Lecture 1.5: Regularization

Module 1: Foundations of Neural Networks
Deep and Reinforcement Learning

Srikanth Pai

Madras School of Economics

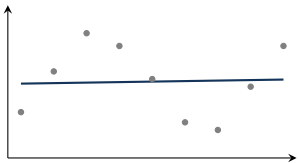
The Other Half of the Story

Optimisation minimises $\hat{R}$. But we care about R, the error on *new* data.

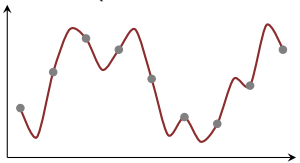
Regularisation is anything that closes this gap: it lowers test error without lowering training error.

Underfitting vs Overfitting

Underfit (too simple)



Overfit (memorises noise)



Both fail on new data. We need a **preference** for the right kind of function.

No Free Lunch $\Rightarrow$ Encode a Prior

No Free Lunch (Wolpert, 1996)

Measure error *off the training set*. Averaged uniformly over all target functions f, any two learning algorithms a_1, a_2 have equal expected off-training-set error:

$$\sum_f \mathbb{E}[E_{\text{ots}} \mid f,\, d,\, a_1] \;=\; \sum_f \mathbb{E}[E_{\text{ots}} \mid f,\, d,\, a_2]\,.$$

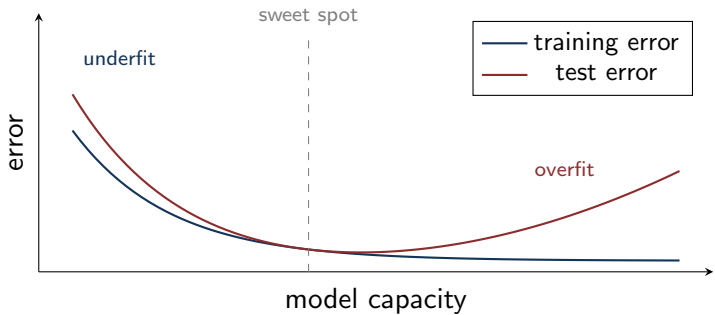
Here d is the training set and E_{ots} is the error on inputs not in d. The average is over a *uniform* prior on f.

To beat chance you must assume some functions are more plausible than others.

Regularisation (Goodfellow, Bengio, Courville, 2016)

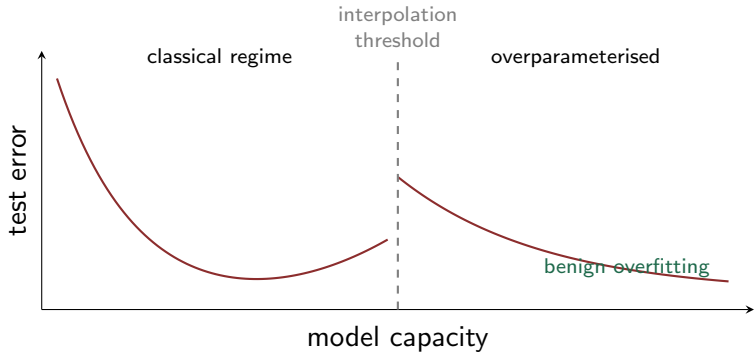
Any modification to the algorithm intended to reduce generalisation error but not training error.

The Classical Picture: Bias–Variance U-Curve



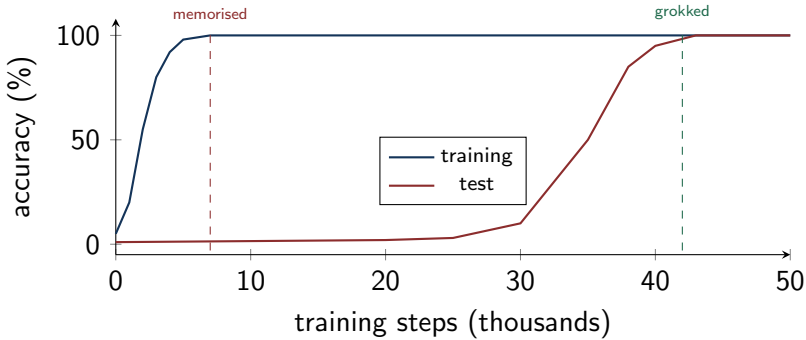
Classical advice: pick the sweet spot, avoid large models. **Modern nets break this.**

Surprise 1: Double Descent



Nakkiran et al. (2019); Belkin et al. (2019). Past the threshold, GD finds the **minimum-norm** solution and test error falls again.

Surprise 2: Grokking (Delayed Generalisation)



Power et al. (2022), $a + b \bmod 97$. **Early stopping would kill it**; **weight decay speeds it up**.

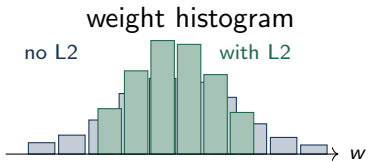
Fix 1: L2 / Weight Decay

Add $\frac{\lambda}{2}\|\mathbf{w}\|^2$ to the loss. The step becomes

$$\mathbf{w} \leftarrow \underbrace{(1-\eta\lambda)}_{\text{decay}}\mathbf{w} - \eta\nabla_{\mathbf{w}}L.$$

Prior view: Gaussian prior $\mathbf{w} \sim \mathcal{N}(0, \lambda^{-1}I)$: absent data, weights should be small.

