EVML

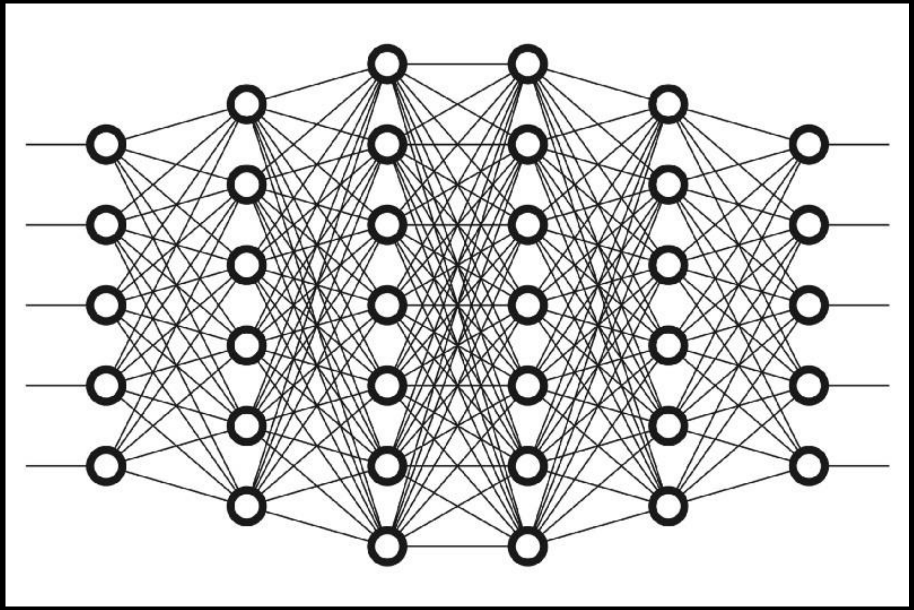
DEEP NEURAL NETWORKS

JEROEN VEEN

HAN_UNIVERSITY
OF APPLIED SCIENCES

CONTENTS

- Recap ANN and example
- Vanishing and exploding gradients
- Transfer learning
- Training optimization
- Learning rate scheduling
- Regularization



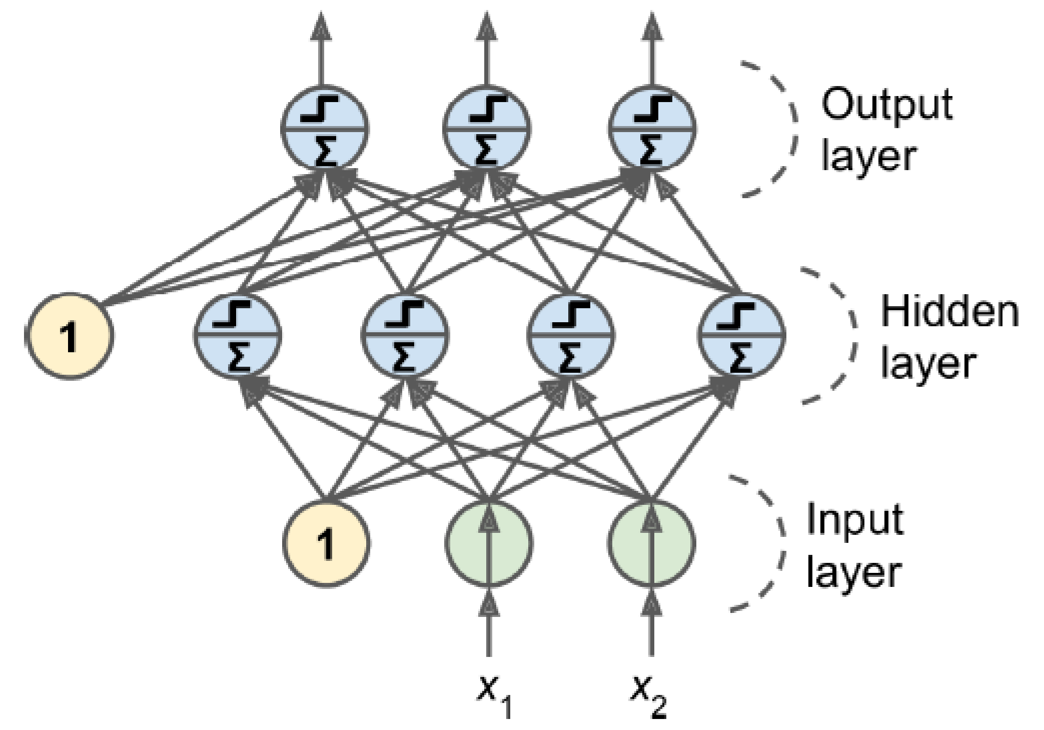
Use: https://alexlenail.me/NN-SVG/index.html
to draw nice maps

