LESSON 2: NEURAL NETWORKS

Savannah Thais

Intro to Machine Learning

Princeton Wintersession 2021

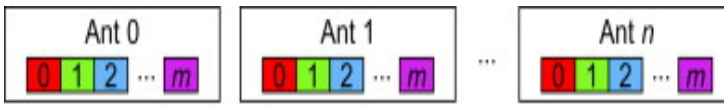
01/26/2021

Outline for Today

- Quick Review
- In-between Algorithm: Decision Trees
- Conceptual + Mathematical Foundation of NNs
- NN Training Considerations
- Coding Exercises

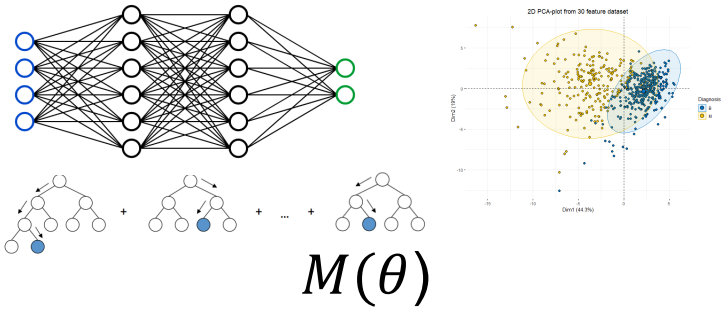
Training A Supervised Model

Training data



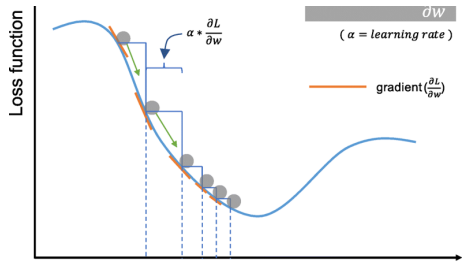
$$D_{train}=(\vec{x}, y)$$

Choose a Model



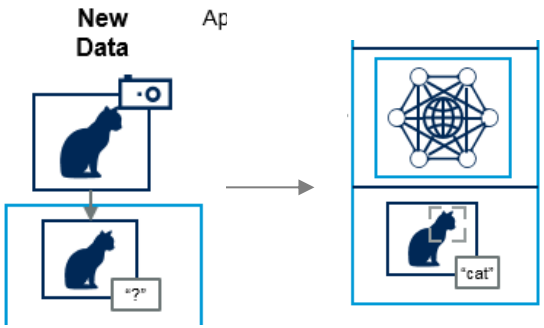
$$M(\theta)$$

Optimize Parameters



Loss function: L

Make an Inference!



Knowledge Check: Key Terms

- Supervised Learning:
- Unsupervised Learning:
- Loss Function:
- Model Parameter:
- How should you split your training data?:
- Inference:
- Overfitting/underfitting:

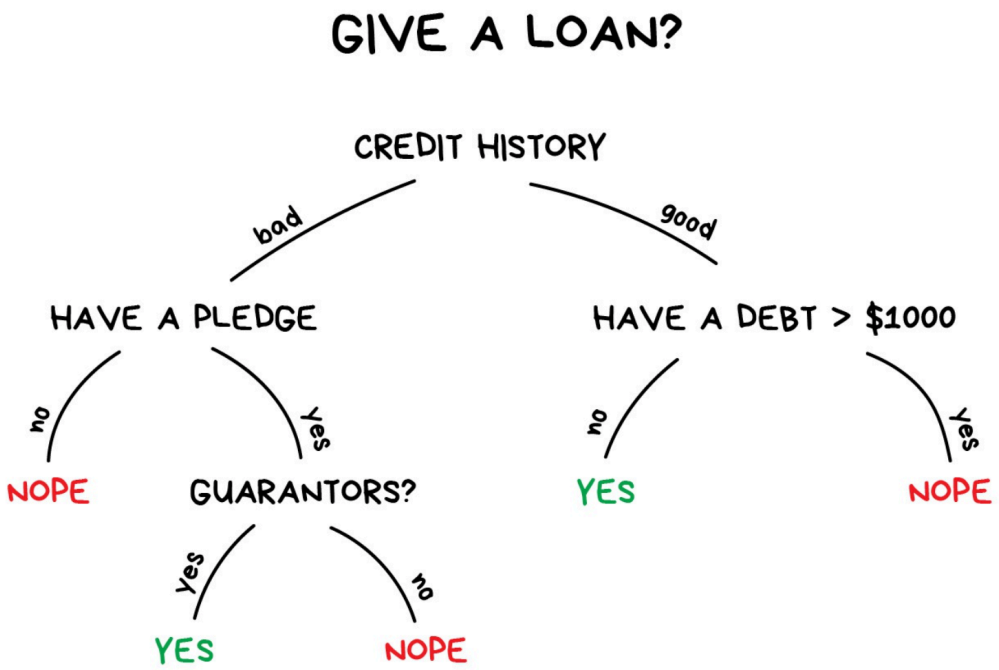
Knowledge Check: Key Terms

- Supervised Learning: algorithm provided with labels during training
  - Example: building a classifier to flag spam emails by training on examples of normal and spam emails
- Unsupervised Learning: algorithm tries to learn underlying structure in the data (no labels)
  - Example: separating groups of similar fruits
- Loss Function: quantifies how well the current model is achieving the learning objective
  - Example: mean squared error for a regression algorithm
- Model Parameter: a learnable component of a model (loss function is evaluated at a specific set of model parameters)
  - Example: the slope parameter in a linear regression
- How should you split your training data?: training (used by the model to adjust model parameters), validation (used to check for over-fitting and measure model performance during development), test (used to quantify final model performance)
- Inference: Using a trained model to make a conclusion about a new, unseen data point
- Overfitting/underfitting: learning too much about the specific training samples (over) or not learning enough about the data patterns (under)

Decision Trees

Decision Trees

Input data is processed through a series of linear cuts



1. Try a range of cuts on each feature
2. Select cut with lowest cost
3. Repeat 1+2 for all branchings
4. Meet stopping criteria
5. Assign a label to each final leaf

- No global loss function, uses 'greedy local metric'
- Pros: interpretable, easy to build
- Cons: assumes simple linear relationship, easy to overtrain, high variance

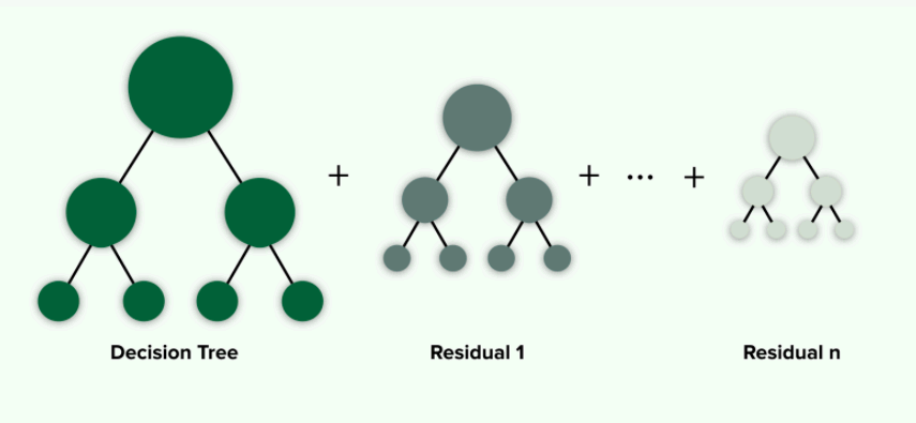
further reading

Forests of Trees

Multiple decisions trees are combined into ensemble classifier

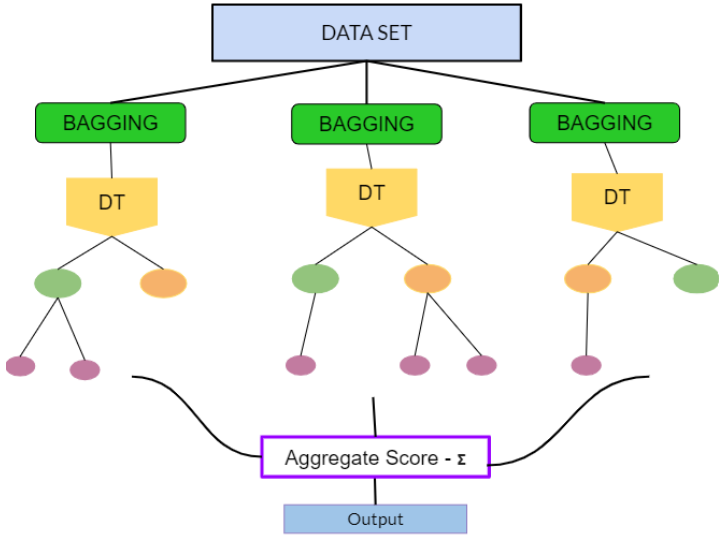
Boosting:

