{\mathcal{D}}(x)\right)^2\right] = \underbrace{\left(f(x)-\mathbb{E}_{\mathcal{D}}\hat{f}_{\mathcal{D}}(x)\right)^2}_{\text{bias}^2} + \underbrace{\operatorname{Var}_{\mathcal{D}}\left(\hat{f}_{\mathcal{D}}(x)\right)}_{\text{variance}} + \underbrace{\sigma^2}_{\text{noise}}$$

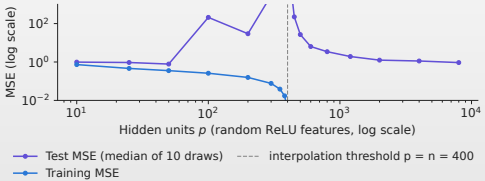
- **where**
  - $y = f(x) + \varepsilon$, $\operatorname{Var}(\varepsilon) = \sigma^2$, $\hat{f}_{\mathcal{D}}$ the model fitted on a random training sample $\mathcal{D}$, derivation in the appendix
- Regularisation trades a little bias for a large cut in variance
  - The noise σ^2 is a floor no model can beat.
- Classical U-curve: test error first falls, then rises with capacity
  - Very large networks can show a second descent (next frame).

- **In practice:** with few observations variance dominates: prefer simple models or strong regularisation.

(Hastie, Tibshirani & Friedman, 2009, ch. 7)

Past the interpolation threshold, more hidden units lower the test error again

- **In one sentence:** p random, frozen rectified linear units (ReLU), minimum-norm output, $n = 400$ block groups.



Source: StatLib, California housing, 1990 census (Pace and Barry, 1997). Data: 400 training and 5,000 test block groups per draw, 10 draws.

- Test mean squared error (MSE): 0.781 at $p = 50$, 1.0×10^{6} at $p = n$, 0.932 at $p = 8\,000$ (median of 10 draws).

- **In practice:** near $p = n$ a bigger model can be far worse, far past it size helps again.

(Belkin et al., 2019; Nakkiran et al., 2021)

NN_L3_double_descent

Outline

- Generalisation and overfitting
- Penalties on the weights
- Dropout, early stopping and batch normalisation
- Validation design
- Case study: card fraud detection
- Wrap-up

Weight decay as an L2 penalty and as a Gaussian prior on the weights

- **In one sentence:** large weights make the function steep and sensitive to noise; shrinking them makes it smoother.

$$\tilde{\mathcal{L}}(\mathbf{w}) = \mathcal{L}(\mathbf{w}) + \frac{\lambda}{2}\|\mathbf{w}\|_2^2 \;\Rightarrow\; \mathbf{w}_{t+1} = (1-\eta\lambda)\,\mathbf{w}_t - \eta\nabla\mathcal{L}(\mathbf{w}_t); \qquad \mathbf{w} \sim \mathcal{N}(\mathbf{0}, \lambda^{-1}\mathbf{I}) \Rightarrow \hat{\mathbf{w}}_{\mathrm{MAP}} = \arg\min \tilde{\mathcal{L}}$$

- **where**
  - $\lambda > 0$ the penalty strength, $(1-\eta\lambda)$ the decay per stochastic gradient descent (SGD) step, maximum a posteriori (MAP) estimate
- In a linear model this is ridge regression: along eigen-direction k, coefficients shrink by $\frac{d_k}{d_k+\lambda}$ (Hoerl & Kennard, 1970).
- With Adam, an L2 term in the loss is rescaled per weight
  - Adam with decoupled weight decay (AdamW) shrinks every weight by the same factor (Loshchilov & Hutter, 2019).

- **In practice:** weight decay is the first regulariser to tune
  - Search λ on a log grid with validation.

(Bishop & Bishop, 2024, ch. 9)

Experiment yourself: notebook §3

An L1 penalty sets weights exactly to zero

- **In one sentence:** the L1 ball has corners on the axes, so the penalised optimum often lands on one: sparsity.

$$\tilde{\mathcal{L}}(\mathbf{w}) = \mathcal{L}(\mathbf{w}) + \lambda\|\mathbf{w}\|_1, \qquad \text{proximal step:} \quad w_j \leftarrow \operatorname{sign}(u_j)\max\left(|u_j| - \eta\lambda, 0\right), \; u_j = w_j - \eta\,\partial_j\mathcal{L}$$

- **where**
  - $\|\mathbf{w}\|_1 = \sum_j |w_j|$, the soft-thresholding step sets small weights exactly to zero (the lasso of Tibshirani, 1996)
- L1 on the first layer of an MLP acts as variable selection: an input whose outgoing weights are all zero is switched off.
- Elastic net mixes both: $\lambda_1\|\mathbf{w}\|_1 + \frac{\lambda_2}{2}\|\mathbf{w}\|_2^2$.

