EE 628
Deep Learning
Fall 2019

Lecture 3

09/12/2019

Assistant Prof. Sergul Aydore

Department of Electrical and Computer Engineering

STEVENS INSTITUTE of TECHNOLOGY

THE INNOVATION UNIVERSITY®

Overview

- Last lecture we covered
  - Linear neural networks
  - Implementation of linear regression from scratch

- Today, we will cover
  - Softmax regression
  - Multilayer perceptron

Softmax Regression

- In practice, sometimes we are interested in classification: asking not *how much* but *which one.*
  - Does this email belong in the spam folder or the inbox?

Softmax Regression

- In practice, sometimes we are interested in classification: asking not *how much* but *which one.*
  - Does this email belong in the spam folder or the inbox?
  - Is this customer more likely to sign up or not to sign up for a subscription service?

Softmax Regression

- In practice, sometimes we are interested in classification: asking not *how much* but *which one.*
  - Does this email belong in the spam folder or the inbox?
  - Is this customer more likely to sign up or not to sign up for a subscription service?
  - Does this image depict a donkey, a dog, a cat, or a rooster?

Softmax Regression

- In practice, sometimes we are interested in classification: asking not *how much* but *which one.*
  - Does this email belong in the spam folder or the inbox?
  - Is this customer more likely to sign up or not to sign up for a subscription service?
  - Does this image depict a donkey, a dog, a cat, or a rooster?
  - Which movie is user most likely to watch next?

Classification Problems

- Let’s start with a really simple image classification problem

Classification Problems

- Let’s start with a really simple image classification problem
- Each input will be a grayscale 2 x 2 image

Classification Problems

- Let’s start with a really simple image classification problem
- Each input will be a grayscale 2 x 2 image
- Each image can be represented with 4 features: x_1, x_2, x_3, x_4

Classification Problems

- Let's start with a really simple image classification problem
- Each input will be a grayscale 2 x 2 image
- Each image can be represented with 4 features: x_1, x_2, x_3, x_4
- Let's assume each image belongs to one among three categories: *cat*, *chicken* and *dog*

Classification Problems

- Let's start with a really simple image classification problem
- Each input will be a grayscale 2 x 2 image
- Each image can be represented with 4 features: x_1, x_2, x_3, x_4
- Let's assume each image belongs to one among three categories: *cat*, *chicken* and *dog*
- We need to represent the labels. Which one makes more sense?
  - $y \in \{1,\ 2,\ 3\}$
  - $y \in \{(1,\ 0,\ 0),\ (0, 1, 0),\ (0,\ 0,\ 1)\}$

Network Architecture

- In order to estimate multiple classes, we need a model with multiple outputs, one per category.

Network Architecture

- In order to estimate multiple classes, we need a model with multiple outputs, one per category.
- We compute 3 outputs for each input

$$
\begin{aligned}
o_1 &= x_1 w_{11} + x_2 w_{21} + x_3 w_{31} + x_4 w_{41} + b_1 \\
o_2 &= x_1 w_{12} + x_2 w_{22} + x_3 w_{32} + x_4 w_{42} + b_2 \\
o_3 &= x_1 w_{13} + x_2 w_{23} + x_3 w_{33} + x_4 w_{43} + b_3
\end{aligned}
$$

Network Architecture

- In order to estimate multiple classes, we need a model with multiple outputs, one per category.
- We compute 3 outputs for each input

$$o_1 = x_1 w_{11} + x_2 w_{21} + x_3 w_{31} + x_4 w_{41} + b_1$$
$$o_2 = x_1 w_{12} + x_2 w_{22} + x_3 w_{32} + x_4 w_{42} + b_2$$
$$o_3 = x_1 w_{13} + x_2 w_{23} + x_3 w_{33} + x_4 w_{43} + b_3$$