Misclassified data points are given a higher weight in the next tree



Bagging:

Each tree is trained on a random subset of data or input features

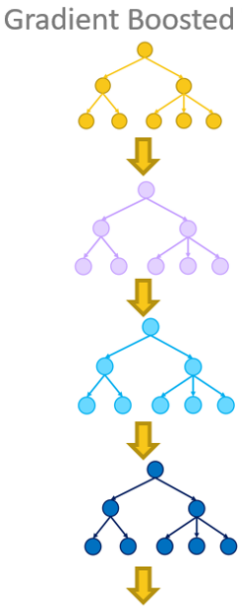


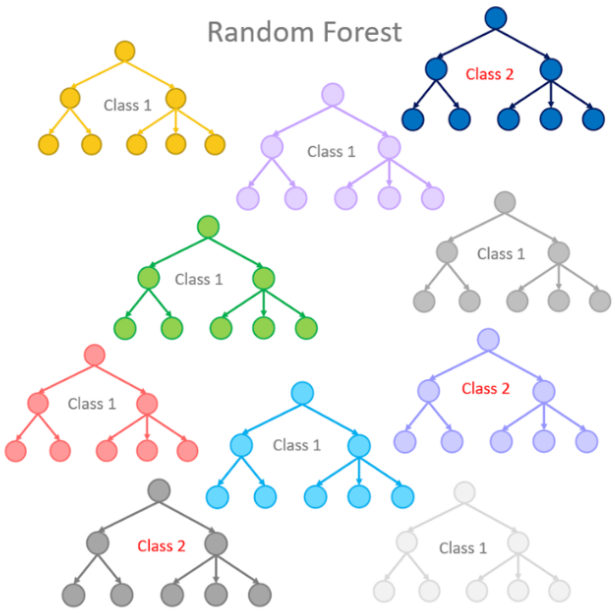
- Final label is typically (weighted) average of label from all trees
- Both methods reduce variance in final decision
- Boosting has a global loss function and can be optimized with gradient descent

Decision Tree Forests Model Card

Multiple decisions trees are combined into ensemble classifier

- Boosting: misclassified data points given a higher weight in next tree
- Bagging: train trees on random subsets of data/subsets of features





- Training: gradient descent (boosting) or greedy local (bagging)
- Classification: assign label to each final leaf in aggregate
- Loss Function:

$$h_m(x) = -\frac{\partial L_{\text{MSE}}}{\partial F} = y - F(x).$$

- Hyperparameters:
  - Depth and pruning
  - Combination function
  - Number of trees

further reading

Pros:

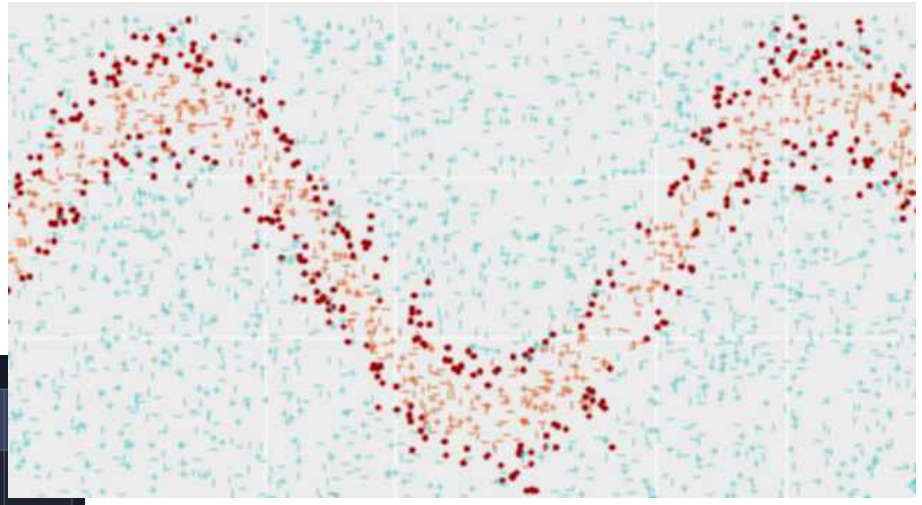
- ~Interpretable
- Reduced variance
- Allows high dimension data

Cons:

- Less precise regression
- Only allows linear discriminant

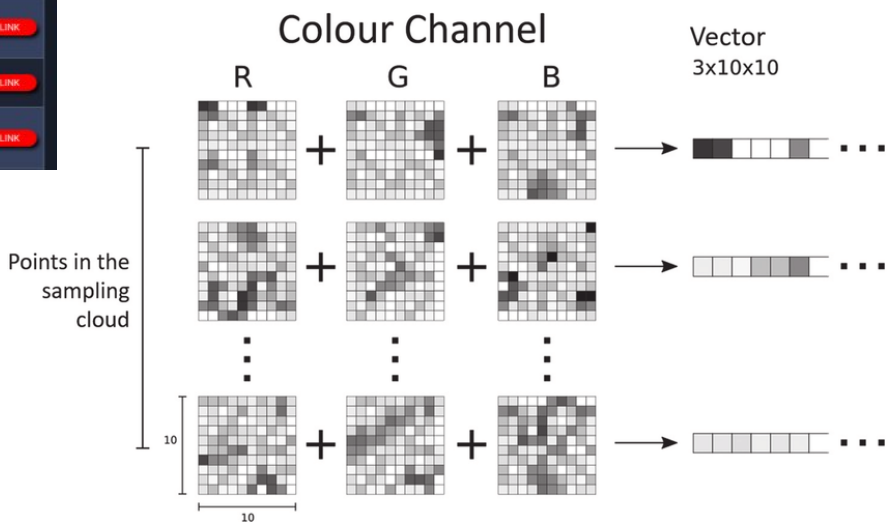
But What About…

Highly non-linear data?



Dataset	Added	Language	Description	Instances	Format	Task	Created	Creator	
A Conversational Question Answering Challenge (CoQA)	01.15.20	English	Dataset for measuring the ability of machines to understand a text passage and answer a series of interconnected questions that appear in a conversation.	127,000+	JSON	Question Answering, Reading Comprehension	2019	Redy et al.	
A Reading Comprehension Benchmark Requiring Discrete Reasoning Over Paragraphs (DROP)	01.15.20	English	Dataset is used to resolve references in a question, perhaps to multiple input positions, and perform discrete operations over them (such as addition, counting, or sorting).	96,000	JSON	Question Answering, Reading Comprehension	2019	Dua et al.	
ABC Australia News Corpus	01.15.20	English	Entire news corpus of ABC Australia from 2003 to 2017.	1,103,664	CSV	Clustering, Events, Sentiment Analysis	2017	Kulkarni	LINK
Activitynet-QA	01.15.20	English	Dataset contains 58,000 human-annotated QA pairs on 5,800 videos derived from the popular ActivityNet dataset. The dataset provides a benchmark for testing the performance of VideoQA models on long-term spatio-temporal.	58,000	JSON	Question Answering, Visual, Commonsense	2019	Yu et al.	LINK
AI2 Reasoning Challenge (ARC)	01.15.20	English	Dataset contains 7,787 genuine grade-school level, multiple-choice science questions.	7,787	JSON, CSV	Question Answering, Reading Comprehension	2018	Clark et al.	LINK
AI2 Science Questions Mercury	01.15.20	English	Dataset consists of questions used in student assessments across elementary and middle school grade levels. Includes questions with diagrams and without.	6,940	JSON, JPG	Reading Comprehension	2017	Allen Institute	LINK
AI2 Science Questions v2.1	01.15.20	English	Dataset consists of questions used in student assessments in the United States across elementary and middle school grade levels. Each question is 4-way multiple choice format and may or may not include a diagram element.	5,060	JSON, CSV	Question Answering, Reading Comprehension	2017	Allen Institute	LINK

Massive datasets?



