Data 515A, Spring 2018: Components of a learning instance

Dimitrios C. Gklezakos

University of Washington

May 1, 2018

Introduction

Our task: Classification of hand-written digits (a.k.a. the "hello world" of ML)

Introduction

Data: 28×28 grayscale images of hand-written digits

Introduction

Goal: Figure out which digit is represented from raw pixel values.

Introduction

Goal: Figure out which digit is represented from raw pixel values.
Idea: Express the class as a function of the image:

Introduction

Goal: Figure out which digit is represented from raw pixel values.
Idea: Express the class as a function of the image:

$$\text{class} = f(\text{image})$$

Introduction

Goal: Figure out which digit is represented from raw pixel values.
Idea: Express the class as a function of the image:

$$\text{class} = \text{model}(\text{image})$$

Logistic Regression

Intuition: The probability that the image belongs to a class is a function of the (transformed) input

Logistic Regression

Intuition: The probability that the image belongs to a class is a function of the (transformed) input

$$P(y = i|x) = \frac{e^{h_i(x)}}{\sum_{j=1}^{10} e^{h_j(x)}}$$

Logistic Regression

Intuition: The probability that the image belongs to a class is a function of the (transformed) input

$$P(y = i|x) = \frac{e^{h_i(x)}}{\sum_{j=1}^{10} e^{h_j(x)}}$$

where:

- x is the input image (vectorized)

Logistic Regression

Intuition: The probability that the image belongs to a class is a function of the (transformed) input

$$P(y = i|x) = \frac{e^{h_i(x)}}{\sum_{j=1}^{10} e^{h_j(x)}}$$

where:

- x is the input image (vectorized)
- y is the image label

Logistic Regression

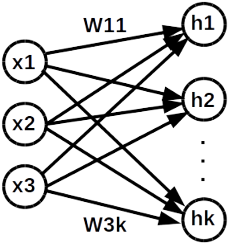
Intuition: The probability that the image belongs to a class is a function of the (transformed) input

$$P(y = i|x) = \frac{e^{h_i(x)}}{\sum_{j=1}^{10} e^{h_j(x)}}$$

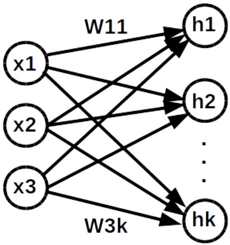
where:

- x is the input image (vectorized)
- y is the image label
- $h(.)$ is a linear transformation of the data (dense layer)

Dense Layer



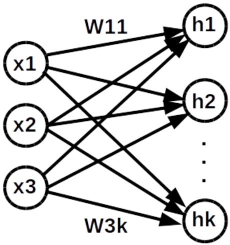
Dense Layer



Linear transformation:

$$h = Wx + b$$

Dense Layer



Linear transformation:

$$h = Wx + b$$

W is the weight matrix and b a bias term.

Softmax Activation

Problem: Convert activations h to probabilities.

Softmax Activation

Problem: Convert activations h to probabilities.
Solution:

$$\text{softmax}(h_i) = \frac{e^{h_i}}{\sum_j e^{h_j}}$$

Softmax Activation

Problem: Convert activations h to probabilities.
Solution:

$$\text{softmax}(h_i) = \frac{e^{h_i}}{\sum_j e^{h_j}}$$

Our model is essentialy a dense layer followed by a softmax activation!

Learning

- Our model has learnable parameters (o.w. no learning): W, b

Learning

- Our model has learnable parameters (o.w. no learning): W, b
- How can we tell that the parameters are good?

Learning

- Our model has learnable parameters (o.w. no learning): W, b
- How can we tell that the parameters are good?
- We need a loss function!

Loss

Cross-entropy loss:

$$L(y, \hat{y}) = -\sum_{j} y_j \log(\hat{y}_j)$$

Loss

Cross-entropy loss:

$$L(y, \hat{y}) = -\sum_j y_j \log(\hat{y}_j)$$

It is a similarity of distributions.

Optimization

Given a set of parameters W, b, how do we improve?

Optimization

Given a set of parameters W, b, how do we improve?

$$W \leftarrow W - \eta \frac{\partial L}{\partial W}$$

$$b \leftarrow b - \eta \frac{\partial L}{\partial b}$$

Gradient descent!

Components of an instance

Components of an instance

- Data

Components of an instance

- Data
- Model

Components of an instance

- Data
- Model
- Loss function

Components of an instance

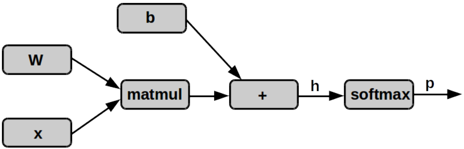
- Data
- Model
- Loss function
- Optimization algorithm

Computational Graph

Idea: Express computation as a graph

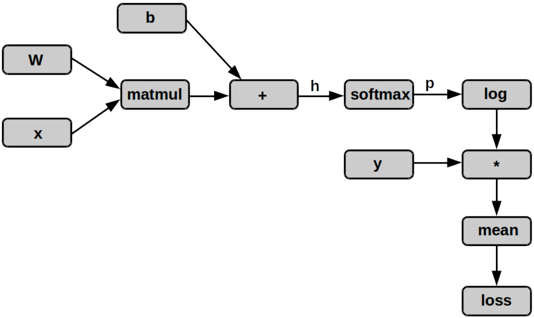
Computational Graph

Idea: Express computation as a graph



Computational Graph

Idea: Express computation as a graph



Computational Graph

- Nodes are no longer values but symbolic tensors

Computational Graph

- Nodes are no longer values but symbolic tensors
- A session runs the requested computation

Computational Graph

- Nodes are no longer values but symbolic tensors
- A session runs the requested computation

Advantages?

Computational Graph

Optimizations:

Computational Graph

Optimizations:

- Dead code elimination

Computational Graph

Optimizations:

- Dead code elimination
- Memory planning

Computational Graph

Optimizations:

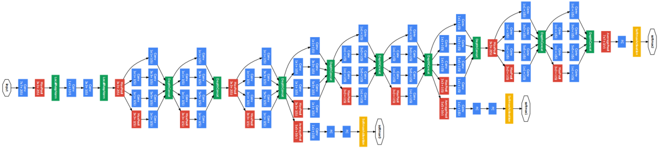
- Dead code elimination
- Memory planning
- Parallelization

Computational Graph

What happens to differentiation if I stack many layers together?

Computational Graph

What happens to differentiation if I stack many layers together?



Computational Graph

Chain rule:

$$\frac{\partial f\left(g(x)\right)}{\partial x} = \frac{\partial f}{\partial g}\frac{\partial g}{\partial x}$$

Computational Graph

Chain rule:

$$\frac{\partial f\left(g(x)\right)}{\partial x} = \frac{\partial f}{\partial g}\frac{\partial g}{\partial x}$$

Nightmare!

Computational Graph

Chain rule:

$$\frac{\partial f\left(g(x)\right)}{\partial x} = \frac{\partial f}{\partial g}\frac{\partial g}{\partial x}$$

Nightmare!

Idea: Build a computational graph for the gradients based on the graph for the execution (Automatic Differentiation).

Deep learning frameworks

Widely used frameworks: Tensorflow, PyTorch, Theano, chainer, MXNet