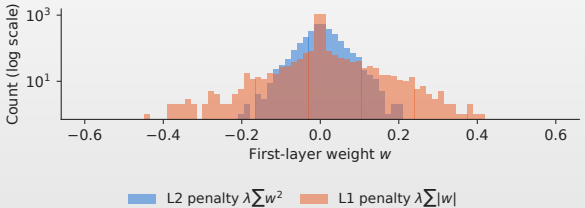
- **In practice:** use L1 when a short list of inputs is needed for the model documentation.

(Tibshirani, 1996)

Experiment yourself: notebook §3

First-layer weights under L1 and L2 penalties

- **In one sentence:** MLP with 64 hidden units on German credit, full-batch Adam, penalty on the first layer only.



Source: UCI Machine Learning Repository, Statlog German credit. Data: 600 training applicants; MLP with 64 hidden units, 3,072 first-layer weights.

- Weights with $|w| < 0.01$: 67.2 % under L1 against 26.3 % under L2 (3 072 weights)
  - Test AUC 0.768 with L1 against 0.761 with L2.

- **In practice:** a stronger L1 penalty would zero whole input columns, which can then leave the data pipeline.

(own computation) NN_L3_l1_l2_weights

Outline

- Generalisation and overfitting
- Penalties on the weights
- Dropout, early stopping and batch normalisation
- Validation design
- Case study: card fraud detection
- Wrap-up

Dropout trains an ensemble of thinned networks that share their weights

- **In one sentence:** randomly silencing units forbids any unit to rely on a few others: features must be useful on their own.

$$\text{training:}\quad \tilde{h}_j = \frac{m_j}{1-p}\, h_j, \quad m_j \overset{\text{i.i.d.}}{\sim} \text{Bernoulli}(1-p); \qquad \text{test:}\quad \tilde{h}_j = h_j$$

- **where**
  - p the dropout rate, m_j the random mask, dividing by $1-p$ (inverted dropout) keeps $\mathbb{E}[\tilde{h}_j] = h_j$, so nothing changes at test time
- A network with H droppable units samples from 2^H sub-networks
  - At test time the full network approximates their average (Srivastava et al., 2014).
- Typical rates: 0.1–0.3 in a tabular MLP, 0.5 in large fully connected layers
  - In PyTorch `model.eval()` switches it off.

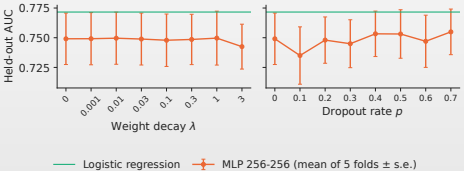
- **In practice:** forgetting `model.eval()` before scoring makes every prediction random.

(Srivastava et al., 2014)

Experiment yourself: notebook §4

On 600 applicants, regularisation does not close the gap to logistic regression

- **In one sentence:** MLP 256-256, 150 epochs without early stopping; held-out AUC over 5 folds for each λ and p.



Source: UCI Machine Learning Repository, Statlog German credit. Data: 1,000 applicants, 5-fold cross-validation, no early stopping (150 epochs).

- AUC 0.749 unregularised, 0.750 at $\lambda = 1$, 0.755 at $p = 0.7$
  - Logistic 0.772
  - Weight norm 18.0 vs 10.6 at $\lambda = 3$.

- **In practice:** on very small data, a smaller model beats a large regularised one.

(own computation)

NN_L3_wd_dropout

Early stopping: keep the weights of the best validation epoch

- **In one sentence:** training longer fits more noise; stop when the validation loss has not improved for a while.

$$t^* = \arg\min_{t \le T} \mathcal{L}_{\text{val}}(\mathbf{w}_t), \quad \text{stop when } t - t^* \ge P; \qquad \text{quadratic loss: } \eta\, t \approx \frac{1}{\lambda}$$

- **where**
  - P the patience in epochs, the last relation says that t steps of size η act like an L2 penalty of strength $\lambda \approx 1/(\eta t)$ (appendix)
- The validation set is used for the stopping decision: it is no longer an unbiased estimate, so a separate test set is needed.
- Free, cheap and compatible with every other regulariser (Prechelt, 1998).