HAN_UNIVERSITY
OF APPLIED SCIENCES

AGENDA

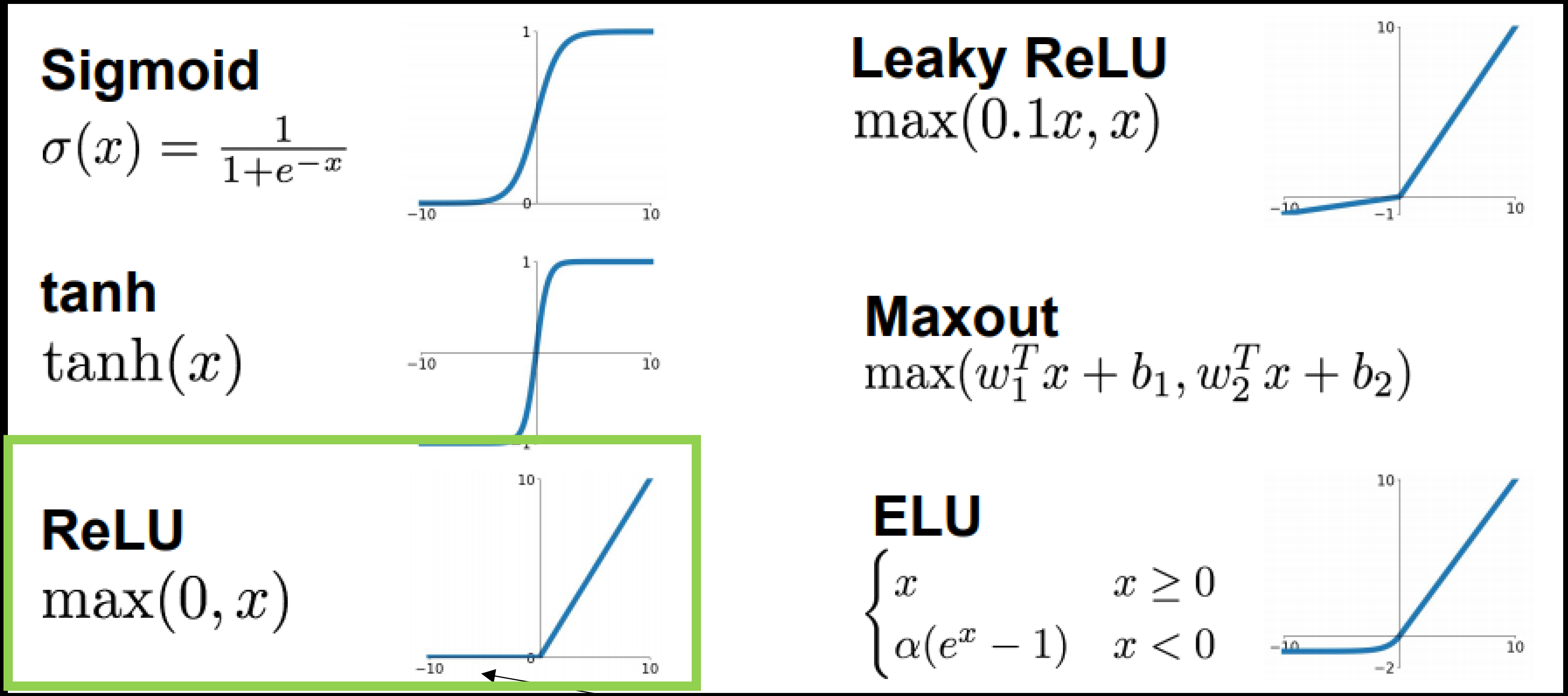
- DL portfolio walkthrough
- Tensorboard, visualizing the training process
- Data augmentation
- Storing and loading models
- Experimental route to optimize architectures
- Fine-tuning neural network hyperparameters

