Day 6: Neural Networks

Summer STEM: Machine Learning

Department of Electrical and Computer Engineering
NYU Tandon School of Engineering
Brooklyn, New York

Outline

Machine Learning Problem Pipeline

1. Formulate the problem: regression, classification, or others?
2. Gather and visualize the data
3. Design the model and the loss function
4. Train your model
   (a) Perform feature engineering
   (b) Construct the design matrix
   (c) Choose regularization techniques
   (d) Tune hyper-parameters using a validation set
   (e) If the performance is not satisfactory, go back to step (a).
5. Evaluate the model on a test set

Outline

Motivation

- Cannot rely on closed form solutions
  - Computation efficiency: operations like inverting a matrix is not efficient
  - For more complex problems such as neural networks, a closed-form solution is not always available
- Need an optimization technique to find an optimal solution
  - Machine learning practitioners use **gradient**-based methods

Gradient Descent Algorithm

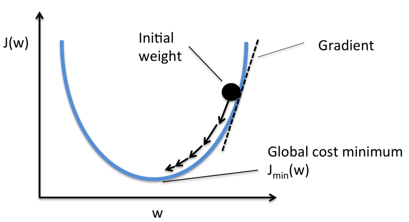
- Update Rule

  Repeat{

  $$\mathbf{w}_{new} = \mathbf{w} - \alpha \nabla J(\mathbf{w})$$

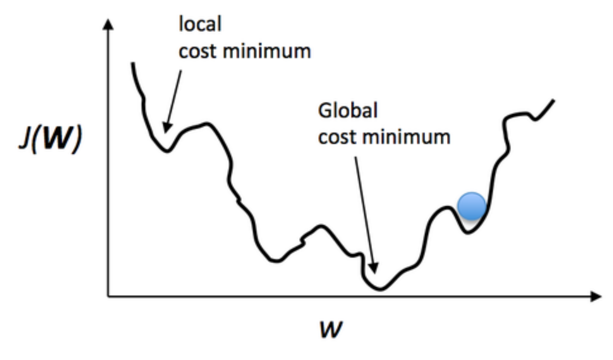
  }

  α is the learning rate

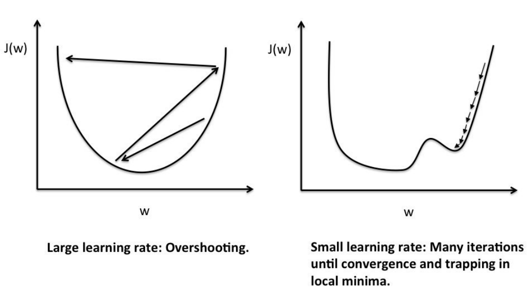


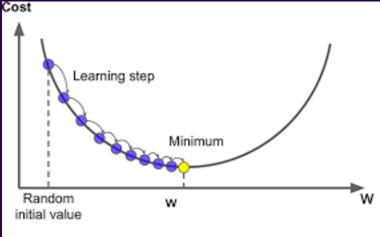
General Loss Function Contours

- Most loss function contours are not perfectly parabolic
- Our goal is to find a solution that is very close to global minimum by the right choice of hyper-parameters



Understanding Learning Rate

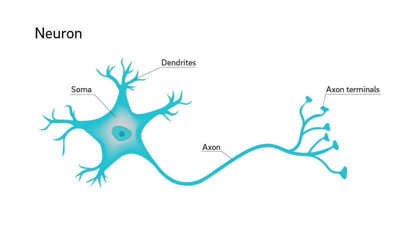




Correct learning rate

Outline

Biological Neuron



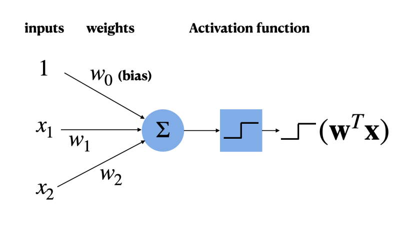
Source: David Baillot/UC San Diego

Mathematical Neuron: Perceptron

Biological Neuron:

- A neuron can receive electrochemical signals from other neurons;
- A neuron fires once its accumulated electric charge passes a certain threshold.
- Neurons that fire together wire together.

Mathematical Neuron (Perceptron)



Relation to Logistic Regression

What if we use the sigmoid function as the activation?

$$f(\mathbf{x}) = \sigma(\mathbf{w}^T \mathbf{x})$$

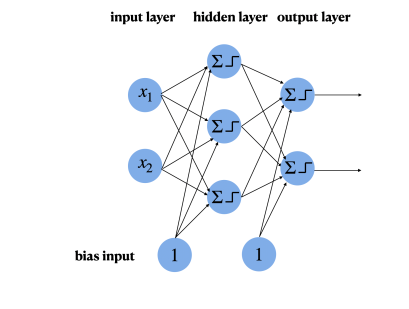
Decision boundary is a line: how is this supposed to revolutionize machine learning?

