

COMPSCI 689

Lecture 10: Multilayer Perceptrons

Brendan O'Connor

College of Information and Computer Sciences
University of Massachusetts Amherst



Slides by Benjamin M. Marlin (marlin@cs.umass.edu).

Outline

1 Neural Networks

2 Learning for NNs

3 Deep Learning Tools

Learning for Sigmoid NN Regression

- Learning sigmoid neural networks for regression follows the same basic pattern.
- All we need to do is switch to a regression loss. The model defined so far already has a linear output layer.
- The derivation of the gradient is nearly identical regardless of the loss used.

Regularization

- Like with linear regression, logistic regression, and SVMs, regularizing neural networks typically improves generalization performance.

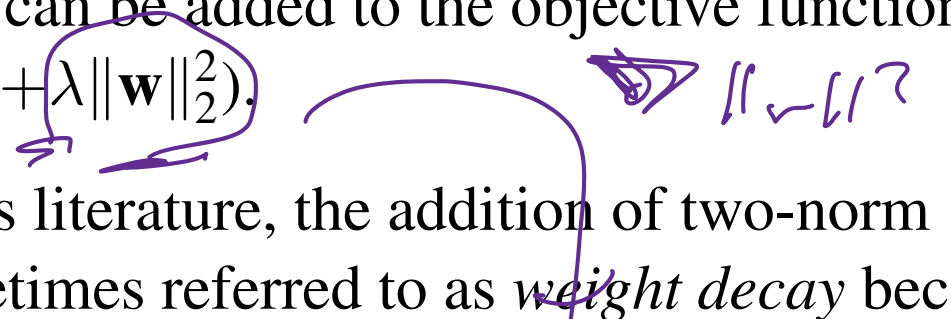
Regularization

- Like with linear regression, logistic regression, and SVMs, regularizing neural networks typically improves generalization performance.
- Standard regularizers can be added to the objective function to accomplish this. (i.e. $+\lambda\|\mathbf{w}\|_2^2$).

Regularization

- Like with linear regression, logistic regression, and SVMs, regularizing neural networks typically improves generalization performance.
- Standard regularizers can be added to the objective function to accomplish this. (i.e. $+\lambda\|\mathbf{w}\|_2^2$).
- In the neural networks literature, the addition of two-norm regularization is sometimes referred to as *weight decay* because of the contribution to the learning step (i.e., $-\lambda\mathbf{w}$).

Regularization

- Like with linear regression, logistic regression, and SVMs, regularizing neural networks typically improves generalization performance.
- Standard regularizers can be added to the objective function to accomplish this. (i.e. $+\lambda\|\mathbf{w}\|_2^2$).
- In the neural networks literature, the addition of two-norm regularization is sometimes referred to as *weight decay* because of the contribution to the learning step (i.e., $-\lambda\mathbf{w}$).
- The number of layers and the width of the layers are also both important architectural hyper-parameters that impact model capacity.

Optimization Details

- **Optimizers:** Second-order optimizers are rarely used to learn neural network models. Instead, first-order methods are typically used (i.e., gradient descent).

Optimization Details

- **Optimizers:** Second-order optimizers are rarely used to learn neural network models. Instead, first-order methods are typically used (i.e., gradient descent).
- **Stochastic Approximation:** It is also typical to compute gradients using sub-sets of the data. This typically speeds convergence. In this literature, this is often referred to as *mini-batch* learning or *stochastic gradient descent*.

Optimization Details

- **Optimizers:** Second-order optimizers are rarely used to learn neural network models. Instead, first-order methods are typically used (i.e., gradient descent).
- **Stochastic Approximation:** It is also typical to compute gradients using sub-sets of the data. This typically speeds convergence. In this literature, this is often referred to as *mini-batch* learning or *stochastic gradient descent*.
- **Step Sizes:** Fixed step sizes or fixed step size decay schedules were often used in early work. In this literature, the step size is often referred to as the *learning rate*.

Optimization Details

- **Optimizers:** Second-order optimizers are rarely used to learn neural network models. Instead, first-order methods are typically used (i.e., gradient descent).
- **Stochastic Approximation:** It is also typical to compute gradients using sub-sets of the data. This typically speeds convergence. In this literature, this is often referred to as *mini-batch* learning or *stochastic gradient descent*.
- **Step Sizes:** Fixed step sizes or fixed step size decay schedules were often used in early work. In this literature, the step size is often referred to as the *learning rate*.
- **Acceleration:** There are a wide array of approaches that either modify the gradient direction or the step sizes to attempt to achieve faster convergence on certain classes of problems (i.e., use of momentum, Adadelata, Adagrad, Adam, RMSprop, etc.).

Sigmoid Neural Network Limitations

- Unlike SVMs, logistic regression and OLS linear regression, multi-layer sigmoid neural network models have many local optima.