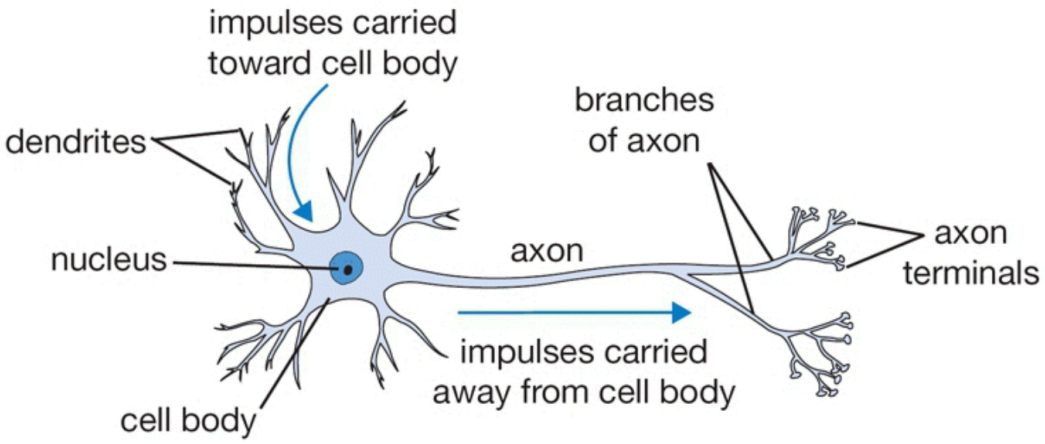
Data with many input features?

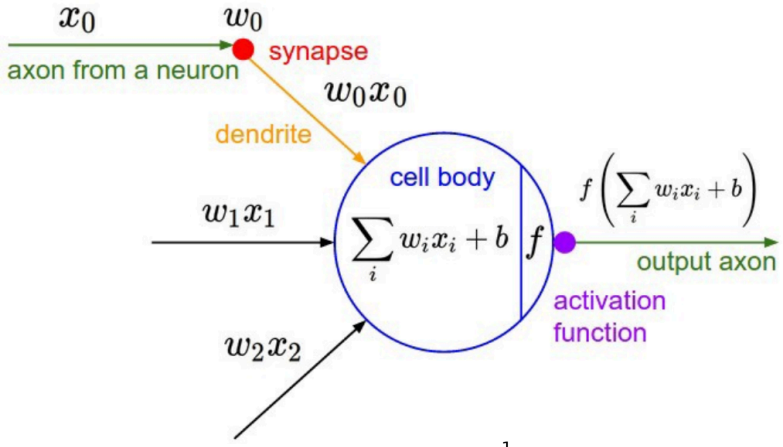
Neural Networks!

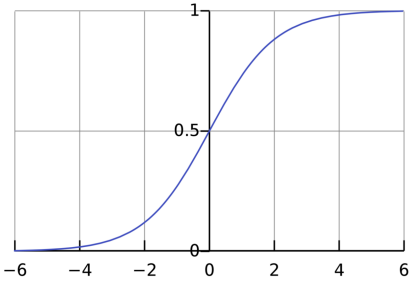
Neural Networks

Individual nodes based on biological neurons

- Input data transformed linearly
- Processed through non-linear activation function to determine if information is passed on

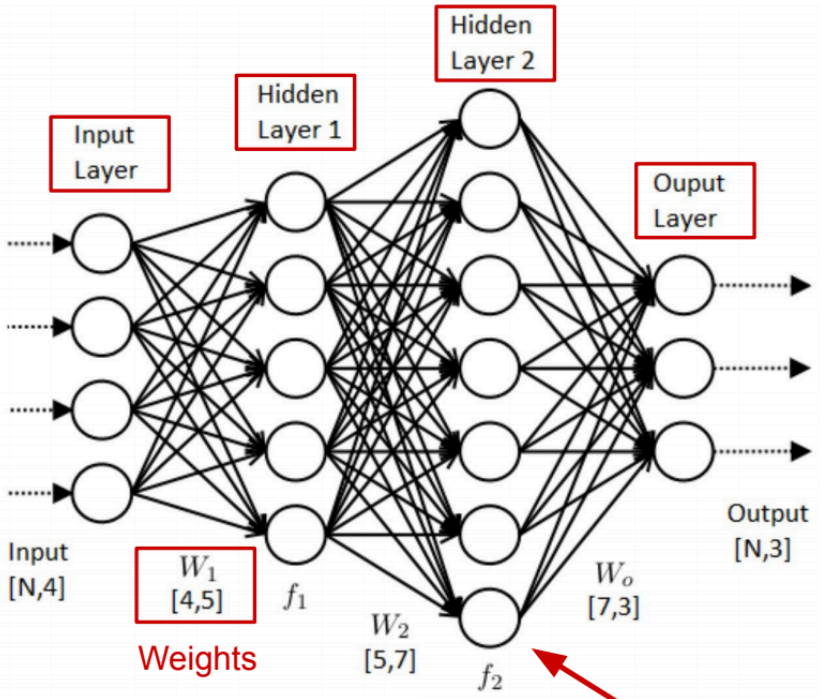






Network Math

- In practice, nodes are combined into layers
- Linear transformations between layers represented as matrices of weights W and biases b
- Each node calculates activation function of weighted sum of inputs to determine if information is passed on
- Extremely powerful dimensional transformation
  - Can efficiently train very deep networks to 'approximate any function'

