Introduction to Deep Learning

Lecture 03: Training the Model

MSc in Data Science & Artificial Intelligence Strategy

Saeed VARASTEH YAZDI

em lyon business school

Paris

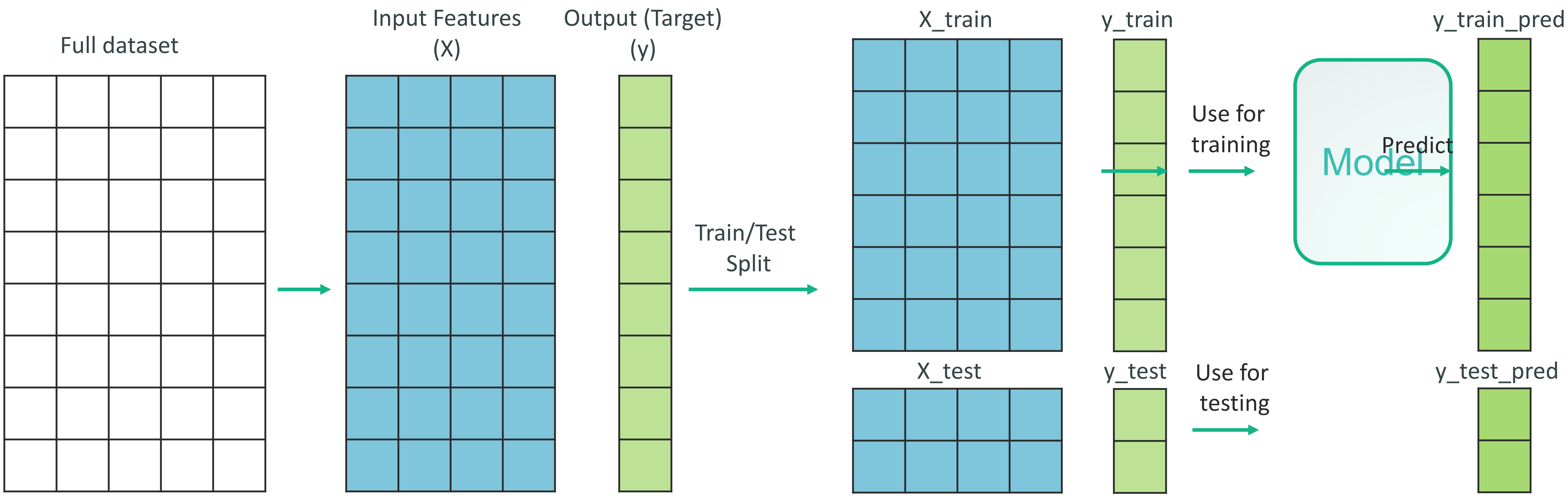
Training the Model

What is training a neural network?

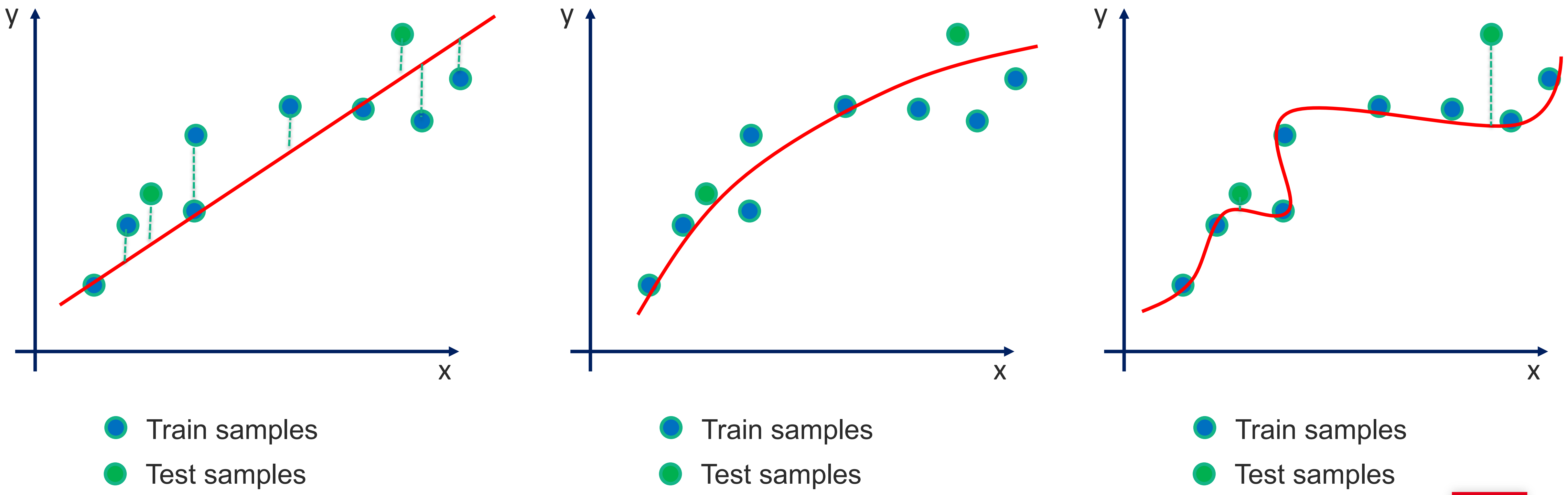
Given a dataset with ground truth training pairs