RECALL ANNS



Source: Géron, ISBN: 9781492032632

ACTIVATION FUNCTIONS

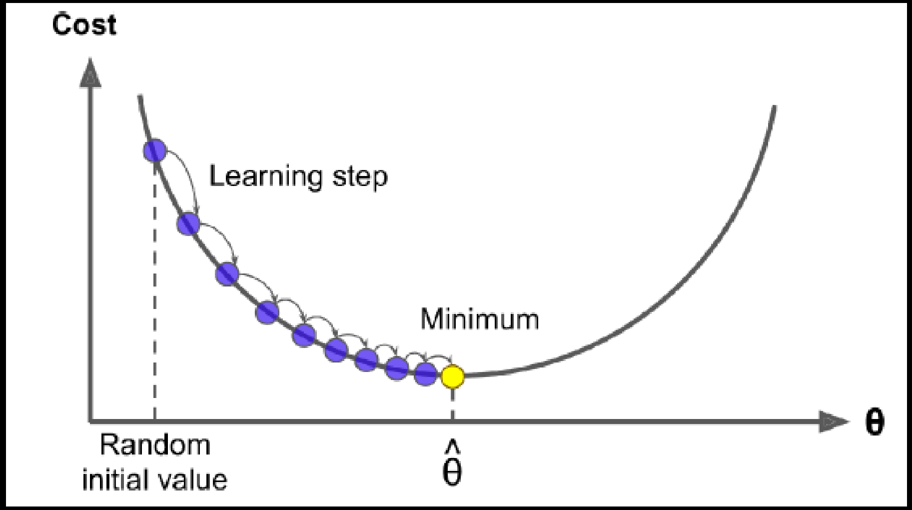


Source: http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture6.pdf

TRAINING

- Multi-dimensional optimization problem

- Gradient descent



Learning rate

$$w_{i,j}^{(\text{next step})} = w_{i,j} + \eta\boxed{\left(y_j - \hat{y}_j\right)}x_i$$

Input value

Connection weights

error

EXERCISE: BASIC IMAGE CLASSIFICATION

See tf_quickstart.py

- Train a shallow net, see also Géron 10.2.1

- Trouble downloading the datasets? Let me know

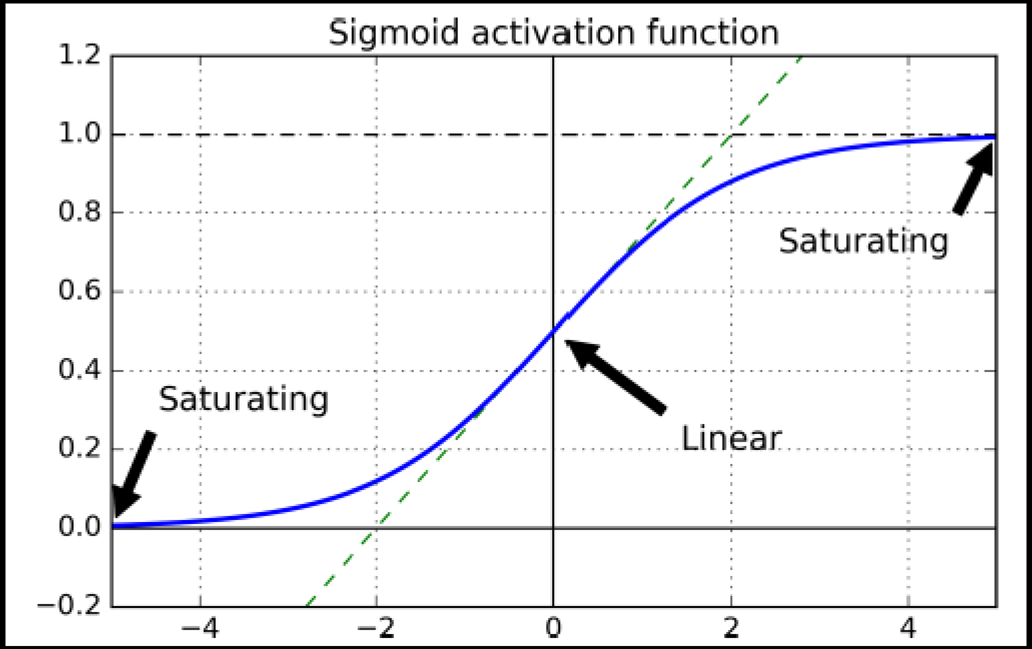
- Training can take quite some time…
  - >> Use a small number of weights in the hidden layers
  - >>Work on colab.research.google.com