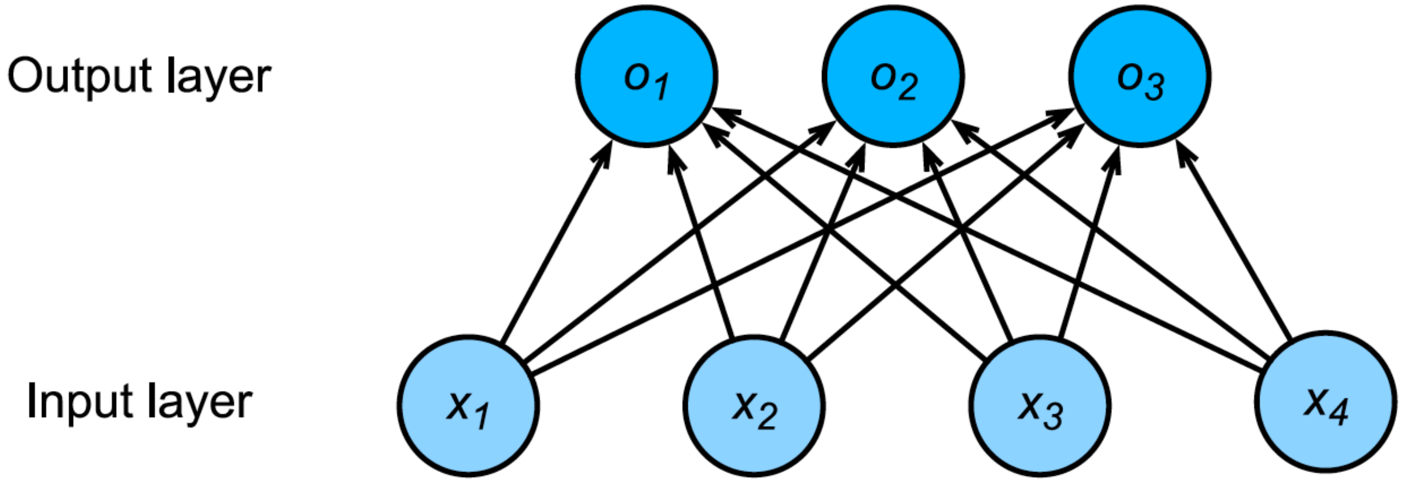

Fig. 5.4.1: Softmax regression is a single-layer neural network.

Softmax Operation

- Let’s use linear algebra notation for our model $\mathbf{o} = \mathbf{W}\mathbf{x} + \mathbf{b}$

Softmax Operation

- Let’s use linear algebra notation for our model $\mathbf{o} = \mathbf{W}\mathbf{x} + \mathbf{b}$
- We want our outputs to correspond to probabilities to indicate our confidence that the item belongs to each category

Softmax Operation

- Let's use linear algebra notation for our model $\mathbf{o} = \mathbf{W}\mathbf{x} + \mathbf{b}$
- We want our outputs to correspond to probabilities to indicate our confidence that the item belongs to each category
- Challenges:
  - We need to guarantee that each output is nonnegative
  - Probabilities sum up to 1

Softmax Operation

- Let's use linear algebra notation for our model $\mathbf{o} = \mathbf{W}\mathbf{x} + \mathbf{b}$
- We want our outputs to correspond to probabilities to indicate our confidence that the item belongs to each category
- Challenges:
  - We need to guarantee that each output is nonnegative
  - Probabilities sum up to 1
- Softmax function does it all!

$$\hat{\mathbf{y}} = \mathrm{softmax}(\mathbf{o}) \text{ where } \hat{y}_i = \frac{\exp(o_i)}{\sum_j \exp(o_j)}$$

Softmax Operation

- Let’s use linear algebra notation for our model $\mathbf{o} = \mathbf{W}\mathbf{x} + \mathbf{b}$
- We want our outputs to correspond to probabilities to indicate our confidence that the item belongs to each category
- Challenges:
  - We need to guarantee that each output is nonnegative
  - Probabilities sum up to 1
- Softmax function does it all!

$$\hat{\mathbf{y}} = \mathrm{softmax}(\mathbf{o}) \text{ where } \hat{y}_i = \frac{\exp(o_i)}{\sum_j \exp(o_j)}$$

- Vectorization for minibatches:

$$\mathbf{O} = \mathbf{X}\mathbf{W} + \mathbf{b}$$

$$\hat{\mathbf{Y}} = \mathrm{softmax}(\mathbf{O})$$

Loss Function

- Log likelihood: check how well we predicted what we observe

$$p(Y|X) = \prod_{i=1}^{n} p(y^{i}|x^{(i)}) \text{ and thus } -\log p(Y|X) = \sum_{i=1}^{n} \underbrace{-\log p(y^{(i)}|x^{(i)})}$$

$$-\log p(y^{(i)}|x^{(i)}) = -\sum y_j \log \hat{y}_j$$

Loss Function

- Log likelihood: check how well we predicted what we observe

$$p(Y|X) = \prod_{i=1}^{n} p(y^i|x^{(i)}) \text{ and thus } -\log p(Y|X) = \sum_{i=1}^{n} \underbrace{-\log p(y^{(i)}|x^{(i)})}$$

$$-\log p(y^{(i)}|x^{(i)}) = -\sum_j y_j \log \hat{y}_j$$

- Compute the derivative with respect to o_j.