Find optimal weights $\boldsymbol{W}$ using stochastic gradient descent, such that the loss function is minimized

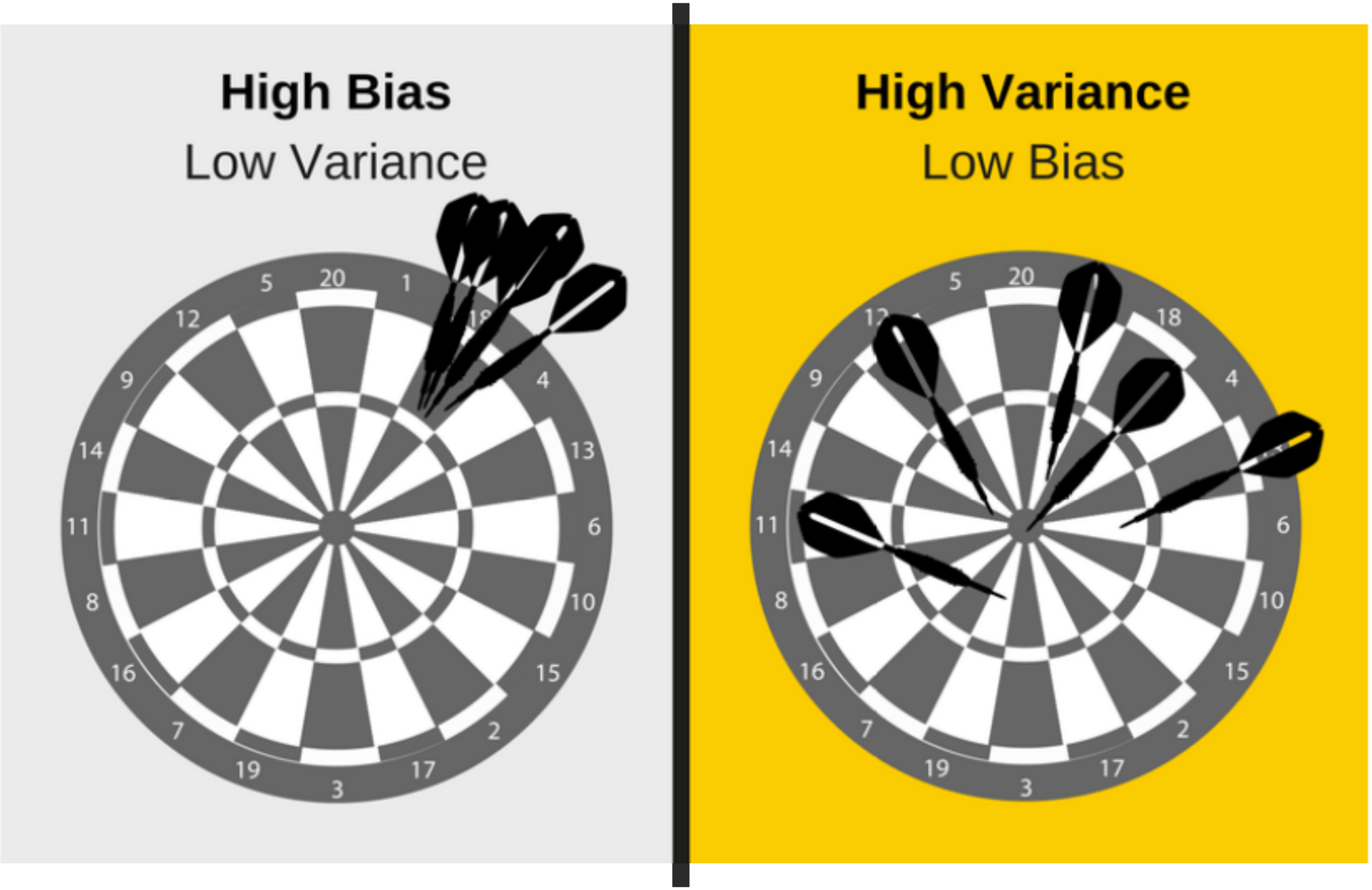
Recall: Machine Learning



Over and Under Fitting



Bias-Variance Trade-off



Bias-Variance Trade-off

High bias, low variance algorithms train models that are consistent, but inaccurate *on average*.

High variance, low bias algorithms train models that are accurate *on average*, but inconsistent.

But why is there a tradeoff?

Low variance algos tend to be **less complex**, with simple or rigid underlying structure.

- e.g. Regression
- e.g. Naive Bayes
- *Linear algos*