- Train and validate

> Instead of passing a validation set using the `validation_data` argument, you could set `validation_split` to the ratio of the training set that you want Keras to use for validation. For example, `validation_split=0.1` tells Keras to use the last 10% of the data (before shuffling) for validation.

UNSTABLE GRADIENTS

- https://www.youtube.com/watch?v=qO_NLVjD6zE&t=105s
- Variance of the outputs > variance inputs leads to saturation



SOLUTIONS TO SPEED UP TRAINING

- Unstable gradients results in very slow training

- Smart weight initialization
- Non-saturating activation function
- Batch normalization

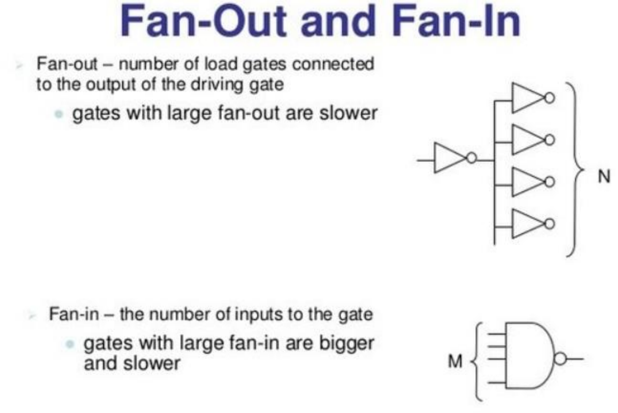
GLOROT AND HE INITIALIZATION

- Signals need to flow properly in both directions

- fan_{in} = number of inputs
- fan_{out} = number of neurons

Table 11-1. Initialization parameters for each type of activation function

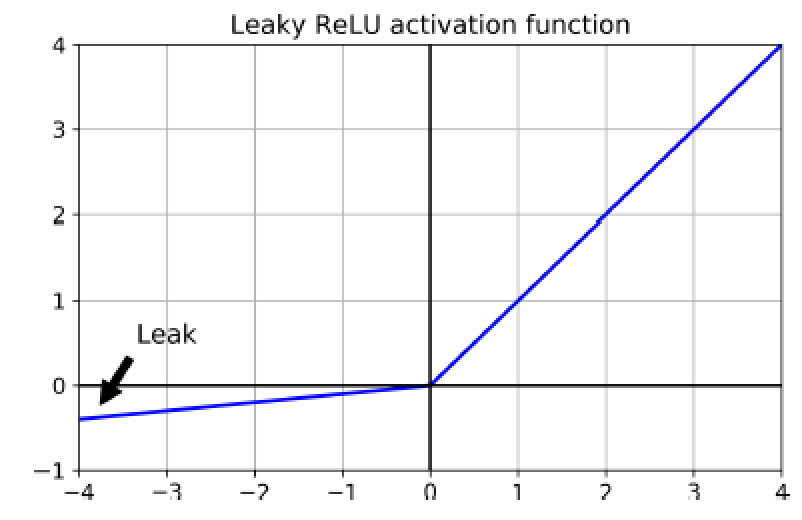
Initialization	Activation functions	σ^2 (Normal)
Glorot	None, tanh, logistic, softmax	$1 / fan_{avg}$
He	ReLU and variants	$2 / fan_{in}$
LeCun	SELU	$1 / fan_{in}$