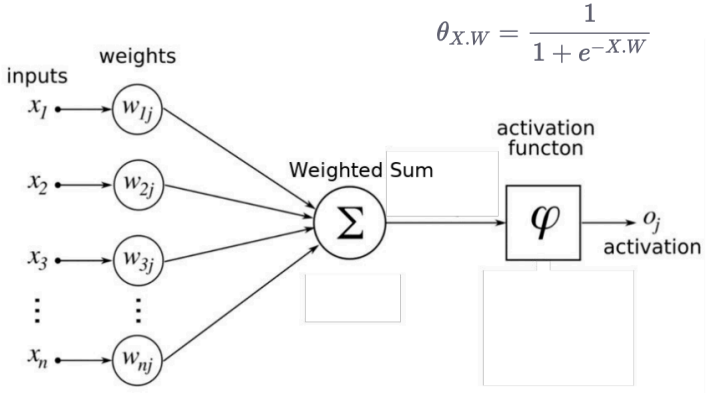


$$h_1 = \varphi(W_1 x + b_1)$$

$$h_2 = \varphi(W_2 h_1 + b_2)$$

$$\vdots$$

$$y = f(h_n)$$

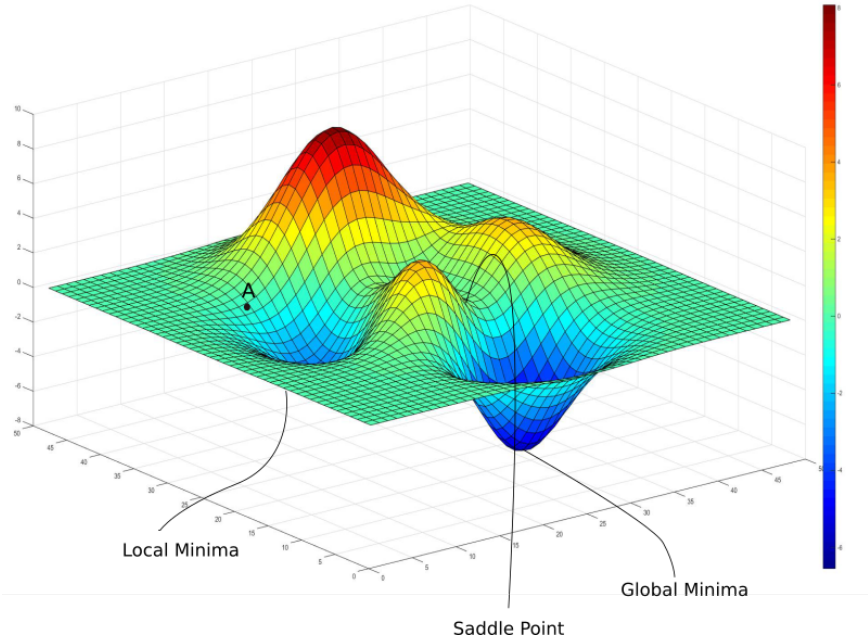


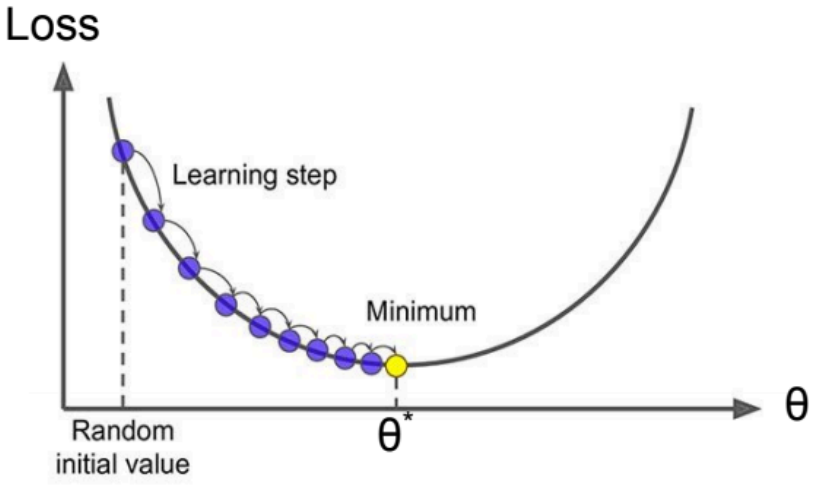
Gradient Descent

Intuition: Want to efficiently find the model parameters θ that minimize the loss function L

- We normally minimize things by evaluating the derivatives
  - This is the basis of gradient descent → your loss function must be differentiable!
  - But this is very difficult for complicated models with many parameters
- Basic procedure: at each training iteration (epoch) update model parameters to move evaluation point in opposite direction of gradient of loss function in direction of steepest ascent

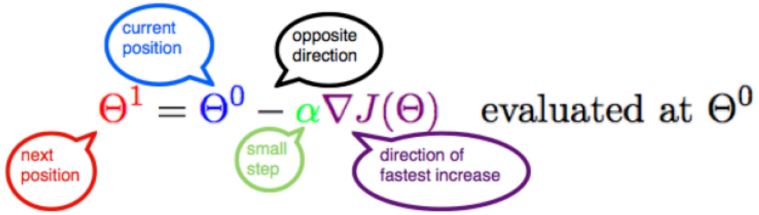
$$W_{k+1} \leftarrow W_k - \alpha \nabla L(W_k)$$

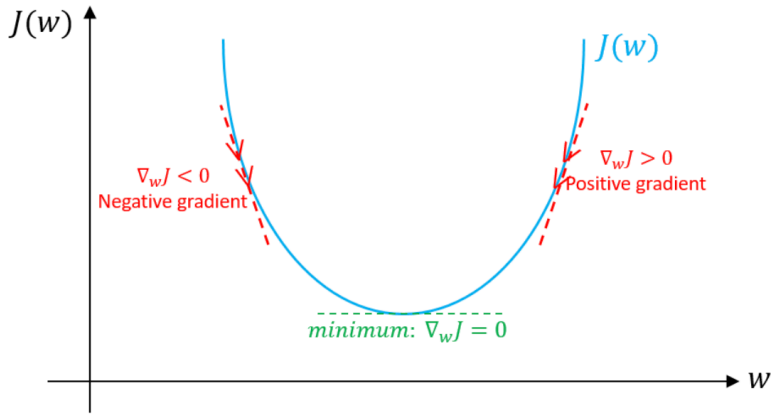


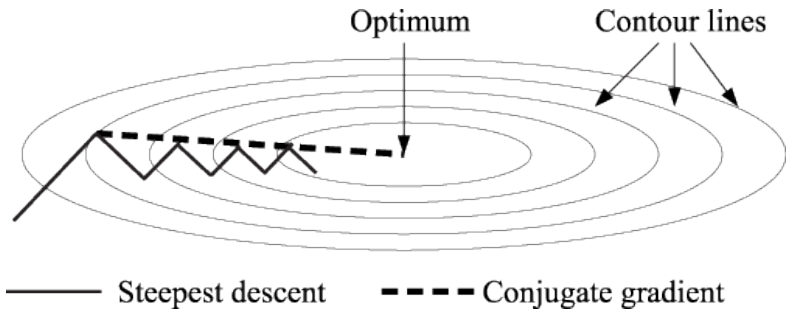


Gradient Descent

- Basic procedure: at each training iteration (epoch) update model parameters to move evaluation point in opposite direction of gradient of loss function in direction of steepest ascent
  - Essentially, finding the slope of the loss function with current model values and adjusting values to move loss function towards the minimum
  - Step/adjustment size is determined by learning rate α (hyperparameter)
- Note: or non-convex functions initial parameterization maters



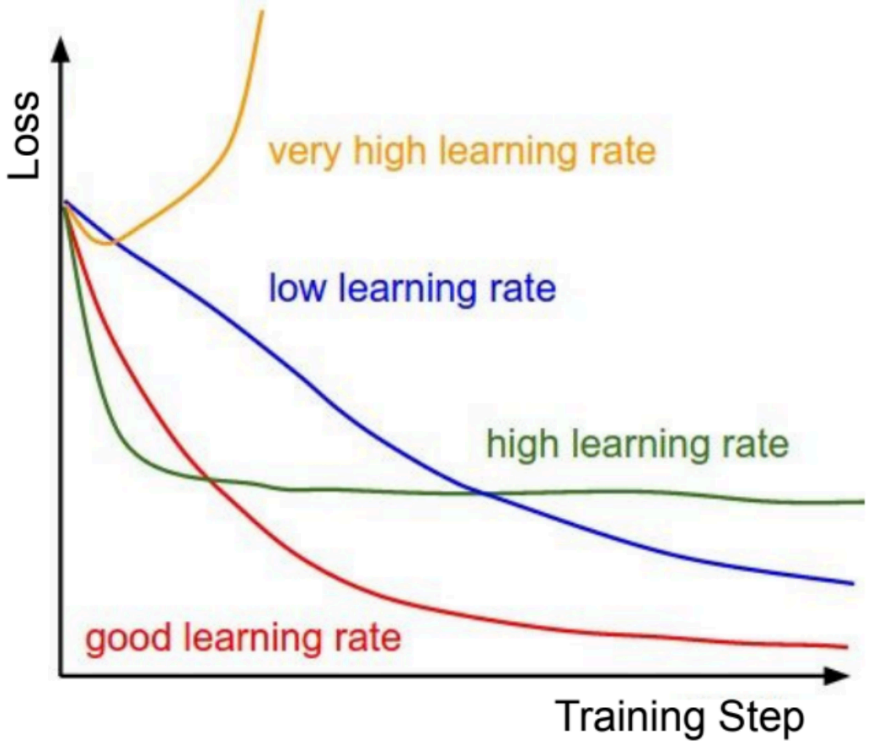


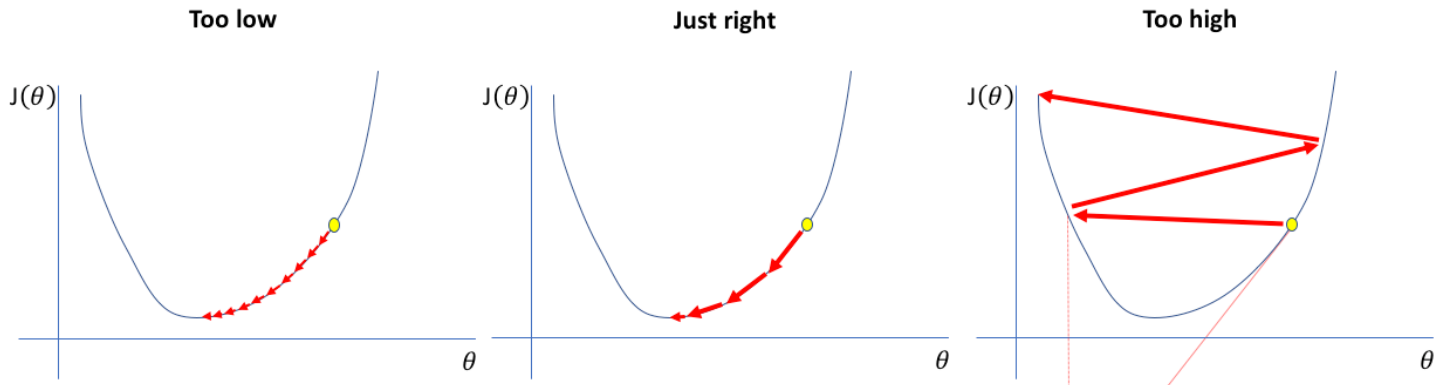


Full calculation example further reading

Learning Rate

- Choosing a suitable learning rate is critical for convergence and avoiding local minima
- But, no hard and fast rule for selection
  - This is a case for hyperparameter optimization