- *Parametric algos*

Low bias algos tend to be **more complex**, with flexible underlying structure.

- e.g. Decision trees
- e.g. Nearest neighbors
- *Non-linear algos*
- *Non-parametric algos*

Bias-Variance Trade-off

Within each algo family, there's a tradeoff too...

For example, regression can be **regularized** to further reduce complexity.

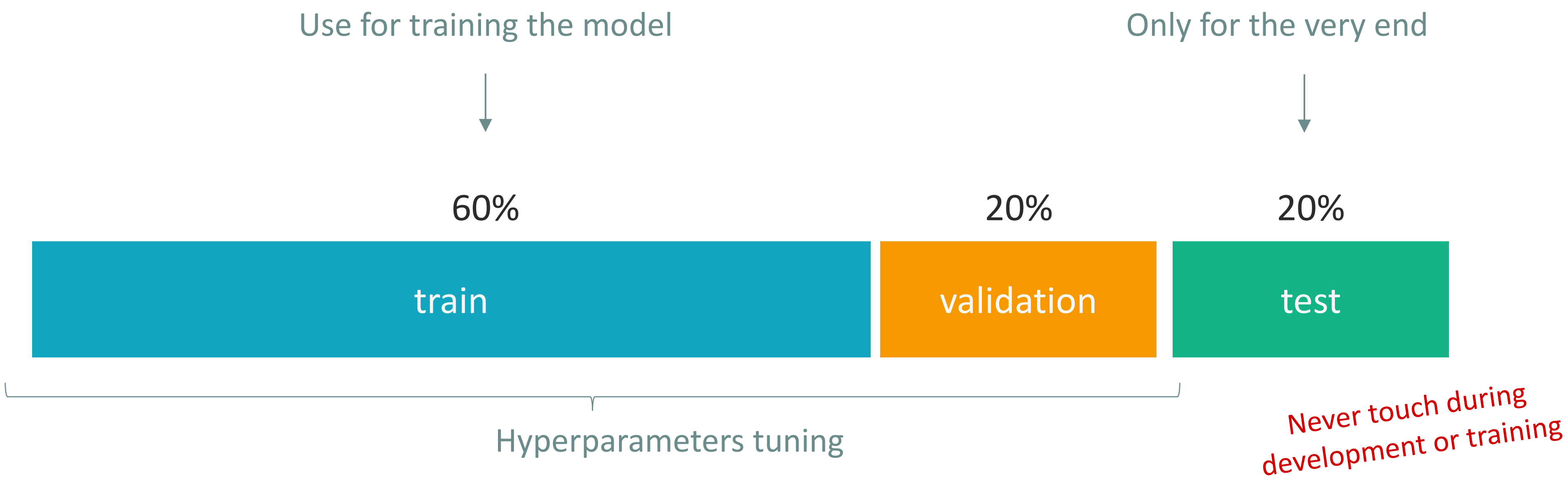
For example, decision trees can be **pruned** to reduce complexity.

That's why a solid model training methodology is key.

Algos that are not complex enough produce **underfit** models that can't learn the signal from the data.