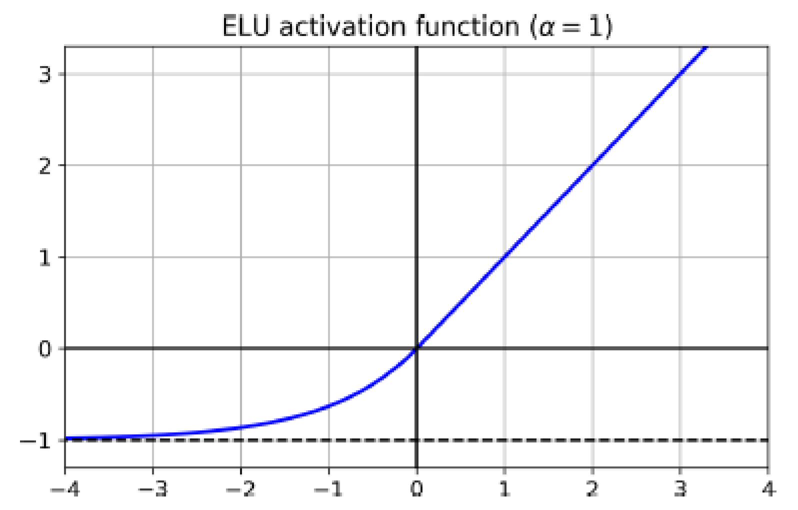


- See also https://www.youtube.com/watch?v=8krd5qKVw-Q&t=300

NONSATURATING ACTIVATION FUNCTIONS

- Standard ReLU may lead to ‘dying’ neurons
- ReLU variants: leaky ReLU, PReLU, ELU, SELU