Sigmoid Neural Network Limitations

- Unlike SVMs, logistic regression and OLS linear regression, multi-layer sigmoid neural network models have many local optima.
- In practice, it can be hard to find a combination of layer sizes and regularization that yields good generalization performance when learning from small or moderate amounts of data.

Sigmoid Neural Network Limitations

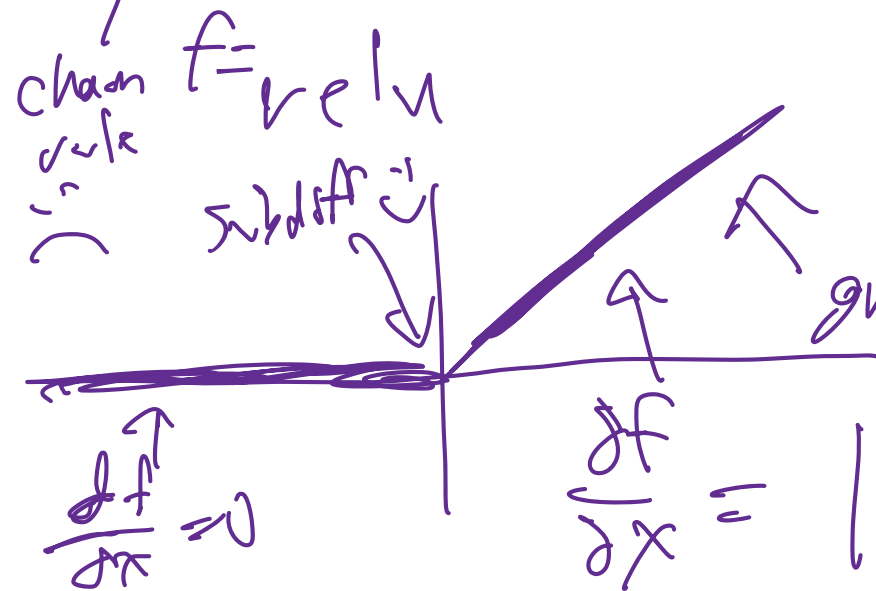
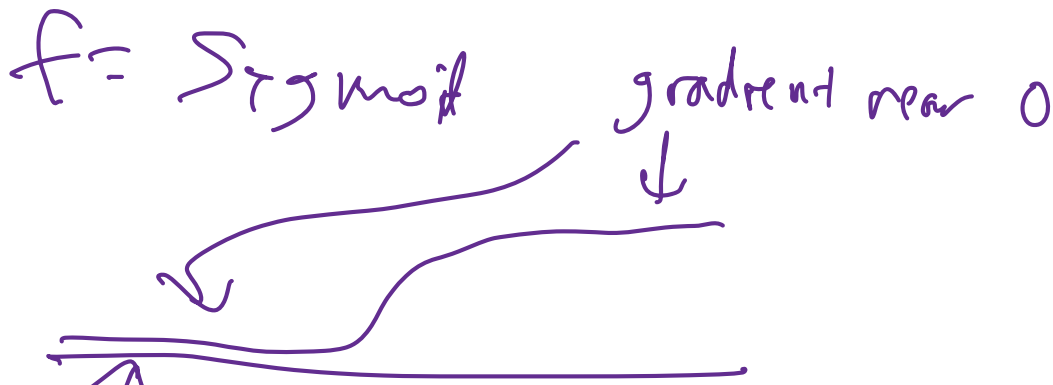
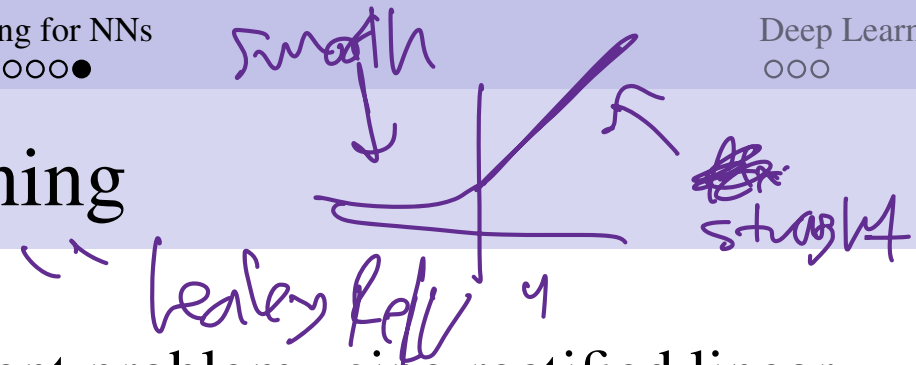
- Unlike SVMs, logistic regression and OLS linear regression, multi-layer sigmoid neural network models have many local optima.
- In practice, it can be hard to find a combination of layer sizes and regularization that yields good generalization performance when learning from small or moderate amounts of data.
- In addition, the use of sigmoid functions for the hidden units results in a *vanishing gradient* problem that can further slow down learning for deep networks.