Algo that are too complex produce **overfit** models that memorize the noise instead of the signal.

Data Split


Use for training the model
Only for the very end
60%
20%
20%
train
validation
test
Hyperparameters tuning
Never touch during development or training

Bias-Variance Trade-off

Training error is high?

Training error not going down

Bigger model
Different architecture
Longer training

Bias!

Training and validation error are high?

You don't have the ability to estimate your model from limited data

More data
Different architecture
Regularization

Variance!

Validation error is high?

Validation error not going down

Train and validation are not similar
Different architecture
Generate data

Train-Test mismatch

Testing error is high?

Tests are different than during training

More validation data

Overfit the validation

Over and Under Fitting

Overfitting

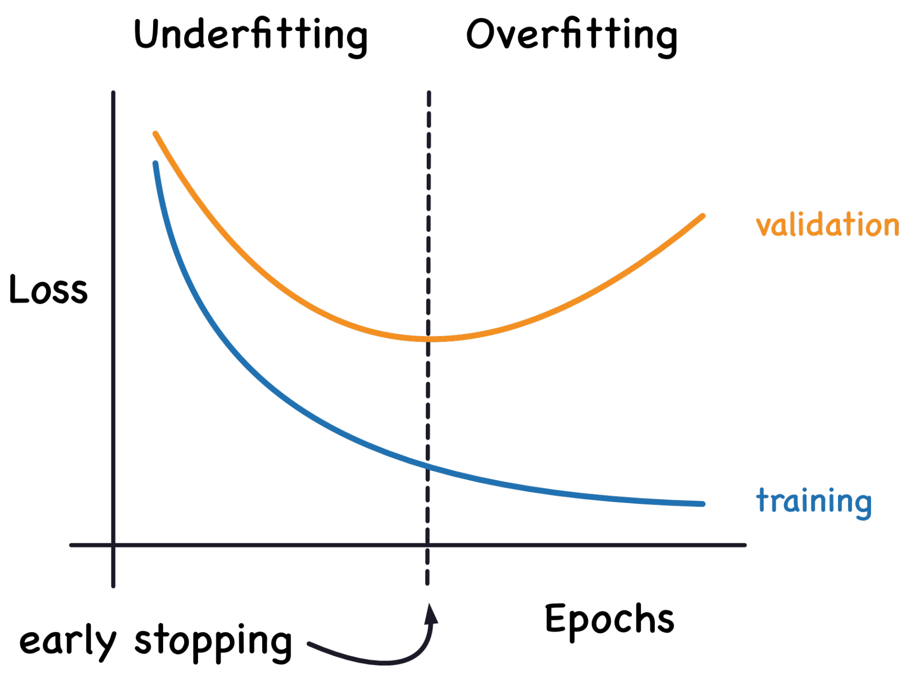
Training loss decreases and validation loss increases

- Fewer parameters, less complexity
- Regularization
- Early stopping
- Ensembling (Better suited for machine learning)
- Cross Validation (Better suited for machine learning)

Underfitting

Training and validation losses decrease even at the end of training

- Increase the size or number of parameters in the model.
- Increase the complexity of the model. Use a more powerful model.
- Increase training time until cost function is minimized.



Regularization

Regularization Techniques

Regularization: Penalize Model Complexity

Add regularization term to the loss function	Other regularizations	Model training techniques
L2 regularization	Max-out	Early stopping
L1 regularization	Dropout	
	Label smoothing	

L1 and L2 Regularizations

A Cat classifier takes different inputs

L1 regularization

Furry

Has two eyes

Has a tail

Has paws

Has two ears

L1 regularization will focus all the attention to a few key (selected) feature

L2 regularization

Furry

Has two eyes

Has a tail

Has paws

Has two ears

L2 regularization will take all features (with similar importance) into account to make decisions

L1 and L2 Regularizations

$$L(\theta) = \frac{1}{m}\sum_{i=1}^{m}(\boldsymbol{\theta}^T \boldsymbol{x}^{(i)} - y^{(i)})^2 \boxed{+ R(\boldsymbol{\theta})}$$