Loss Function

- Log likelihood: check how well we predicted what we observe

$$p(Y|X) = \prod_{i=1}^{n} p(y^i|x^{(i)}) \text{ and thus } -\log p(Y|X) = \sum_{i=1}^{n} \underbrace{-\log p(y^{(i)}|x^{(i)})}$$

$$-\log p(y^{(i)}|x^{(i)}) = -\sum_j y_j \log \hat{y}_j$$

- Compute the derivative with respect to o_j.
- Cross entropy loss: one of the most commonly used losses for multiclass classification $l(\mathbf{y}, \hat{\mathbf{y}}) = -\sum y_j \log(\hat{y}_j)$

Loss Function

- Log likelihood: check how well we predicted what we observe

$$p(Y|X) = \prod_{i=1}^{n} p(y^i|x^{(i)}) \text{ and thus } -\log p(Y|X) = \sum_{i=1}^{n} \underbrace{-\log p(y^{(i)}|x^{(i)})}$$

$$-\log p(y^{(i)}|x^{(i)}) = -\sum_j y_j \log \hat{y}_j$$

- Compute the derivative with respect to o_j.
- Cross entropy loss: one of the most commonly used losses for multiclass classification

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

- Kullback Leibler Divergence: measures similarity between two distributions

$$D(p||q) = \sum_j p(j) \log \frac{p(j)}{q(j)}$$

Loss Function

- Log likelihood: check how well we predicted what we observe

$$p(Y|X) = \prod_{i=1}^{n} p(y^i|x^{(i)}) \text{ and thus } -\log p(Y|X) = \sum_{i=1}^{n} \underbrace{-\log p(y^{(i)}|x^{(i)})}$$

$$-\log p(y^{(i)}|x^{(i)}) = -\sum_j y_j \log \hat{y}_j$$

- Compute the derivative with respect to o_j.
- Cross entropy loss: one of the most commonly used losses for multiclass classification

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

- Kullback Leibler Divergence: measures similarity between two distributions

$$D(p||q) = \sum_j p(j) \log \frac{p(j)}{q(j)}$$

- Minimizing $D(p\ ||\ q\ )$ with respect to q is equivalent to minimizing the cross-entropy loss. (PROVE!)