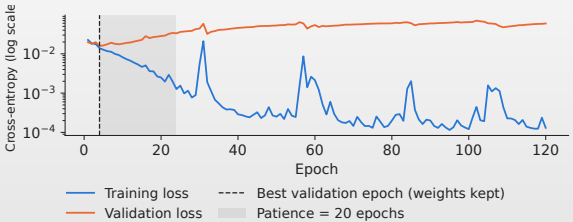
- **In practice:** save the best checkpoint, not the last one.

(Prechelt, 1998; Bishop & Bishop, 2024, ch. 9)

Experiment yourself: notebook §5

On card fraud, validation loss bottoms out early while training loss keeps falling

- **In one sentence:** MLP 256-256, time-ordered split; dashed: best validation epoch; grey: the patience window.



Source: ULB Machine Learning Group and Worldline, credit-card transactions (OpenML 1597). Data: time-ordered split: 17,354 training, 5,785 validation transactions.

- Best epoch 4, stop at 24 (patience 20)
  - Validation loss 0.0159 there, 0.0593 after 120 epochs.

- **In practice:** with rare events the network starts memorising the few training frauds within a handful of epochs.

(own computation)

NN_L3_early_stopping

Batch normalisation standardises each unit over the mini-batch

- **In one sentence:** keeping pre-activations centred and scaled lets deep networks train with larger steps.

$$\hat{z}_{ij} = \frac{z_{ij} - \mu_{\mathcal{B},j}}{\sqrt{\sigma^2_{\mathcal{B},j} + \epsilon}}, \qquad y_{ij} = \gamma_j \hat{z}_{ij} + \beta_j, \qquad \mu_{\mathcal{B},j} = \frac{1}{B} \sum_{i \in \mathcal{B}} z_{ij}$$

- **where**
  - $\sigma^2_{\mathcal{B},j}$ the batch variance of unit j, γ_j, β_j learned scale and shift, at test time running averages replace the batch statistics
- Batch normalisation (BN) smooths the loss landscape, which tolerates larger learning rates (Ioffe & Szegedy, 2015).
- The batch statistics add noise to each example's output: a mild regulariser
  - Layer normalisation (Lecture 7) avoids the batch dependence.

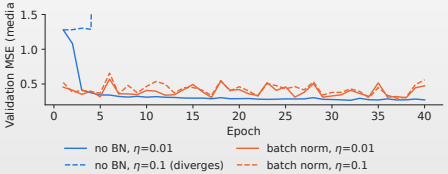
- **In practice:** with batch norm, `model.eval()` is mandatory and very small batches should be avoided.

(Ioffe & Szegedy, 2015)

Experiment yourself: notebook §6

Batch norm lets a deep MLP train at a rate at which it otherwise diverges

- **In one sentence:** 6 hidden layers of 64 units on California housing, SGD with momentum; medians of 3 seeds.



Source: StatLib, California housing, 1990 census (Pace and Barry, 1997). Data: 6 hidden layers of 64 units, SGD with momentum 0.9, 3 seeds.

- At $\eta = 0.1$: 3 of 3 seeds diverge without BN, best validation MSE 0.283 with BN
  - At $\eta = 0.01$: 0.265 without, 0.277 with BN.

- **In practice:** batch norm is mainly an optimisation aid
  - Keep early stopping, its validation error fluctuates.

(own computation)

NN_L3_batchnorm

Other regularisers: more data, noise, ensembles

- **Data augmentation**: label-preserving transformations (crops and flips for images, Lecture 4)
  - For tabular data, rarely safe.
- **Noise injection**: Gaussian noise on the inputs is equivalent, to first order, to an L2 penalty on the input gradient.
- **Label smoothing**: train on $(1-\epsilon)y + \epsilon/2$ instead of $0/1$ targets
  - Curbs over-confident probabilities.
- **Ensembles**: average several seeds or models
  - Variance falls roughly as $1/M$ for M weakly correlated members.
- **Smaller model or fewer inputs**: the oldest regulariser.
- **More data**: the only one with no side effects
  - One more year of transactions beats any penalty.