L1 regularization

$$R(\boldsymbol{\theta}) = \sum_j |\theta_j|$$

$$\boldsymbol{\theta}_1 \rightarrow 0 + 0.75 + 0 = 0.75$$

$$\boldsymbol{\theta}_2 \rightarrow 0.25 + 0.5 + 0.5 = 1.$$

Ignores 2 features
Enforce sparsity

$$\boldsymbol{x} = [1, 2, 1]$$

$$\boldsymbol{\theta}_1 = [0, 0.75, 0]$$

$$\boldsymbol{\theta}_2 = [0.25, 0.5, 0.25]$$

L2 regularization

$$R(\boldsymbol{\theta}) = \sum_j {\theta_j}^2$$

$$\boldsymbol{\theta}_1 \rightarrow 0 + 0.75^2 + 0 = 0.56$$

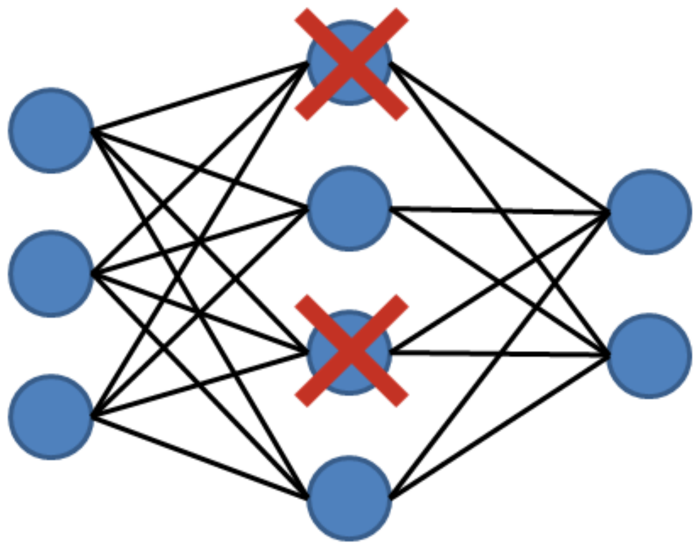
$$\boldsymbol{\theta}_2 \rightarrow 0.25^2 + 0.5^2 + 0.25^2 = 0.37$$

Takes information from all features
Enforce similar weights values

Dropout Regularization

- At training time, in each forward pass, turn off some neurons with probability 1-p (or present with probability p)

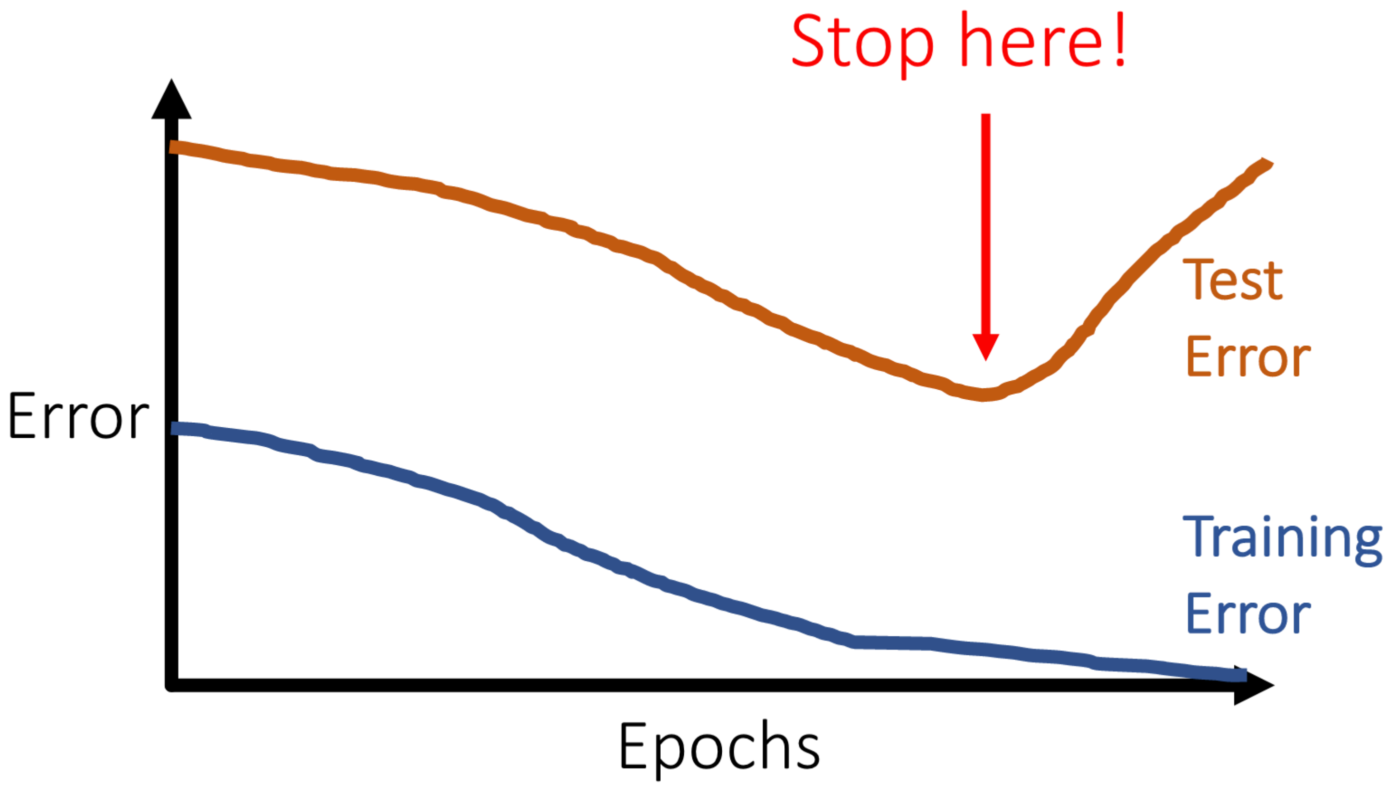
- At test time, to have deterministic behaviour, multiply output of neuron by p