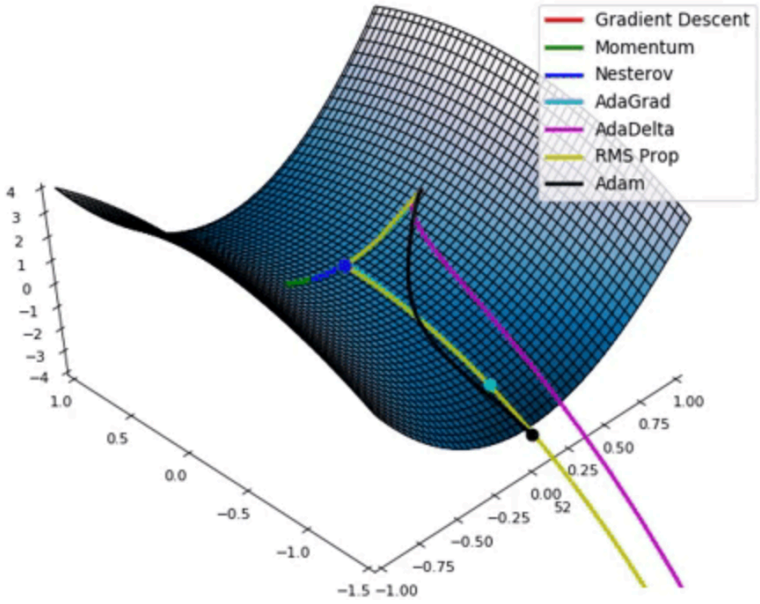
Gradient Descent Variants

There are several algorithms to speed up gradient descent and avoid local minima

- Stochastic GD: calculate gradient over a (random) minibatch of examples m
- Adagrad: adjust learning rate to individual features
- Momentum: add a fraction γ of previous gradient to update vector
- RMSprop: accumulate gradient only for a select window
- Adam: accumulate past gradients and their squares

Gradient estimate

$$W_{k+1} \leftarrow W_k - \alpha \frac{1}{m} \sum_{i=1}^{m} \nabla L(x_i; W_k)$$



further reading

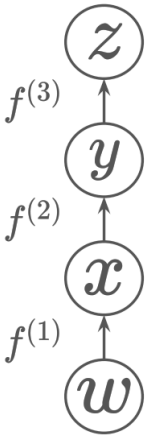
Back Propagation

Updates to individual network parameters are propagated from the cost function through network using chain-rule

$$\frac{\partial C}{\partial w_{jk}^l} = \frac{\partial C}{\partial z_j^l}\frac{\partial z_j^l}{\partial w_{jk}^l} \qquad chain\ rule$$

$$z_j^l = \sum_{k=1}^{m} w_{jk}^l a_k^{l-1} + b_j^l \qquad by\ definition$$

$$m\ -\ number\ of\ neurons\ in\ l-1\ layer$$



$$\frac{\partial z}{\partial w} = \frac{\partial z}{\partial y}\frac{\partial y}{\partial x}\frac{\partial x}{\partial w} = f^{(3)\prime}(y) f^{(2)\prime}(x) f^{(1)\prime}(w)$$

$$\frac{\partial z_j^l}{\partial w_{jk}^l} = a_k^{l-1} \qquad by\ differentiation\ (calculating\ derivative)$$

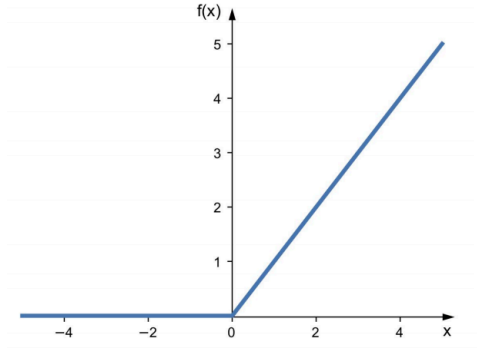
$$\frac{\partial C}{\partial w_{jk}^l} = \frac{\partial C}{\partial z_j^l} a_k^{l-1} \qquad final\ value$$

- The activation functions must also be differentiable
- If any intermediate activations have small or 0 derivatives the gradients will not flow back (vanishing gradient problem)

further reading

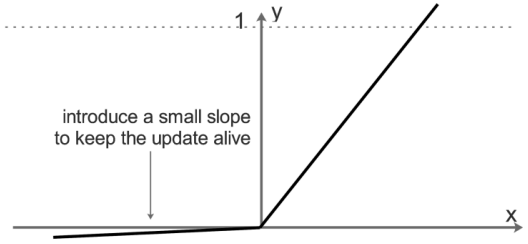
Activation Functions

Typically used in hidden layers



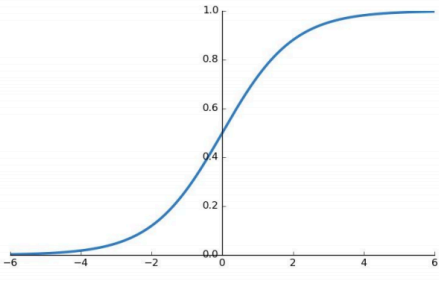
- Always non-negative
- Computationally cheap
- Passes strong gradients for x > 0
- Dies for x < 0 → leads to neurons with sparse activity

$$\mathrm{ReLU}(x) = \max(0, x)$$



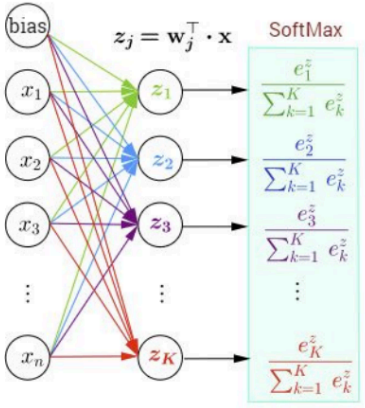
$$\text{Leaky ReLU}(x) = \max(\alpha x, x)$$
$$0 < \alpha < 1$$

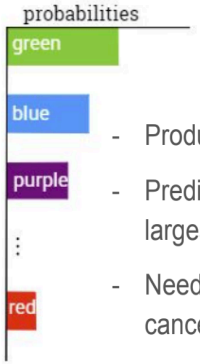
Typically used in output layers



- Bounded between 0 and 1
- Useful to squash layer output to represent binary probability → Bernoulli output distribution
- Expensive to compute
- Saturates at low and high input values → small slopes → low gradient signal → needs a **Log** in the loss function to cancel the effect of the **Exp**

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





- Produces a distribution over classes
- Predicted class is the one with the largest probability
- Needs a **Log** in the loss function to cancel the **Exp**

$$\mathrm{softmax}(\mathbf{z})_i = \frac{e^{z_i}}{\sum_j e^{z_j}}$$

source

The Universal Approximation Theorem

"A single hidden layer neural network with a linear output unit can approximate any continuous function arbitrarily well, given enough hidden units" – Hornik, 1991