Sigmoid Neural Network Limitations

- Unlike SVMs, logistic regression and OLS linear regression, multi-layer sigmoid neural network models have many local optima.
- In practice, it can be hard to find a combination of layer sizes and regularization that yields good generalization performance when learning from small or moderate amounts of data.
- In addition, the use of sigmoid functions for the hidden units results in a *vanishing gradient* problem that can further slow down learning for deep networks.
- As a result, there were few examples of deep neural network models that resulted in performance that exceeded hand-crafted features, basis expansions and kernels until the last decade.

Towards Modern Deep Learning

- 1 Address the vanishing gradient problem using rectified linear units: $relu(x) = \max(0, x)$.



$$h = f(wx)$$

$$\frac{\partial R}{\partial w} = \dots = \frac{\partial f}{\partial w} \dots$$

$$\text{Sigmoid}(wx = \text{neg}) \Rightarrow \text{deriv} \approx 0$$

$\max(x, 0)$
gradient passes thru in chain rule

Towards Modern Deep Learning

- 1 Address the vanishing gradient problem using rectified linear units: $relu(x) = \max(0, x)$.
- 2 Have access to lots of labeled data (ie: millions of examples).

Towards Modern Deep Learning

- 1 Address the vanishing gradient problem using rectified linear units: $relu(x) = \max(0, x)$.
- 2 Have access to lots of labeled data (ie: millions of examples).
- 3 Do the computing on GPUs to achieve significant speedups (i.e.: model training takes days instead of weeks or months).

Towards Modern Deep Learning

- 1 Address the vanishing gradient problem using rectified linear units: $relu(x) = \max(0, x)$.
- 2 Have access to lots of labeled data (ie: millions of examples).
- 3 Do the computing on GPUs to achieve significant speedups (i.e.: model training takes days instead of weeks or months).
- 4 The breakthrough paper in this area is Krizhevsky, Sutskever, and Hinton. *Imagenet classification with deep convolutional neural networks*. Advances in neural information processing systems 25, 2012.

Outline

- 1 Neural Networks
- 2 Learning for NNs
- 3 Deep Learning Tools**

Deep Learning Modeling Tools

- As we have seen, deep learning models just require (sub-) gradients of the loss with respect to the parameters to enable learning.

Deep Learning Modeling Tools

- As we have seen, deep learning models just require (sub-) gradients of the loss with respect to the parameters to enable learning.
- However, correctly deriving gradients from complex model architectures by hand can be tedious and is error prone.

Deep Learning Modeling Tools

- As we have seen, deep learning models just require (sub-) gradients of the loss with respect to the parameters to enable learning.
- However, correctly deriving gradients from complex model architectures by hand can be tedious and is error prone.
- As a result, many deep learning tools exist (i.e., PyTorch, TensorFlow, Caffe, Keras) that allow models to be specified only in terms of the *forward pass* computation (from inputs to loss).

Deep Learning Modeling Tools

- As we have seen, deep learning models just require (sub-) gradients of the loss with respect to the parameters to enable learning.
- However, correctly deriving gradients from complex model architectures by hand can be tedious and is error prone.
- As a result, many deep learning tools exist (i.e., PyTorch, TensorFlow, Caffe, Keras) that allow models to be specified only in terms of the *forward pass* computation (from inputs to loss).
- This makes it much easier to quickly change the model architecture because only the specification of the forward pass needs to be updated.

Autodiff

- An idea called *automatic differentiation* is used to convert the forward pass specification into the gradient computations needed for learning.

Autodiff

- An idea called *automatic differentiation* is used to convert the forward pass specification into the gradient computations needed for learning.
- A computation graph is extracted from the specification of the forward pass.

Autodiff

- An idea called *automatic differentiation* is used to convert the forward pass specification into the gradient computations needed for learning.
- A computation graph is extracted from the specification of the forward pass.
- The chain rule is then applied to each node in the computation graph to transform it into a computation graph for computing the gradient of the objective.

Autodiff

- An idea called *automatic differentiation* is used to convert the forward pass specification into the gradient computations needed for learning.
- A computation graph is extracted from the specification of the forward pass.
- The chain rule is then applied to each node in the computation graph to transform it into a computation graph for computing the gradient of the objective.
- Importantly, this procedure yields analytic gradients and learning is identical to classical backprop.

Autodiff

- An idea called *automatic differentiation* is used to convert the forward pass specification into the gradient computations needed for learning.
- A computation graph is extracted from the specification of the forward pass.
- The chain rule is then applied to each node in the computation graph to transform it into a computation graph for computing the gradient of the objective.
- Importantly, this procedure yields analytic gradients and learning is identical to classical backprop.
- As a byproduct of the representations used, it's possible to compile both computation graphs against arbitrary numerical libraries for either CPUs or GPUs.