Intuitions:

- Prevent “co-adaptation” of units,
- Increase robustness to noise
- Train implicit ensemble

Early Stopping

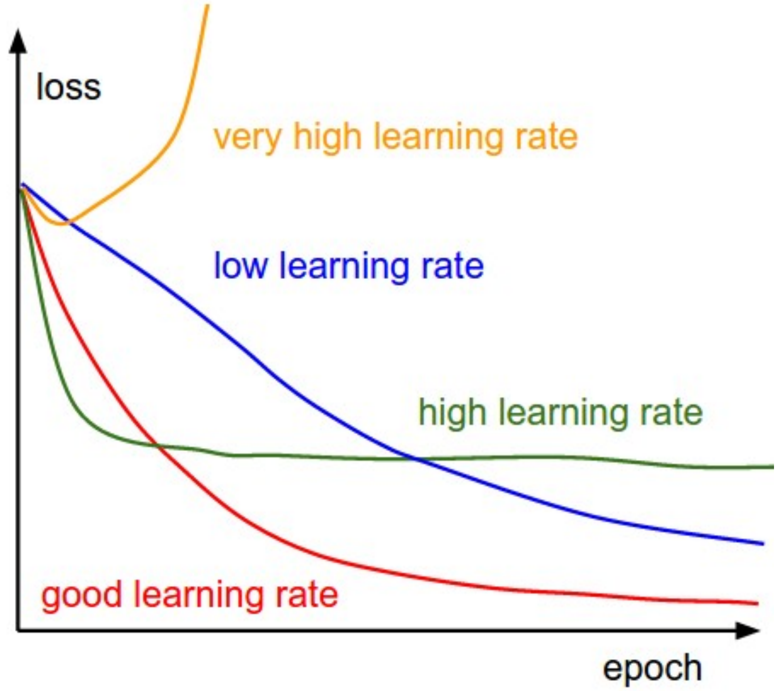


-[Introduction to Deep Learning]-

Training the Model

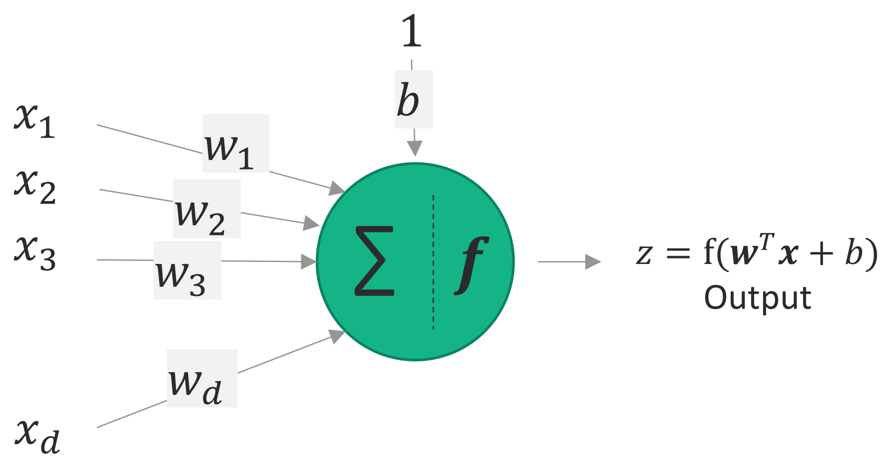
Learning Rate

- Too big = diverge, too small = slow convergence
- No "one learning rate to rule them all"
- Start from a high value and keep cutting by half if model diverges
  - If diverges too quickly, reduce by a factor of 10
- When to decay?
  - Loss is exploding or fluctuating
  - Loss has stopped decreasing
- When to increase?
  - Decreasing, but very slowly
  - Overfitting
  - Jump out of local minima



