LING 506 - TOPICS IN COMPUTATIONAL LINGUISTICS

Introductory Machine Learning

Yan Tang

Department of Linguistics, UIUC

Week 15

Last week…

- The history and development of Artificial Neuron Networks (ANN)

- The perceptron: one of the simplest ANN architecture
  - The structure of perceptron
  - The Threshold Linear Unit (TLU)
  - The output is a sum of the weighed inputs, processed by a Heaviside step function
  - Training

Last week…

- The Multilayer Perceptron
  - Solved the 'XOR' issue that the classical Perceptron cannot deal with
  - Complexity: hidden layers; deep neural network (DNN)
  - Feedforward Neural Network
- Backpropagation: an efficient algorithm for computing gradient through the network
- Activation functions: non-linear transformation
  - The sigmoid (logistic) function
  - The tangent sigmoid (hyperbolic tangent) function
  - The Rectified Linear Unit function (ReLU)

Tuning ANN

- The flexibility of ANN:
  - + Gives more control on the model behaviour and complexity
  - - Involves many hyperparameters
- Common hyperparameters in ANN architecture:
  - Number of hidden layers
  - Number of neurons / layer
  - Learning rate
  - Activation function for each layer
  - Weight initialisation
  - Number of epochs
  - …

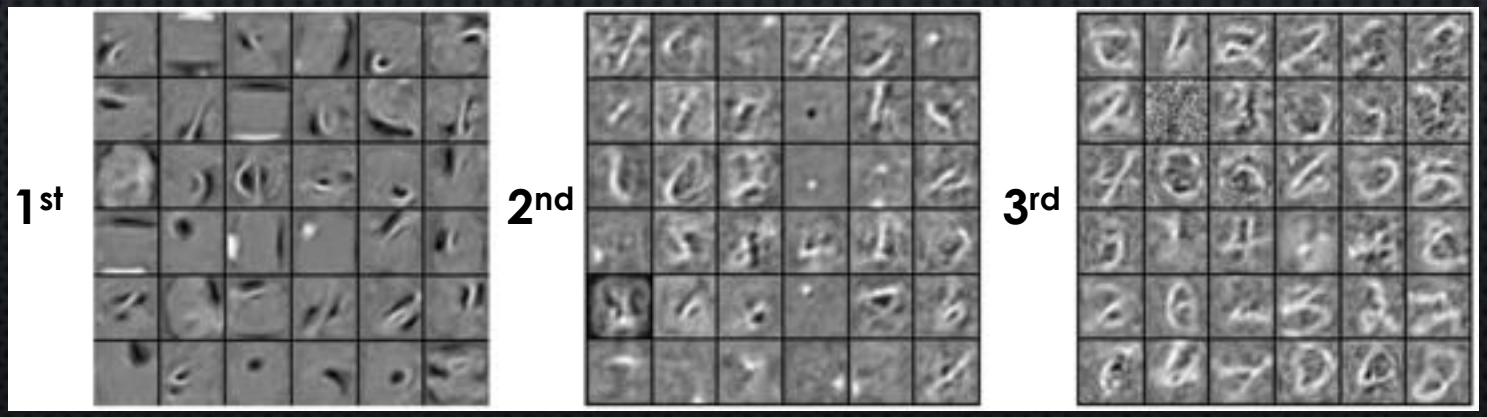
Number of hidden layers

- One single hidden layer can lead to reasonable results for many problems
  - In theory, a single hidden layer MLP can model highly complex problems *with enough neurons*
- Deep ANN structures offers better *parameter efficiency*
  - Use exponentially fewer neurons than shallow networks
  - Achieve better model performance with same amount of training data

Number of hidden layers

The general functions of hidden layers at different stages:

- Lower hidden layers: model low-level structures
- Mid hidden layers: combine low-level structures to model intermediate-level structures
- Higher hidden layers: combine intermediate-level structures with output layer to model high-level structure, i.e. the output



Erhan et al, 2009

Number of hidden layers

- The advantages of the hierarchical structure of DNN:
  - Fast convergence
  - Improve the model's generalisability
  - Reuse of lower layers – transfer learning

- General practice on the choice of No. of hidden layers
  - Start with one or two layers
  - Increase the layer until overfitting starts
  - Reuse parts from trained structures with good performance for similar task

Number of neurons in a hidden layer

- The number of neurons in the input layer is determined by the number of input features

- The number of neurons in the output layer depends on the problem:
  - Regression: one / dimension
  - Classification: one for binary, one/class for multiclass, one/label for multilabel binary.