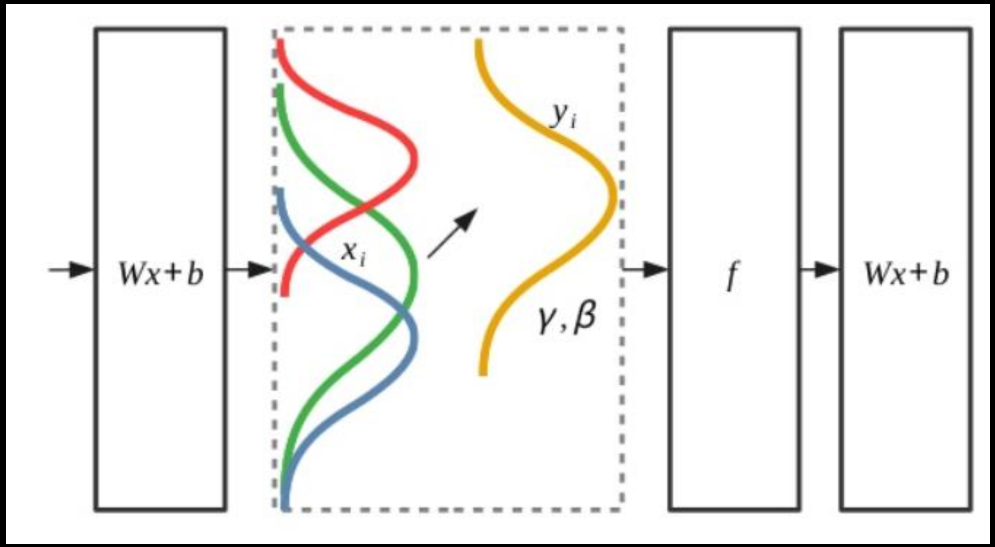


Source: Géron, ISBN: 9781492032632

HAN_UNIVERSITY OF APPLIED SCIENCES

BATCH NORMALIZATION

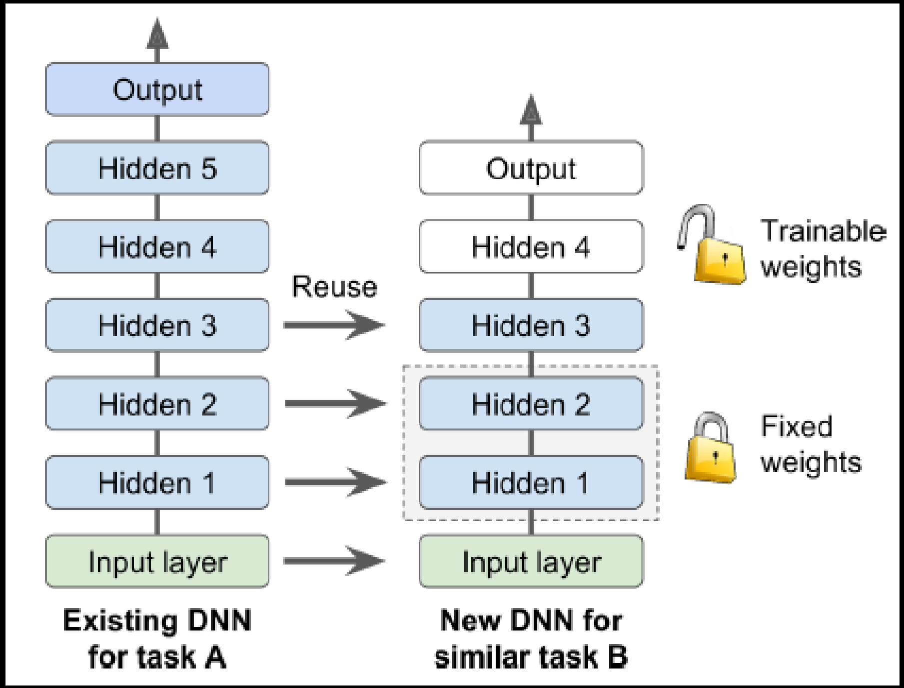
- Mitigate both vanishing and exploding gradients
- Normalize each hidden layer's input during training



- Input mean and std (μ,σ), output scale and offset (γ,β) parameters are learned over entire batch

TRANSFER LEARNING

- Reuse pretrained (lower) layers to speed up training, requiring significantly less data