- **In practice:** our fraud case averages 5 seeds: a cheap ensemble that also reports the uncertainty.

(Bishop & Bishop, 2024, ch. 9)

Outline

- Generalisation and overfitting
- Penalties on the weights
- Dropout, early stopping and batch normalisation
- Validation design
- Case study: card fraud detection
- Wrap-up

Hold-out, cross-validation and nested cross-validation

- **In one sentence:** every choice made by looking at data must be paid for with data the choice has not seen.

$$\widehat{\mathrm{CV}}_K = \frac{1}{K}\sum_{k=1}^{K}\frac{1}{|\mathcal{D}_k|}\sum_{i\in\mathcal{D}_k}\ell\big(y_i, \hat{f}^{(-k)}(\mathbf{x}_i)\big)$$

- **where**
  - $\mathcal{D}_k$ fold k, $\hat{f}^{(-k)}$ the model fitted on the other $K-1$ folds, usually $K=5$ or 10, stratified on the label
- Hold-out: one train/validation/test split
  - Cheap, but noisy on small data (next frame).
- Nested cross-validation: an inner loop tunes hyperparameters, an outer loop measures the whole tuning procedure.
- Report the spread across folds, not only the mean
  - Compare models with the corrected t-test of Lecture 1.

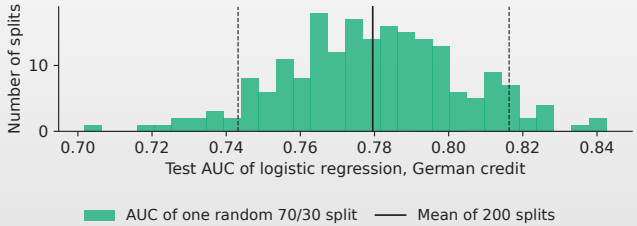
- **In practice:** the reported performance must come from data used neither for fitting nor for choosing.

(Hastie, Tibshirani & Friedman, 2009, ch. 7)

Experiment yourself: notebook §7

One random hold-out split can move the AUC by several points

- **In one sentence:** logistic regression on German credit, 200 random stratified 70/30 splits of the same applicants.



Source: UCI Machine Learning Repository, Statlog German credit. Data: 1,000 applicants, 200 random stratified splits.

- Test AUC from 0.702 to 0.842
  - 90 % of splits in [0.743; 0.816], standard deviation 0.024.

- **In practice:** on a small dataset, one split cannot separate two models: use repeated cross-validation.

(own computation)

NN_L3_holdout_noise

Information leakage into the validation set

⊡ **In one sentence:** any information from the evaluation rows that reaches training inflates the score and disappears in production.

Leak	Example	Fix
Preprocessing on all rows	scaler or imputer fitted before the split	fit on the training part only
Resampling before the split	copies of a fraud in both train and test	oversample inside the training fold
Ignoring time	random split of a time series or of payments	train on the past, test on the future
Target in the inputs	'days past due' used to predict default	keep only information known at decision time

⊡ Time series: expanding-window or blocked cross-validation; never shuffle (Bergmeir & Benítez, 2012).

⊡ **In practice:** write down, for every input, the moment it becomes known.

(Kaufman et al., 2012)

Experiment yourself: notebook §8

Oversampling before the split overstates performance

- **In one sentence:** the same MLP 128-128 and the same transactions, evaluated three ways; 5 seeds each.



Source: ULB Machine Learning Group and Worldline, credit-card transactions (OpenML 1597). Data: the same 28,924 transactions, MLP 128-128 with early stopping, 5 seeds.

- PR-AUC 0.843 with a time-ordered split, 0.853 with a random split, 0.921 with frauds copied 20 times before splitting.

- **In practice:** only the time-ordered estimate describes what the bank will see after deployment.

(own computation)

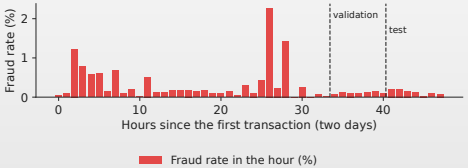
NN_L3_split_leakage

Outline

- Generalisation and overfitting
- Penalties on the weights
- Dropout, early stopping and batch normalisation
- Validation design
- Case study: card fraud detection
- Wrap-up

Fraud is rare and clustered in time

- **In one sentence:** Université libre de Bruxelles (ULB) and Worldline data, Open Machine Learning platform (OpenML): two days of card payments.