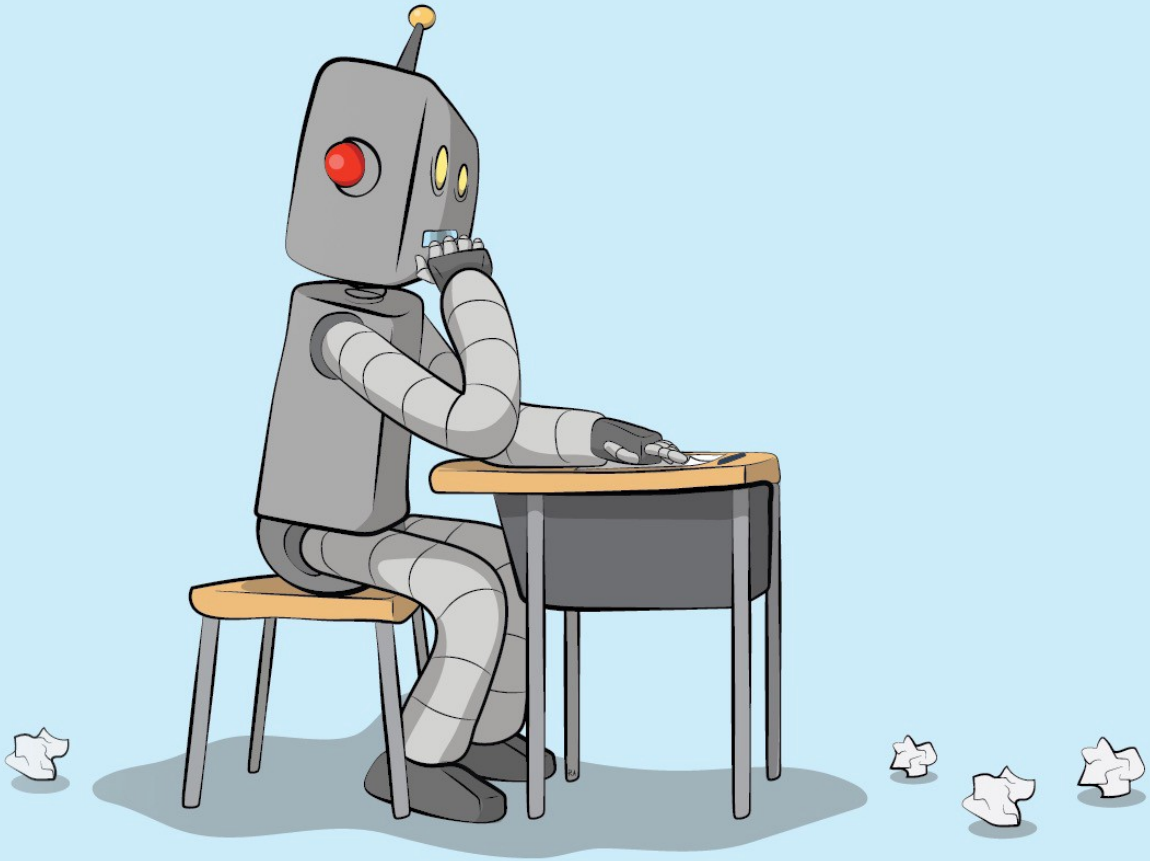
$$n_1(x) = Relu(-5x - 7.7)$$
$$n_2(x) = Relu(-1.2x - 1.3)$$
$$n_3(x) = Relu(1.2x + 1)$$
$$n_4(x) = Relu(1.2x - .2)$$
$$n_5(x) = Relu(2x - 1.1)$$
$$n_6(x) = Relu(5x - 5)$$

$$Z(x) = -n_1(x) - n_2(x) - n_3(x) + n_4(x) + n_5(x) + n_6(x)$$

Of course, in practice it is very difficult to optimize a specific network to find such an approximation

source

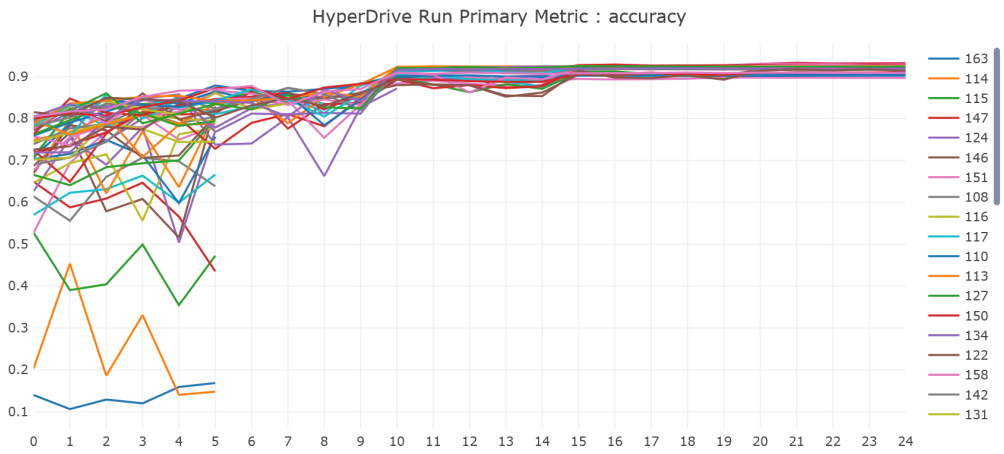
NN Training Considerations

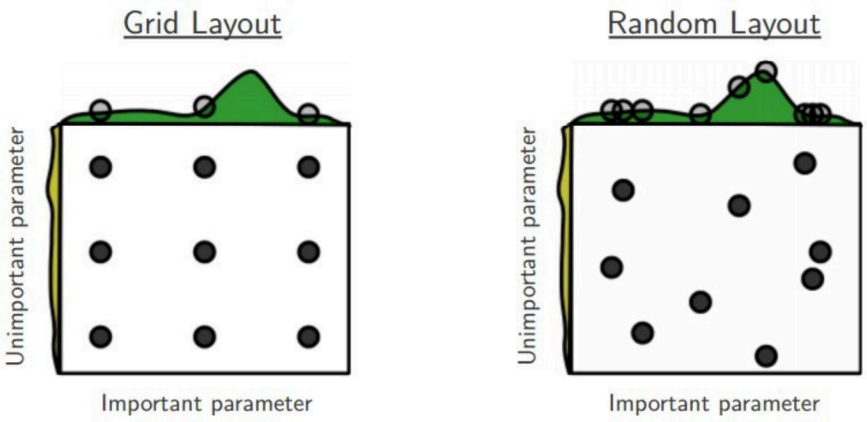


Hyperparameter Optimization

Hyperparameters are model settings that are not learned

- Neural networks have several hyper parameters that substantially affect performance
  - Learning rate, training epochs, network architecture, momentun batch size…
- Multiple methods to select hyperparameters
  - Manual: set by hand based on prior experience
  - Searches: prepare a grid of allowed parameter space and test combinations
  - Surrogate models: optimize a model of the validation loss as a function of hyperparameters (Bayesian optimization, genetic algorithms)