Multi-layer Perceptron (MLP)

We need more neurons and we need to connect them together!

- Many ways to do that...
- Today: multi-layer perceptron/fully connected feed-forward network.

MLP Example



- What is the shape of the input and output?
- How many parameters does this model have?
- What activation function would you use for the output layer? Why?

More about MLPs

- Many choices for the activation function: Sigmoid, Tanh, ReLU, Swish, etc.
- Many choices for the number of hidden layers and the number of neurons per layer.
- MLPs can approximate any continuous function given enough data.
- MLPs can overfit, but we know many effective ways of regularization.

Exercise: TensorFlow Keras Basics

Open `demo_tf_keras_basics.ipynb`

Outline

Deep Learning

What does deep learning stand for?

- Deep: Neural network architectures with many hidden layers.
- Learning: Optimizing model parameters given a dataset.

In general, the deeper the model is, the more parameters we need to learn and the more data is needed.

Large-Scale Machine Learning

For deep learning systems to perform well, large datasets are required

- COCO 330K images
- ImageNet 14 million images

Challenges:

- Memory limitation: GeForce RTX 2080 Ti has 11 GB memory, while ImageNet is about 300 GB.
- Computation: Calculating gradients for the whole dataset is computationally expensive (slow), and we need to do this many times.

Stochastic Gradient Descent

Idea: Instead of calculating the gradients from the whole dataset, do it only on a subset.

- Randomly select B samples from the dataset
- The loss for this subset

$$\tilde{J}(\mathbf{w}) = \frac{1}{B}\sum_{i=1}^{B} \|y - \hat{y}_i\|^2$$

- Update Rule

  $Repeat\{$

  $$\mathbf{w}_{new} = \mathbf{w} - \alpha \nabla \tilde{J}(\mathbf{w})$$

  $\}$

Yet Another Hyper-Parameter

This gives a noisy gradient

$$\nabla \tilde{J}(\mathbf{w}) = \nabla J(\mathbf{w}) + \epsilon$$

- **SGD**: $B = 1$, gives very noisy gradients
- **(batch) GD**: $B = N$, $\epsilon = 0$, expensive to compute
- **Mini-batch GD**: Pick a small B, typical values are $32, 64$, rarely more than 128 for image inputs

Some Noise Helps

Even if we can, we rarely set $B = N$. In fact, some noise in the gradients might help to

- escape from local minima,
- escape from saddle points, and
- improve generalization.

Outline

Overparameterized Models

- Modern deep learning models are heavily overparameterized, i.e. the number of learnable parameters is much larger than the number of the training samples.

Overparameterized Models

- Modern deep learning models are heavily overparameterized, i.e. the number of learnable parameters is much larger than the number of the training samples.
  - ResNet: State-of-the-art vision model, 10-60 million parameters

Overparameterized Models

- Modern deep learning models are heavily overparameterized, i.e. the number of learnable parameters is much larger than the number of the training samples.
  - ResNet: State-of-the-art vision model, 10-60 million parameters
  - GPT-3: State-of-the-art language model, 175 billion parameters

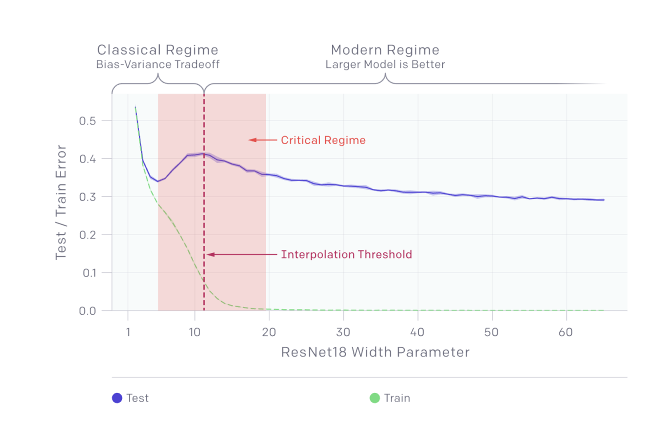
Overparameterized Models

- Modern deep learning models are heavily overparameterized, i.e. the number of learnable parameters is much larger than the number of the training samples.
  - ResNet: State-of-the-art vision model, 10-60 million parameters
  - GPT-3: State-of-the-art language model, 175 billion parameters
- Conventional wisdom: Such models overfit.

Overparameterized Models

- Modern deep learning models are heavily overparameterized, i.e. the number of learnable parameters is much larger than the number of the training samples.
  - ResNet: State-of-the-art vision model, 10-60 million parameters
  - GPT-3: State-of-the-art language model, 175 billion parameters
- Conventional wisdom: Such models overfit.
- It is not the case in practice!

Double Descent Curve



Source: OpenAI

Outline

Let's solve the mini-project with MLPs!

Open `lab_mlp_fish_market_keras.ipynb`