Source: ULB Machine Learning Group and Worldline, credit-card transactions (OpenML 1597). Data: September 2013, 2 days, 284,807 transactions (weighted sample).

- 284 807 payments, fraud rate 0.173 %
  - We keep all 492 frauds and 10 % of legitimate payments (weight 10)
  - Inputs: 28 principal component analysis (PCA) scores and the log amount.

- **In practice:** the hourly rate swings strongly: a random split would mix calm and attack hours.

(Dal Pozzolo et al., 2015)

NN_L3_fraud_by_hour

Performance measures for rare events: precision, recall and PR-AUC

- **In one sentence:** flagging nothing is 99.8 % accurate; the bank cares how many alerts are frauds and how many frauds raise an alert.

$$\text{precision} = \frac{TP}{TP + FP}, \quad \text{recall} = \frac{TP}{TP + FN}, \quad \text{PR-AUC} = \sum_k (R_k - R_{k-1})\, P_k$$

- **where**
  - TP, FP, FN true positives, false positives, false negatives at a cut-off, P_k, R_k precision and recall at the k-th threshold
- PR-AUC depends on the fraud rate (a random score gets PR-AUC = fraud rate)
  - AUC does not, which hides the false alarms (Saito & Rehmsmeier, 2015).
- Undersampled data: every legitimate payment counts with weight 10.00 in FP, so precision is that of the full population.

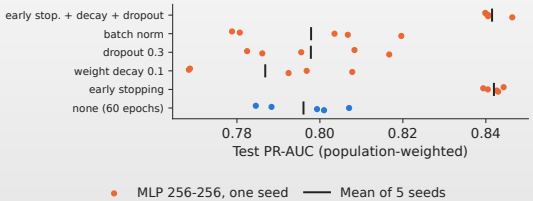
- **In practice:** report precision at the recall the investigation team can handle.

(Saito & Rehmsmeier, 2015)

Experiment yourself: notebook §9

On the test hours, early stopping is the regulariser with the clearest gain

- **In one sentence:** MLP 256-256, time-ordered split, 5 seeds per configuration; precision weighted to the population.



Source: ULB Machine Learning Group and Worldline, credit-card transactions (OpenML 1597). Data: time-ordered test set: last 5,785 transactions (75 frauds); 5 seeds.

- Test PR-AUC: none 0.796, early stopping 0.842, decay 0.787, dropout 0.798, batch norm 0.798, all 0.841.

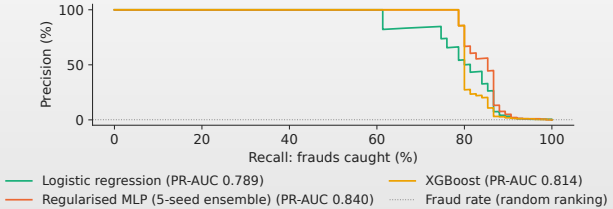
- **In practice:** with 60 fixed epochs, the other regularisers stay within the seed noise.

(own computation)

NN_L3_reg_ablation

The regularised MLP compared with logistic regression and gradient boosting

- **In one sentence:** test hours (last 20 %); the MLP averages 5 seeds with early stopping, decay and dropout.



Source: ULB Machine Learning Group and Worldline, credit-card transactions (OpenML 1597). Data: time-ordered test set: 5,785 transactions, 75 frauds, population-weighted.

- PR-AUC: logistic 0.789, MLP 0.840, extreme gradient boosting (XGBoost) 0.814
  - Recall at 80 % precision 74.7, 80.0, 80.0 %.

- **In practice:** with 75 test frauds, a few points of difference are within the noise.

(own computation)

NN_L3_fraud_pr

Class weights and the prior correction for undersampled data

- **In one sentence:** weighting the rare class changes the scores' level, not only their order; after undersampling, probabilities must be corrected.

$$\mathcal{L}_w = -\frac{1}{n}\sum_i \Big[w_+\, y_i \log p_i + w_-\,(1-y_i)\log(1-p_i)\Big], \qquad p = \frac{\beta\, p_s}{\beta\, p_s - p_s + 1}$$

- **where**
  - $w_\pm$ class weights, p_s the probability learned on data where legitimate payments were kept with rate β (here 0.1), p the corrected probability
- Class weights or resampling change the effective cut-off
  - The ranking, and thus PR-AUC, changes little.
- Whatever is done, the cut-off should come from costs (next frame), and probabilities must be corrected before they are used as such.