- Works best when the inputs have similar low-level features



gradual unfreezing from the top

iterate until right number of layers

Source: Géron, ISBN: 9781492032632

TRAINING OPTIMIZATION

- Cost functions / Loss functions
  measure of correctness of a prediction

- e.g. mean squared error
  cross entropy
  log loss

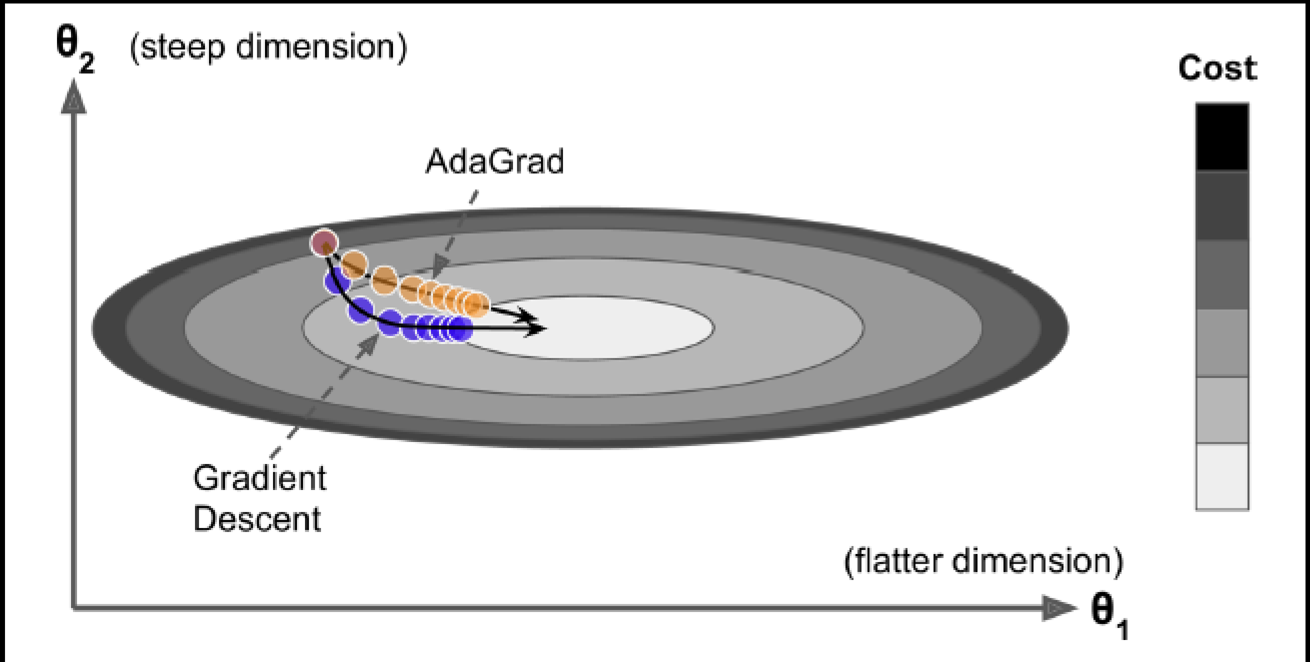
a measure of dissimilarity between the ground truth label probability and the predicted probability of the label