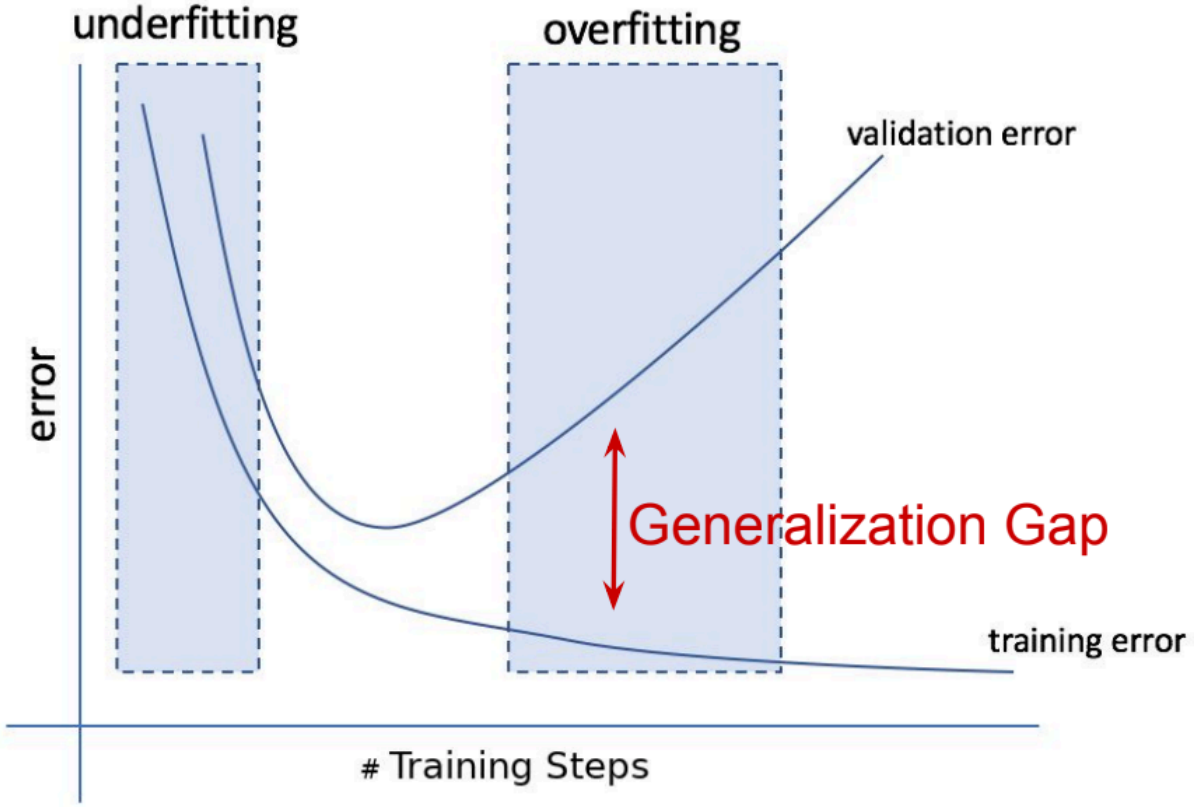


further reading

Over and Under Fitting in NNs

Underfitting

- Check model architecture
- Adjust learning rate
- Train longer
- Tune other hyperparameters



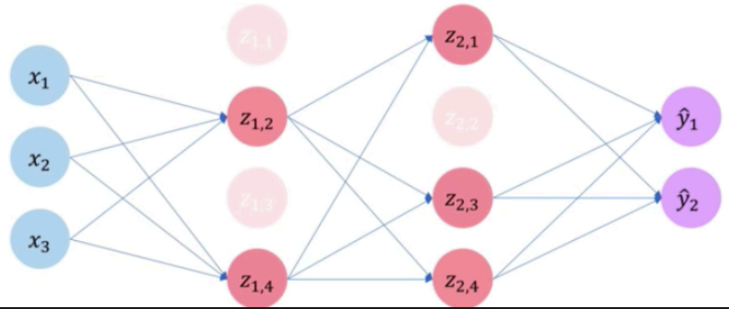
Overfitting

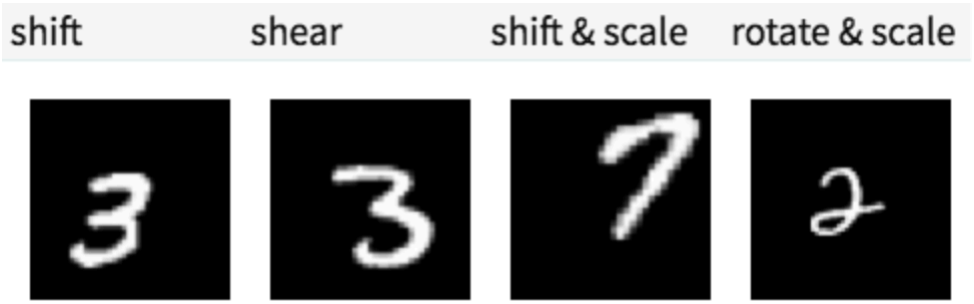
- Can you provide more training data?
- Tune hyperparameters or employ regularization
- Reduce model complexity

Regularization

A group of techniques to penalize a network for overfitting

- Mathematical regularization: add a term to loss function
  - L1: penalize absolute value of weights (can go to 0) $Cost\,function = Loss + \frac{\lambda}{2m} * \sum \|w\|$
  - L2: penalize square of weights (weight decay) $Cost\,function = Loss + \frac{\lambda}{2m} * \sum \|w\|^2$
- Drop out: set some neurons to 0
- Data augmentation: apply a transformation to increase training data
- Early stopping: stop training when validation loss diverges

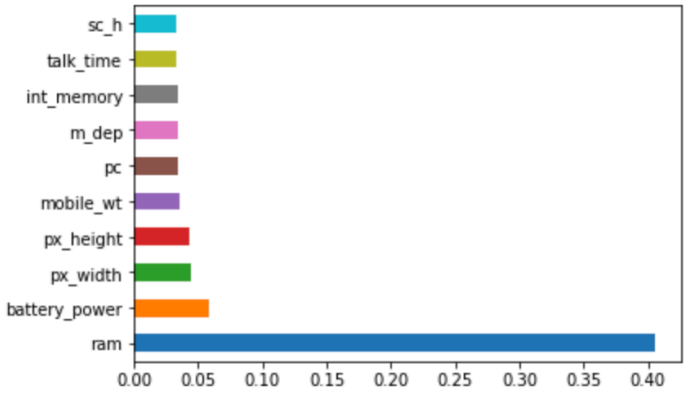


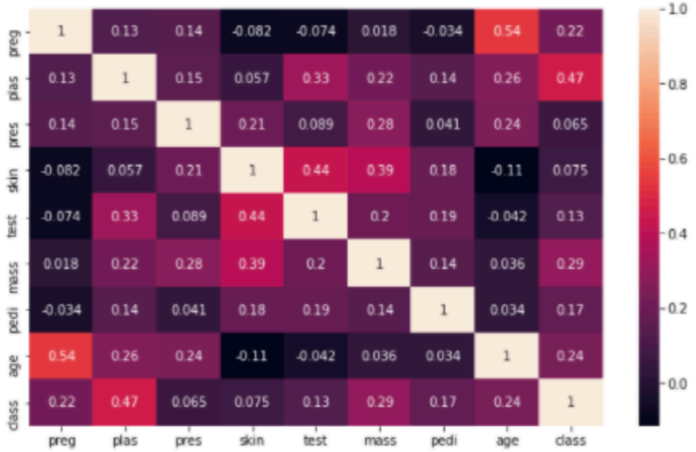


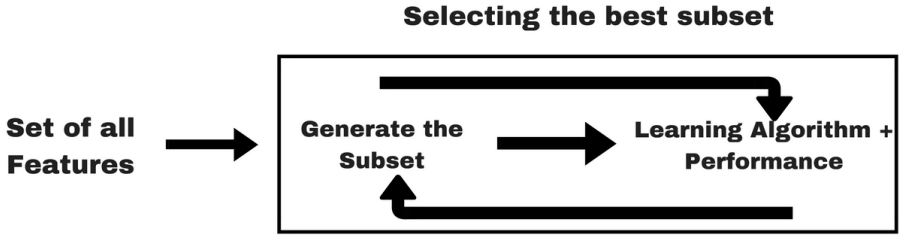
Feature Selection

Careful selection of the features used to train can improve model performance and avoid overtraining

- Filter methods: quantify intrinsic properties of features using univariate statistics
  - Feature importance: predictive power gained from each feature
  - Chi-square test: rank features by chi-square with output
  - Correlation matrices: eliminate highly correlated features
  - Variance threshold: drop features with low variance
- Wrapper Methods: search space of possible feature subsets
  - Forward/backward: iteratively add or remove features
  - Exhaustive: test every possible subset of features