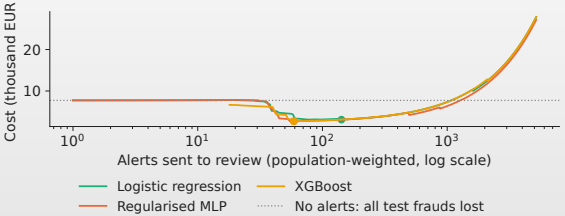
- **In practice:** never report probabilities from a resampled model to the business without the prior correction.

(Dal Pozzolo et al., 2015)

Experiment yourself: notebook §10

Number of alerts and the share of fraud losses avoided

- **In one sentence:** a missed fraud costs its amount; each alert costs a review of 5 euro (EUR), a constant chosen a priori.



Source: ULB Machine Learning Group and Worldline, credit-card transactions (OpenML 1597). Data: test period (last 20 % of the two days); review cost 5 EUR per alert.

- Test losses with no alerts 7 729 EUR
  - At the best alert count: logistic 3 093, MLP 2 694, XGBoost 2 695 EUR.

- **In practice:** the alert budget is the business decision, the model only ranks.

(own computation)

NN_L3_fraud_cost

Summary of regularisation methods

Method	What it does	Main knob	On our data
Weight decay	shrinks all weights	λ	smaller weights, same AUC (German)
L1	sets weights to zero	λ	most first-layer weights near zero
Dropout	trains an ensemble of sub-networks	p	within seed noise
Early stopping	stops before memorising	patience P	clear gain on fraud
Batch norm	stabilises training	batch size	trains at η where plain SGD diverges
Smaller model	fewer parameters	width, depth	logistic beats MLP on German credit

- **In practice:** start with early stopping and a modest model, then tune weight decay
  - Add dropout and batch norm for deep networks.

Outline

- Generalisation and overfitting
- Penalties on the weights
- Dropout, early stopping and batch normalisation
- Validation design
- Case study: card fraud detection
- Wrap-up

Key takeaways

- Overfitting shows as a falling training loss with a rising validation loss
  - On small data it starts within a few epochs.
- Weight decay (Gaussian prior), L1 (sparsity), dropout (ensemble), early stopping (implicit L2) and batch norm (optimisation aid) control capacity in different ways.
- Very large models can show double descent: the classical U-curve is not the whole story.
- Validation must mimic deployment: time-ordered splits, preprocessing and resampling inside the training part only (PR-AUC 0.843 honest vs 0.921 leaked).
- For rare events, use PR-AUC, precision at a workable recall and the cost of errors, with several seeds.

Key equations

$$\mathbb{E}(y - \hat{f})^2 = \text{bias}^2 + \text{Var}(\hat{f}) + \sigma^2$$

$$\mathbf{w}_{t+1} = (1 - \eta\lambda)\mathbf{w}_t - \eta\nabla\mathcal{L}$$

$$w_j \leftarrow \text{sign}(u_j)(|u_j| - \eta\lambda)_+$$

$$\tilde{h}_j = m_j h_j/(1 - p)$$

$$\hat{z} = (z - \mu_{\mathcal{B}})/\sqrt{\sigma^2_{\mathcal{B}} + \epsilon}$$

$$\eta\, t^* \approx 1/\lambda$$

$$P = \frac{TP}{TP+FP},\ R = \frac{TP}{TP+FN}$$

$$p = \beta p_s/(\beta p_s - p_s + 1)$$

Research from our group: fraud detection and early warning

- **Explainable artificial intelligence (AI) for fraud** (Diaconu & Pele, 2026): on the Institute of Electrical and Electronics Engineers, Computational Intelligence Society (IEEE-CIS) fraud data, XGBoost and CatBoost had the higher AUC, while a graph neural network (GNN), graph sample and aggregate (GraphSAGE), captured relational fraud patterns that tabular models cannot describe.
- **Explaining at several levels**: their explainable AI (XAI) framework combines Shapley additive explanations (SHAP) for variables with node and edge ablation for the graph
  - The paper concludes that fraud explanations should combine several model types — replicated in Case study 1 (course site).
- **Early warning of crypto 'zombies'** (Będowska-Sójka, Wójcik & Pele, 2026): machine-learning early warning systems that predict which crypto-assets become 'zombie' assets
  - An early-warning model must be validated forward in time, as in Section 4.
- Findings as stated in the papers; numbers on the other slides come from our own computations.