Number of neurons in a hidden layer

- A Pyramid structure:
  - More neurons in lower layer; fewer and fewer in high layers.
  - Rationale: low-level high-dimensional features can be combined into fewer low-dimensional high-level features
- A rectangle structure:
  - Same number of neurons in *all* hidden layers
  - Results in similar or even better performance
  - Simplifies turning to only one parameter, i.e. number of neurons for *all layers* instead of *per layer*

Number of neurons in a hidden layer

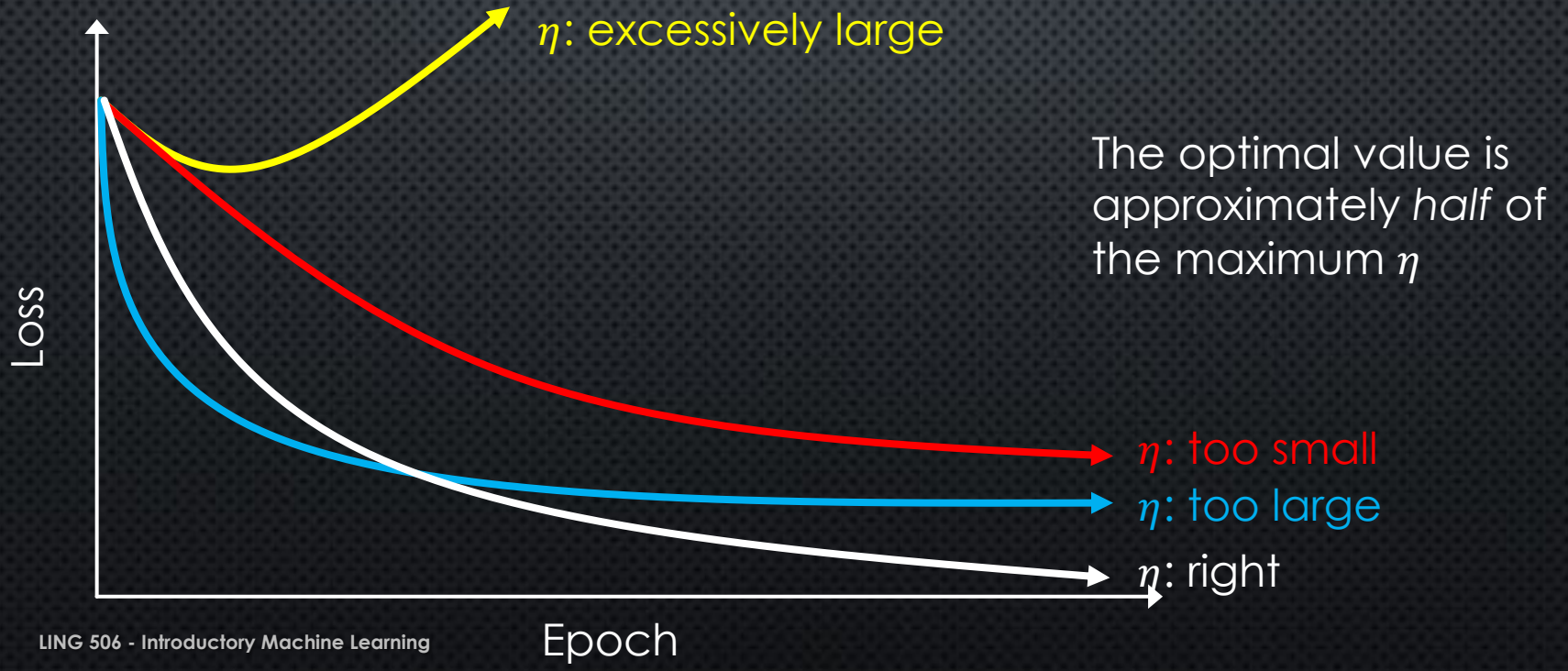
- Similar to choosing number of hidden layers, increase the number of neurons until overfitting occurs

- A simpler and more efficient approach:
  - Use more layers and neurons than the problem requires
  - Limit overfitting using early stopping and regularisation, e.g. bigger value for alpha
  - Help with avoiding bottleneck layers

- In general, the effect of increasing the number of layers is strong than increasing the number of neurons/layer