Activation Functions

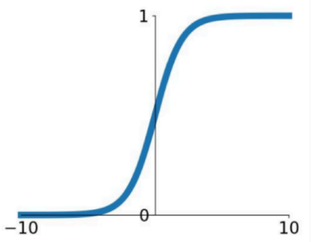
The choice of f



Activation Functions

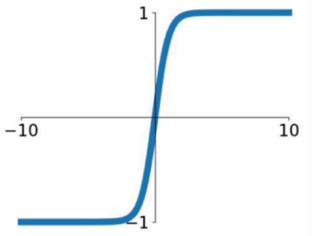
Sigmoid

$$\sigma(x) = \frac{1}{1+e^{-x}}$$



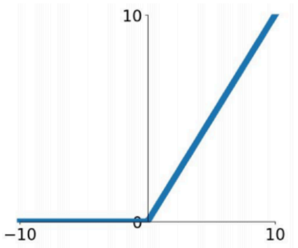
tanh

$$\tanh(x)$$



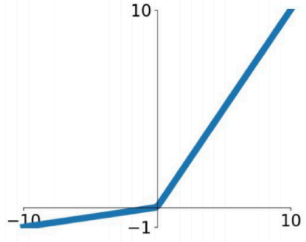
ReLU

$$\max(0, x)$$



Leaky ReLU

$$\max(0.1x, x)$$

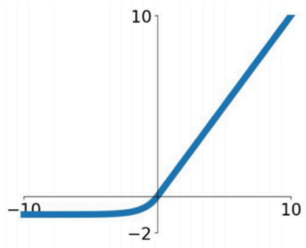


Maxout

$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$

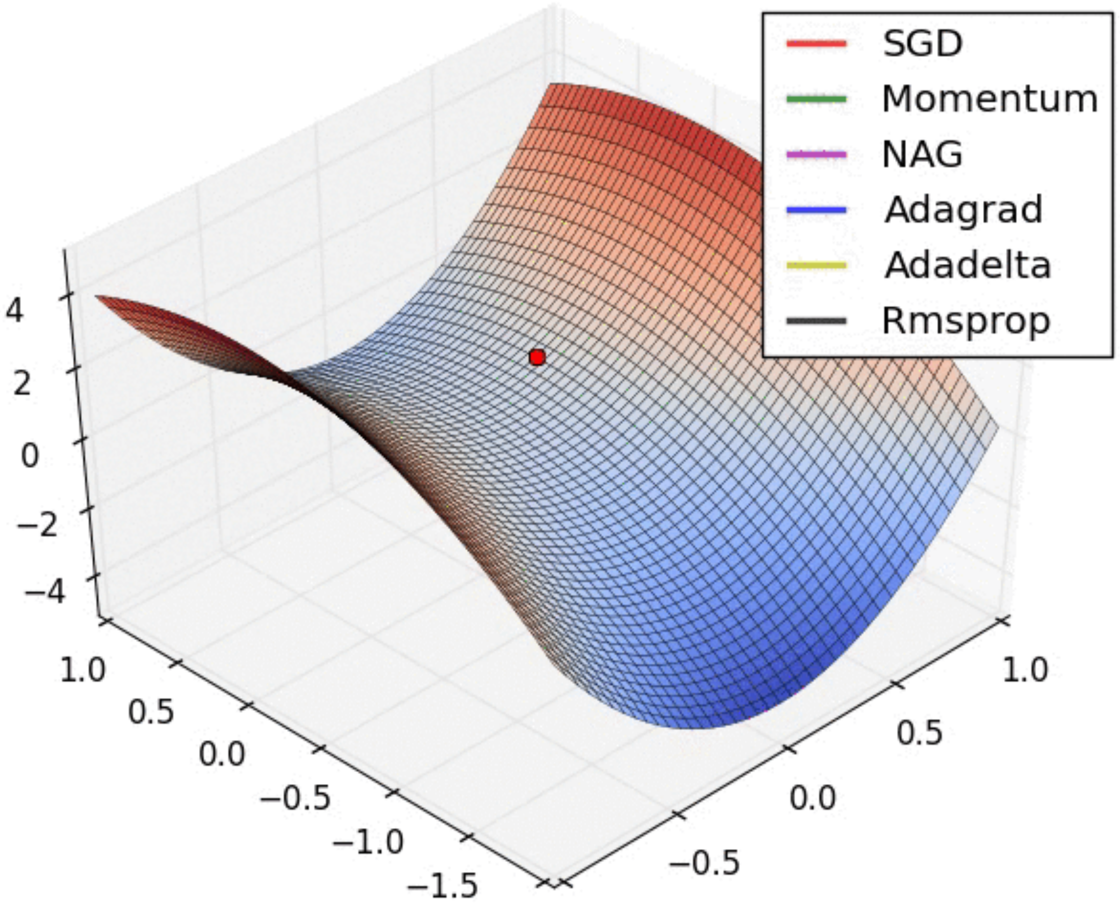


Activation Functions

- Use ReLU. Be careful with your learning rates
- Try out Leaky ReLU / Maxout / ELU. Revert back to ReLU!
- Try out tanh (if you're feeling adventurous) but don't expect much!
- Don't use sigmoid, unless you've time to spare and feel like exploring neural networks from the 90s

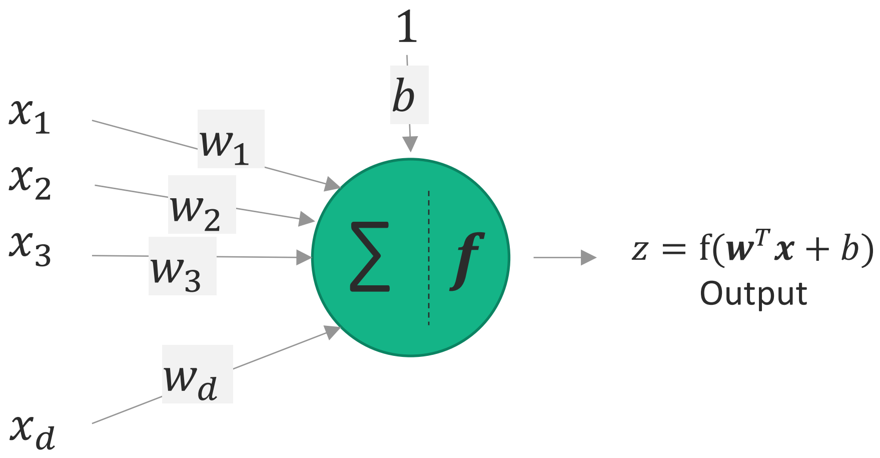
Optimizers

- Stochastic Gradient Decent (SGD)
- ADAM
- Nesterov’ Momentum
- Adagrad
- AdaDelta
- RPSProp
- etc.



Weights Initialization

How to initialize w s



Weights Initialization

Initialize all $W = 0$
All activations, loss, gradients will be zero!

Initialize all $W = constant$
Weights remain same per layers, not good, longer training time

Initialize all $W = random\ (Normal, Xavier, Uniform, \dots)$
Best option

Other Hyperparameters

- Network architecture (e.g., number of layers, number of neurons)
- Number of iterations (epochs)
- Batch size

Common Mistakes

- Forget to toggle **train/eval** mode for the network
- Use test data set for hyperparameter tuning!
- Forget to normalize data
- Forget to set dropout **probability**
- Forget to call **.zero_grad()** before calling **.backward()**
- Passed **softmax** outputs to a loss function that expects raw logits (more on this later)
- Forget to use sigmoid function for binary classification and softmax for multi-label classification

early makers

em lyon business school