Shrinks each direction by $\frac{\lambda_i}{\lambda_i + \lambda}$: the directions the **data barely constrains** are pulled to zero.



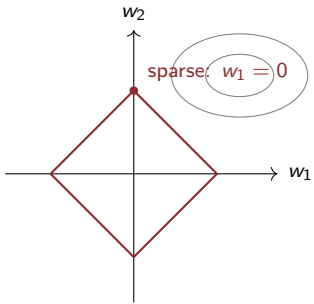
L2 concentrates weights near 0.

Fix 2: L1 / Sparsity

Add $\|\mathbf{w}\|_1 = \sum_j |w_j|$. The step subtracts a *constant* $\eta\lambda\,\mathrm{sign}(w_j)$, so small weights hit **exactly zero**.

Prior view: Laplace prior, sharp peak at 0.

Geometry: the L_1 ball has *corners on the axes*; the loss contour first touches a corner, so some coordinates are zero.



L1 when the solution is sparse or must be interpretable; **L2** when signal is spread across many weights (the deep-learning default).

Warning: L2 $\neq$ Weight Decay under Adam

In **SGD** the two are identical. Under **Adam** they are not: the update divides by $\sqrt{\hat{s}}$, so an L2 term in the loss gets divided too, and high-variance parameters receive **less decay than intended**.

AdamW (Loshchilov & Hutter, 2019) applies decay directly to the weights:

$$\mathbf{w} \leftarrow (1 - \eta\lambda)\mathbf{w} - \eta\, \hat{\mathbf{m}}/(\sqrt{\hat{\mathbf{s}}} + \varepsilon).$$

Rule: with Adam, use AdamW weight decay, never an L2 penalty in the loss.

Fix 3: Dropout

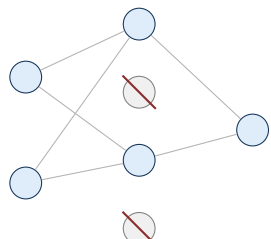
Each step, drop each hidden unit with probability $1 - p$:

$$\tilde{h}_j = m_j h_j, \quad m_j \sim \text{Bernoulli}(p).$$

Units cannot rely on each other, so no **co-adaptation**.

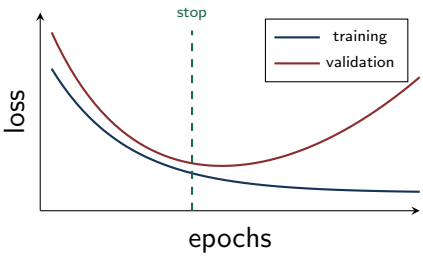
n units $\Rightarrow 2^n$ thinned subnetworks, all sharing weights. Test time $\approx$ averaging the ensemble.

Test: scale by p (or use inverted dropout: scale by $1/p$ in training).



dropped units (red) sit out this step

Fix 4: Early Stopping



Stop when validation loss stops improving; keep the best checkpoint.

Under a quadratic loss it *is* L2 with $\lambda \approx \frac{1}{\eta T}$: stop early = strong penalty.

Caution: on grokking tasks, stopping at the first trough prevents generalisation.

Fix 5: Batch Normalisation

Normalise each unit's preactivation over the mini-batch, then rescale:

$$\hat{z}_j = \frac{z_j - \mu_j}{\sqrt{\sigma_j^2 + \varepsilon}}, \qquad h_j = \gamma_j \hat{z}_j + \beta_j \quad (\gamma, \beta \text{ learned}).$$

Helps:

- larger learning rates, faster training
- less sensitive to initialisation
- mild regulariser (batch noise)

Notes:

- test time uses running statistics
- often reduces the need for dropout

Which Regulariser to Use?

Situation	**Use**	**Why**
General default	L2 + dropout	robust, widely effective
Sparse / interpretable	L1 (or elastic net)	exact zeros
Deep / transformers	batch/layer norm + AdamW	handles scale
Small dataset	early stopping	no extra hyperparameters
Using Adam	AdamW	L2 $\neq$ weight decay

Every one encodes a prior. The best choice depends on the structure you expect.

Discussion

Talk it through

1. L2 and early stopping are "the same" under a quadratic loss. In what sense? Which hyperparameter of one plays the role of λ in the other?
2. Why does L1 produce *exact* zeros while L2 only makes weights small?
3. You add an L2 penalty to the loss and train with Adam; a colleague says the decay is uneven across parameters. Are they right?
4. Grokking: why would early stopping be a mistake here?

Practice: Two Short Problems

Try it (5 min)

1. Batch norm. A unit has preactivations $z = (3, 1, -1, -3)$ over a batch of 4. With $\varepsilon = 0$, compute $\hat{z}$. If $\gamma = 1, \beta = 0$, what are the outputs, and how do they compare to z?

2. Diagnose. A net on 500 images reaches 99% train, 62% test accuracy. (a) Underfit or overfit? (b) Name two regularisers that would help and why.

Key Takeaways

Regularisation closes the gap between training loss and test loss.

- Every regulariser **encodes a prior**; No Free Lunch says you need one.
- **L2**: Gaussian prior, shrink weights. **L1**: Laplace prior, exact zeros.
- **Dropout**: an ensemble of thinned subnetworks. **Batch norm**: normalise, and regularise mildly.
- **Early stopping** $\approx$ L2; but beware delayed generalisation (grokking).
- With Adam use **AdamW**, not an L2 penalty.