Introduce Fashion-MNIST data

- notebook

Implementation of Softmax from scratch

- notebook

Concise Implementation of Softmax from scratch

- notebook

Multilayer Perceptrons

- The simplest deep networks

Multilayer Perceptrons

- The simplest deep networks
- They consist of many layers of neurons

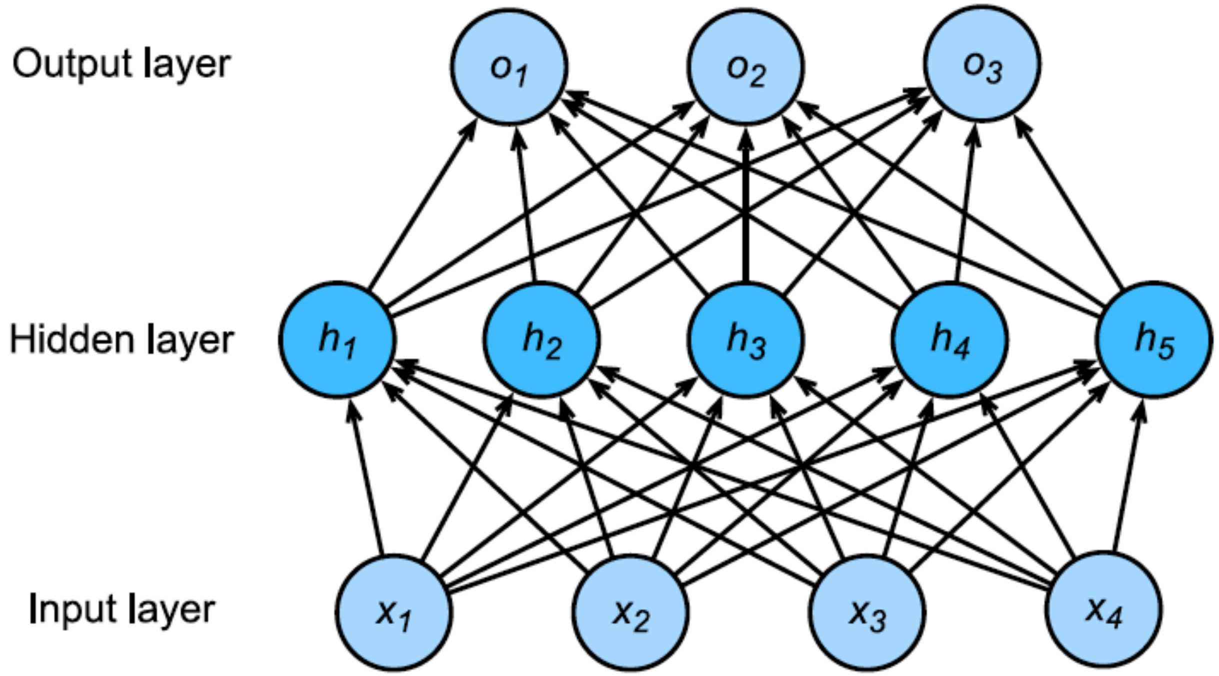

Fig. 6.1.2: Multilayer perceptron with hidden layers. This example contains a hidden layer with 5 hidden units in it.

From Linear to Nonlinear

- In order to get a benefit from multilayer architectures, we need another key ingredient– a nonlinearity σ to be applied to each of the hidden units after each layer's linear transformation

From Linear to Nonlinear

- In order to get a benefit from multilayer architectures, we need another key ingredient– a nonlinearity σ to be applied to each of the hidden units after each layer's linear transformation
- The most popular choice for the nonlinearity these days is the rectified linear unit (ReLU) $\max(x, 0)$.

From Linear to Nonlinear

- In order to get a benefit from multilayer architectures, we need another key ingredient– a nonlinearity σ to be applied to each of the hidden units after each layer's linear transformation
- The most popular choice for the nonlinearity these days is the rectified linear unit (ReLU) $\max(x, 0)$.
- After incorporating these no-linearities it becomes possible to merge layers

$$\mathbf{h} = \sigma(\mathbf{W}_1\mathbf{x} + \mathbf{b}_1)$$

$$\mathbf{o} = \mathbf{W}_2\mathbf{h} + \mathbf{b}_2$$

$$\hat{\mathbf{y}} = \mathrm{softmax}(\mathbf{o})$$

From Linear to Nonlinear

- In order to get a benefit from multilayer architectures, we need another key ingredient– a nonlinearity σ to be applied to each of the hidden units after each layer's linear transformation
- The most popular choice for the nonlinearity these days is the rectified linear unit (ReLU) $\max(x, 0)$.
- After incorporating these no-linearities it becomes possible to merge layers

$$\mathbf{h} = \sigma(\mathbf{W}_1\mathbf{x} + \mathbf{b}_1)$$
$$\mathbf{o} = \mathbf{W}_2\mathbf{h} + \mathbf{b}_2$$
$$\hat{\mathbf{y}} = \mathrm{softmax}(\mathbf{o})$$

- Clearly, we can continue stacking such hidden layers.

Vectorization and mini-batch

- As before, by the matrix $\mathbf{X}$, we denote a mini-batch of inputs.

Vectorization and mini-batch

- As before, by the matrix $\mathbf{X}$, we denote a mini-batch of inputs.
- The calculations to produce outputs from an MLP with two hidden layers can thus be expressed:

$$\mathbf{H}_1 = \sigma(\mathbf{X}\mathbf{W}_1 + \mathbf{b}_1)$$

$$\mathbf{H}_2 = \sigma(\mathbf{H}_1\mathbf{W}_2 + \mathbf{b}_2)$$

$$\hat{\mathbf{Y}} = \mathrm{softmax}(\mathbf{H}_2\mathbf{W}_3 + \mathbf{b}_3)$$

Activation Functions

- notebook

Implementing MLP from scratch

- notebook

Concise Implementation of MLP

- notebook

Next Week

- Overfitting and Underfitting