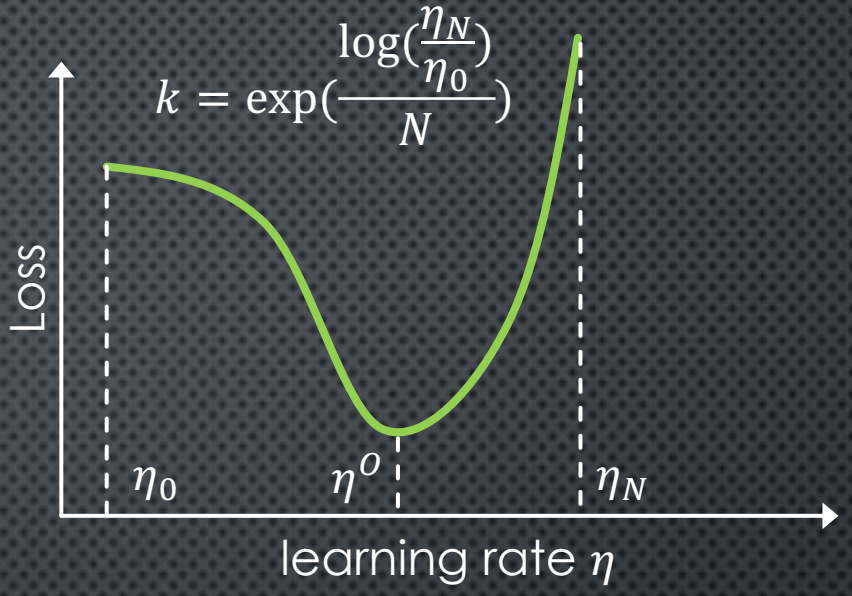
Learning rate

- One of the most important hyperparameters in ANN
  - η too large: model diverges or not converges to the optimal
  - η too small: model takes long to converge



Learning rate

An empirical solution seeking for a good learning rate:



- Choose a rather small η_0 (e.g. 10-6) and train the model for N epochs (e.g. 300).
- Increase η by multiplying a constant k after each epoch until a large value η_N (e.g. 10) in N epochs
- Plot the loss as a function of η. If theη_0, η_N and N are set appropriately, a dip is expected on the curve
- The optimal η^o should be somewhat lower than the reflection point (i.e. the lowest at the dip)

Learning scheduling

- Learning schedules: increase or decrease the learning rate adaptively
  - More computationally efficient than the constant optimal η

- Some learning scheduling:
  - Power scheduling: $\eta(n) = \frac{\eta_0}{(1+\frac{n}{s})^p}$
  - Exponential scheduling: $\eta(n) = \eta_0 0.1^{n/s}$
  - Piecewise constant scheduling
  - Performance scheduling
  -

Optimiser/Solvers

A good optimiser can significantly expedite the training

- Momentum optimisation
  - A speedy version of regular Gradient Descent with a momentum vector
- AdaGrad: Adaptive Subgradient methods
  - Corrects Gradient Descent's moving direction towards the global optimum
- RMSprop
  - Improve AdaGrad by accumulating only the gradients from the most recent iterations

Optimiser/Solvers

- Adam: Adaptive moment estimation
  - A combination of momentum optimisation and RMSprop
  - An adaptive learning rate algorithm
  - Requires less tuning on the learning rate η
- L-BFGS: Limited memory BFGS
  - A quasi-Newton method
  - Does not use mini-batch

Batch size

- The size of minibatches for stochastic optimisers
- Small batch size:
  - Leads to models with better generalisability in less time
  - Size: 2 to 32 samples (Masters and Luschi, 2018) or others
- Large batch size
  - The algorithm gets to see more sample per epoch; use hardware accelerators for more efficient computing
  - Size: as many samples as the memory can hold
  - Training instabilities; the model may not generalise well
- General strategy: use large batch size with learning rate warmup (Goyal et al., 2017). If performance is unstable, reduce the batch size

Other hyperparameters

- Activation function:
  - The ReLU activation function is a good default for all hidden layers.
  - For output layers:
    - Regress: ReLU; Sigmoid or tangent Sigmoid for bounded outputs
    - Classification: Sigmoid for binary problems; Softmax for multiclass classification
- Number of epochs:
  - No need to optimise as long as it is greater than when the early stopping takes place (if enabled)

Tuning ANN:

- Simple approach: try several different combinations of hyperparameters
  - Grid Search with cross-validation
  - Randomised Search with cross-validation
- When the search space is large, randomised search may be more efficient than grid search
  - Still time consuming
  - The lack of focused search area

Tuning ANN: guided randomised search

- Run a quick randomised search on a large search space(s) of hyperparameter values
- Locate the best values for hyperparameters
- Use the values as the centres to run another randomised search for a smaller range around the centres
- Recursively perform step 2 and 3

Tuning ANN: other tuning libraries

- Many other techniques are available for more efficient and accurate space searching
- Principle: exploit a region of the space where proves relatively probable for optimal value(s)
- Some Python libraries:
  - Hyperopt
  - Scikit-Optimize
  - Sklearn-Deap
  - Spearmint
  - Hyperband

Tuning ANN: the future

- Hyperparameter tuning is an active area of research
  - Evolutionary algorithms might lead to unmanned construction of fine-tuned ANN
- Population-based training of ANNs (Jaderberg et al., 2017)
- Google's attempt on searching for hyperparameters and the best ANN architecture
  - Cloud AutoML
- Deep Neuroevolution: train individual ANNs using evolutionary algorithms in place of Gradient Descent (Li et al., 2017)