Explore it visually

- **TensorFlow Playground**
  - Choose the spiral data set, noise 50 and 10 % training data, and a network with several hidden layers: the test loss stays above the training loss, which is overfitting.
  - Set the regularisation to L2, then L1, with rate 0.01: L1 drives many weights to zero, drawn as thin connections, while L2 shrinks them all.

Further reading and resources

- **Bishop & Bishop (2024)**, chapter 9: regularisation: bishopbook.com.
- **Hastie, Tibshirani & Friedman (2009)**, chapter 7: model assessment and selection: hastie.su.domains.
- **Dive into Deep Learning**, sections 3.7, 5.6, 8.5: d2l.ai.
- **Fraud detection handbook** (Le Borgne et al., ULB): fraud-detection-handbook.github.io.
- **This course**: code, data, notebooks, Quantlets — github.com/danpele/NEURAL_BIZ.
- **Leakage**: Kaufman et al. (2012).

Quiz 1: reading the curves

- Training loss keeps falling, validation loss rises after epoch 3.
  - What do you do?

Answer

- The network overfits: keep the weights of epoch 3 (early stopping) and add weight decay or dropout
  - Do not train longer.

Quiz 2: L1 or L2

- Which penalty can switch an input off completely, and why?

Answer

- L1: its soft-thresholding step sets small weights exactly to zero
  - Here 67.2 % of the first-layer weights are below 0.01 (L2: 26.3 %).

Quiz 3: dropout at test time

- With inverted dropout and $p = 0.3$, what must be done to the activations at test time?

Answer

- Nothing: dividing by $1 - p$ during training already keeps the expectation
  - Just call `model.eval()`.

Quiz 4: oversampling

- Why does oversampling frauds before the split raise PR-AUC from 0.843 to 0.921?

Answer

- Copies of the same fraud land in training and test, so the model is tested on frauds it has memorised
  - Part of the gap is also the ignored time order.

Quiz 5: the right metric

- A fraud model has AUC 0.977.
  - Is that enough to deploy it?

Answer

- No: with a fraud rate of 0.173 % a high AUC can hide many false alarms
  - Check PR-AUC (0.840), precision at the target recall and the cost of alerts.

References (1/2)

- 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
- Belkin, M., Hsu, D., Ma, S. & Mandal, S. (2019). Reconciling modern machine-learning practice and the classical bias–variance trade-off. *Proceedings of the National Academy of Sciences*, 116(32), 15849–15854. doi:10.1073/pnas.1903070116
- Bergmeir, C. & Benítez, J. M. (2012). On the use of cross-validation for time series predictor evaluation. *Information Sciences*, 191, 192–213. doi:10.1016/j.ins.2011.12.028
- Bishop, C. M. & Bishop, H. (2024). *Deep Learning: Foundations and Concepts*. Springer. doi:10.1007/978-3-031-45468-4
- Chen, T. & Guestrin, C. (2016). XGBoost: A scalable tree boosting system. *Proceedings of KDD 2016*, 785–794. doi:10.1145/2939672.2939785
- Dal Pozzolo, A., Caelen, O., Johnson, R. A. & Bontempi, G. (2015). Calibrating probability with undersampling for unbalanced classification. *IEEE Symposium Series on Computational Intelligence*, 159–166. doi:10.1109/SSCI.2015.33
- 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
- Hastie, T., Tibshirani, R. & Friedman, J. (2009). *The Elements of Statistical Learning* (2nd ed.). Springer. doi:10.1007/978-0-387-84858-7
- Hoerl, A. E. & Kennard, R. W. (1970). Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12(1), 55–67. doi:10.1080/00401706.1970.10488634
- Hofmann, H. (1994). Statlog (German Credit Data). UCI Machine Learning Repository. doi:10.24432/C5NC77

References (2/2)