- Descending the error curve, feedback on error

- Different kinds of optimization : gradient descent, stochastic gradient descent, adagrad, adam, etc.

FASTER OPTIMIZERS

- Adaptive learning rate algorithms
- https://www.youtube.com/watch?v=mdKjMPmcWjY&t=133s



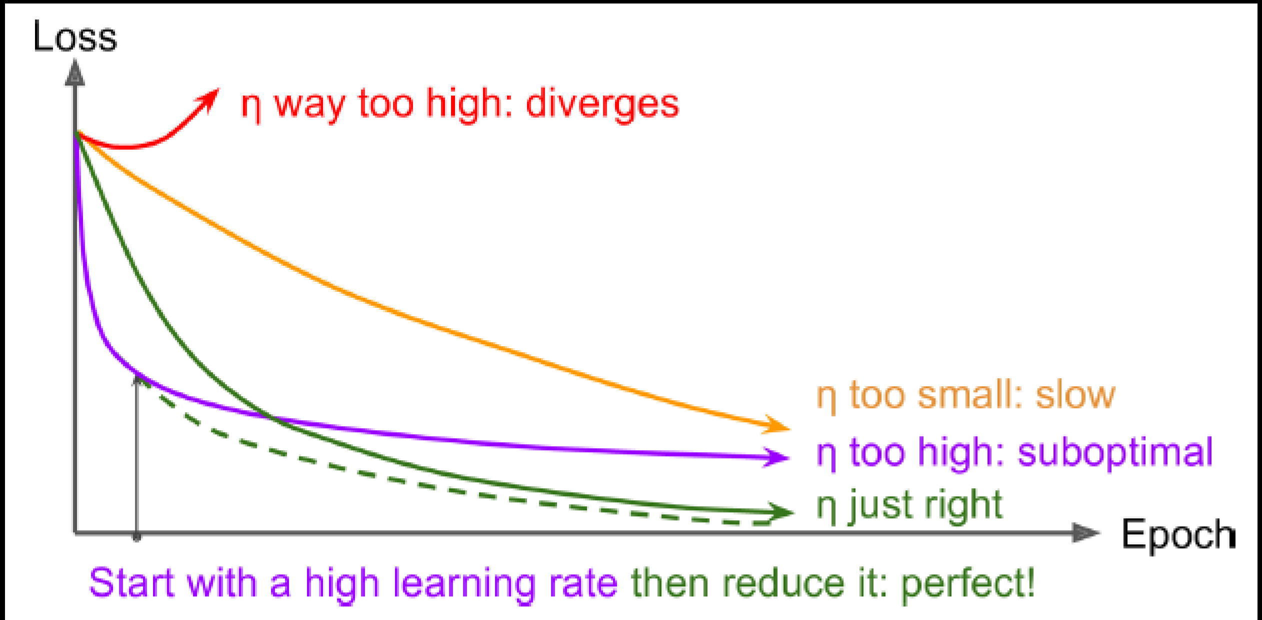
Source: Géron, ISBN: 9781492032632

OPTIMIZER COMPARISON

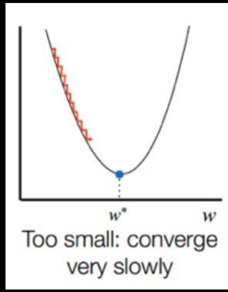
Class	Convergence speed	Convergence quality
`SGD`	*	***
`SGD(momentum=...)`	**	***
`SGD(momentum=..., nesterov=True)`	**	***
`Adagrad`	***	* (stops too early)
`RMSprop`	***	** or ***
`Adam`	***	** or ***
`Nadam`	***	** or ***
`AdaMax`	***	** or ***

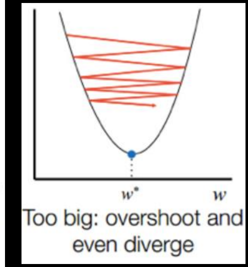
OPTIMIZING CONSTANT LEARNING RATE

- Compare learning curves for various rates and pick optimal, but constant rate



Source: Geron, ISBN: 9781492032632





Source: https://www.jeremyjordan.me/nn-learning-rate/

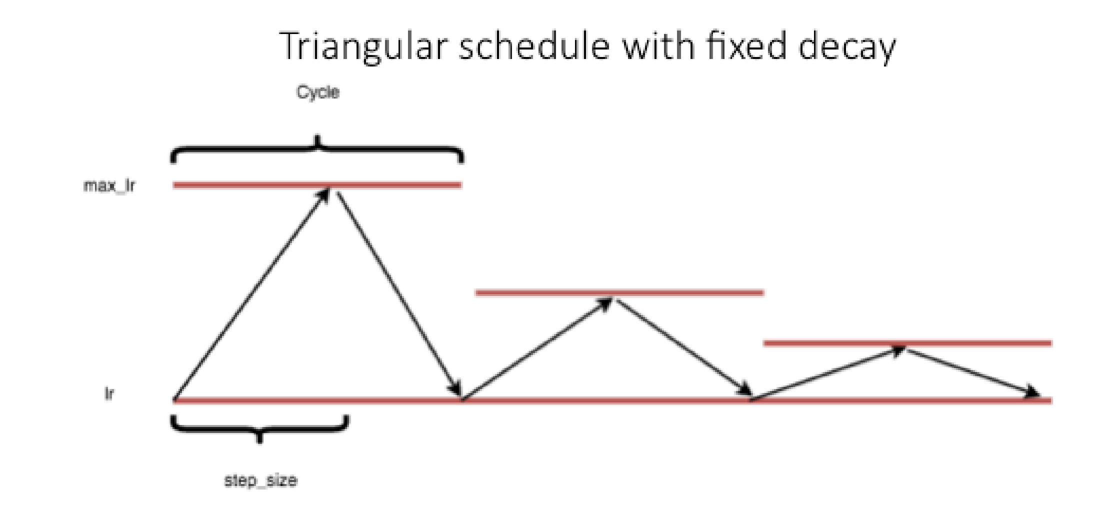
LEARNING RATE SCHEDULING

- Strategies to adapt learning rate while training progresses.

- Power, exponential, piecewise constant scheduling:
  drop learning rate every iteration, e.g.

- Performance scheduling:
  reduce learning rate based on error

- (1cycle) scheduling



REGULARIZATION

- ℓ1 regularization: sparse model
- ℓ2 regularization: constrain weights
- (MC) dropout: ignore neurons during training
- Max-Norm regularization

L_2 and L_1 penalize weights differently:

- L_2 penalizes *weight*^{2}.
- L_1 penalizes |*weight*|.