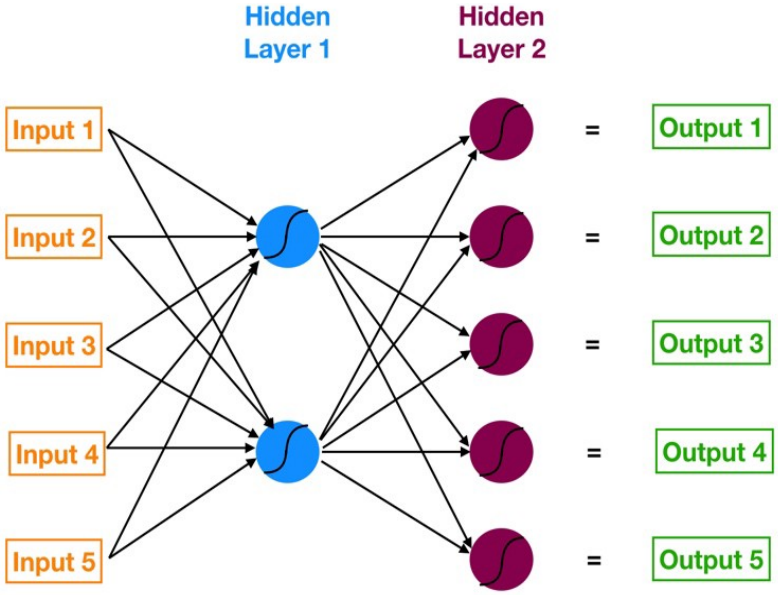


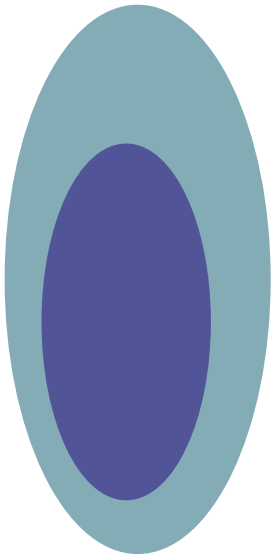


Neural Networks Model Card

Data is transformed into a space with easier decision boundaries

- Transformed linearly through matrix multiplications (learned weights)
- Activation functions applied to each neuron





- Training: GD
- Classification: select decision threshold
- Loss Function: problem dependent (MSE is basic)
- Hyperparameters:
  - Network architecture
  - Learning rate and regularization
  - Activation + loss functions

further reading

Pros:

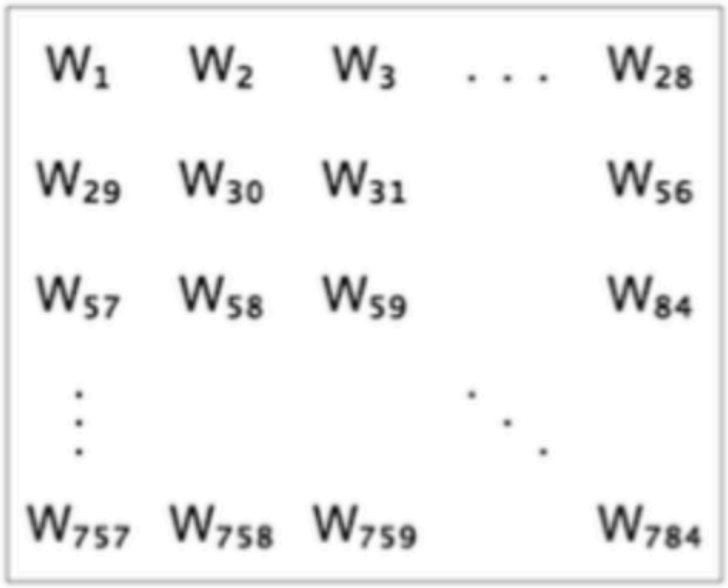
- Non-linear
- Handles high-dimensional input

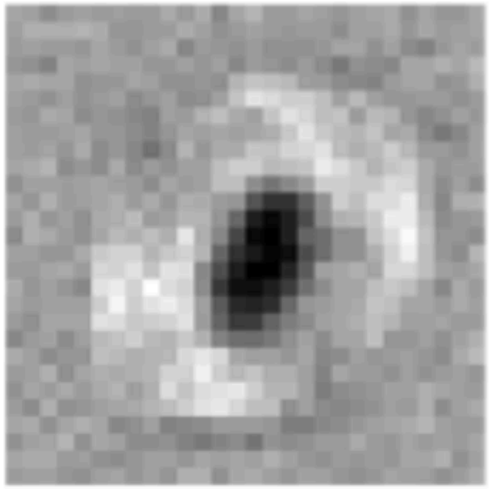
Cons:

- Black box
- Requires extensive training data + time
- Sensitive to hyperparameters

A Real Example of Learning

If time allows, walk through a tutorial to example what NNs learn about the MNIST dataset…



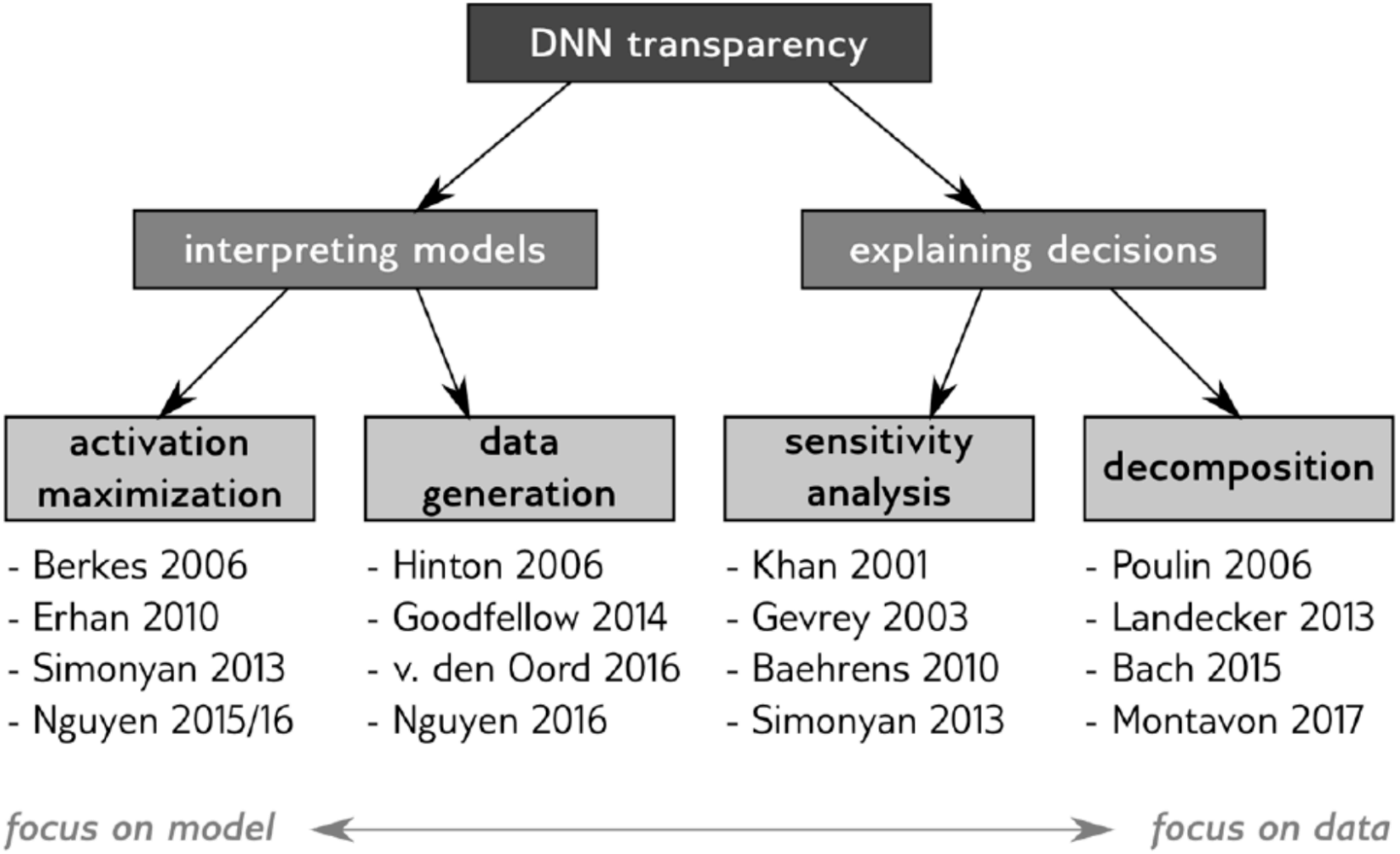


Visualizing the weights for the 0-neuron of an MNIST classifier

Question Time!

- How could you best evaluate a NN performance?
- How would you decide if a NN is right for your problem?
- What are signs during training that your NN is working well (or not)?
- How could you best evaluate a NN performance?
- How might you understand what the network is learning?

A Note on Interpretability



source

Coding Time!

Happy to answer any questions!

link to Day 2 notebook

✉ sthais@princeton.edu @basicsciencesav