- Ioffe, S. & Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. *Proceedings of ICML*, 448–456. arXiv:1502.03167
- Kaufman, S., Rosset, S., Perlich, C. & Stitelman, O. (2012). Leakage in data mining: Formulation, detection, and avoidance. *ACM Transactions on Knowledge Discovery from Data*, 6(4), 1–21. doi:10.1145/2382577.2382579
- Loshchilov, I. & Hutter, F. (2019). Decoupled weight decay regularization. *International Conference on Learning Representations*. arXiv:1711.05101
- Nakkiran, P., Kaplun, G., Bansal, Y., Yang, T., Barak, B. & Sutskever, I. (2021). Deep double descent: where bigger models and more data hurt. *Journal of Statistical Mechanics*, 124003. doi:10.1088/1742-5468/ac3a74
- Pace, R. K. & Barry, R. (1997). Sparse spatial autoregressions. *Statistics & Probability Letters*, 33(3), 291–297. doi:10.1016/S0167-7152(96)00140-X
- Prechelt, L. (1998). Early stopping — but when? In *Neural Networks: Tricks of the Trade*, LNCS 1524, 55–69. Springer. doi:10.1007/3-540-49430-8_3
- Saito, T. & Rehmsmeier, M. (2015). The precision-recall plot is more informative than the ROC plot when evaluating binary classifiers on imbalanced datasets. *PLOS ONE*, 10(3), e0118432. doi:10.1371/journal.pone.0118432
- Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I. & Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15, 1929–1958.
- Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society B*, 58(1), 267–288. doi:10.1111/j.2517-6161.1996.tb02080.x

Appendix: derivation of the bias-variance decomposition

$$\mathbb{E}\big[(y-\hat{f})^2\big] = \mathbb{E}\big[(f+\varepsilon-\hat{f})^2\big] = \sigma^2 + \mathbb{E}\big[(f-\hat{f})^2\big]$$
$$\mathbb{E}\big[(f-\hat{f})^2\big] = \mathbb{E}\big[(f-\bar{f}+\bar{f}-\hat{f})^2\big] = (f-\bar{f})^2 + \mathbb{E}\big[(\hat{f}-\bar{f})^2\big], \qquad \bar{f} = \mathbb{E}_{\mathcal{D}}\hat{f}$$

- The cross terms vanish because $\mathbb{E}[\varepsilon] = 0$, ε is independent of $\mathcal{D}$, and $\mathbb{E}[\hat{f} - \bar{f}] = 0$.
- Averaging over x gives the decomposition of the expected test MSE.

Appendix: why early stopping acts like an L2 penalty

$$\mathcal{L}(\mathbf{w}) \approx \tfrac{1}{2}(\mathbf{w}-\mathbf{w}^*)^\top \mathbf{H}(\mathbf{w}-\mathbf{w}^*),\ \mathbf{w}_0 = \mathbf{0}: \quad \mathbf{w}_t = \left[\mathbf{I} - (\mathbf{I}-\eta\mathbf{H})^t\right]\mathbf{w}^*$$

$$\text{L2: } \tilde{\mathbf{w}} = (\mathbf{H}+\lambda\mathbf{I})^{-1}\mathbf{H}\mathbf{w}^*; \qquad \text{on eigen-direction } k\text{: } 1-(1-\eta h_k)^t \approx \frac{h_k}{h_k+\lambda} \text{ when } \lambda \approx \frac{1}{\eta t}$$

- Both shrink directions of small curvature h_k most
  - Stopping after t steps is a penalty of strength about $1/(\eta t)$.

(Bishop & Bishop, 2024, ch. 9)

Appendix: weight decay and the L2 penalty under SGD and under Adam

$$\text{SGD: } \mathbf{w} - \eta\nabla\left(\mathcal{L} + \tfrac{\lambda}{2}\|\mathbf{w}\|^2\right) = (1-\eta\lambda)\mathbf{w} - \eta\nabla\mathcal{L}$$

$$\text{Adam+L2: } \mathbf{w} - \eta\,\frac{\hat{m}(\nabla\mathcal{L} + \lambda\mathbf{w})}{\sqrt{\hat{v}} + \epsilon} \qquad \text{vs.} \qquad \text{AdamW: } (1-\eta\lambda)\mathbf{w} - \eta\,\frac{\hat{m}(\nabla\mathcal{L})}{\sqrt{\hat{v}} + \epsilon}$$

- With Adam, the L2 gradient is divided by $\sqrt{\hat{v}}$: weights with large gradients are barely decayed.
- AdamW restores uniform decay
  - It is the optimiser used in this lecture's experiments.

(Loshchilov & Hutter, 2019)