EE 628
Deep Learning
Fall 2019

Lecture 4

02/13/2019

Assistant Prof. Sergul Aydore

Department of Electrical and Computer Engineering

STEVENS INSTITUTE of TECHNOLOGY

THE INNOVATION UNIVERSITY®

Overview

- Last lecture we covered
  - Softmax Regression
- Today, we will cover
  - Multilayer perceptron
  - Overfitting/underfitting

Multilayer Perceptrons

- The simplest deep networks

Multilayer Perceptrons

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

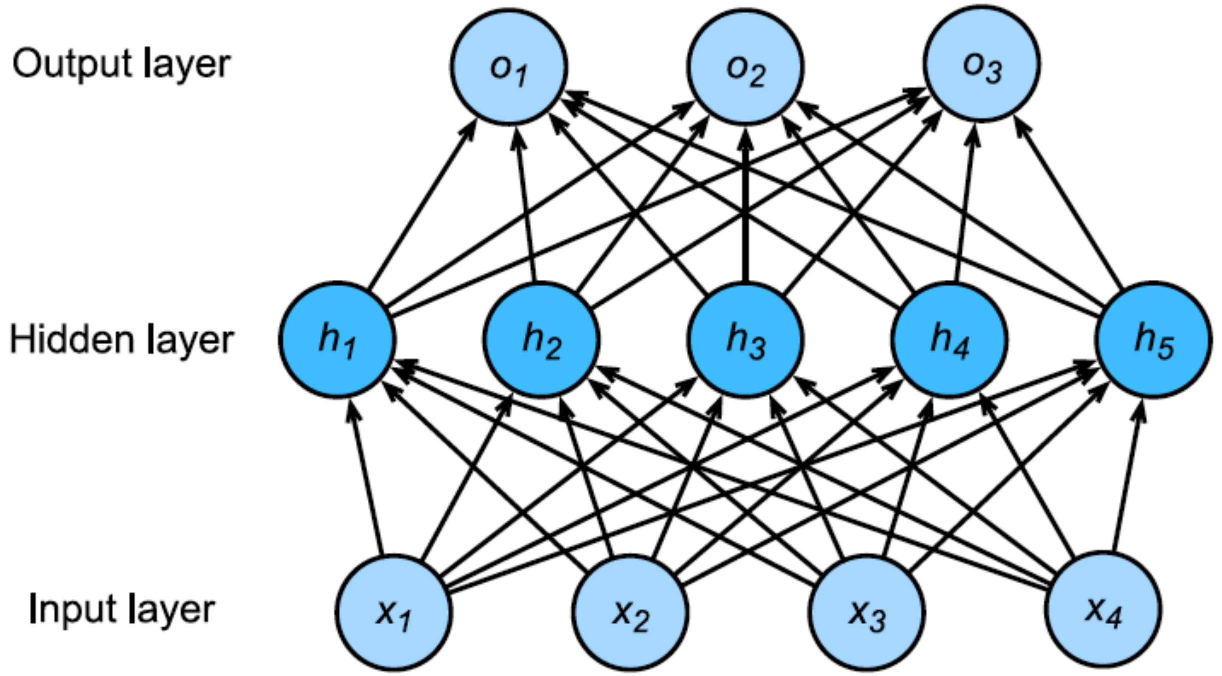

Fig. 6.1.2: Multilayer perceptron with hidden layers. This example contains a hidden layer with 5 hidden units in it.

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

Concise Implementation of MLP

- notebook

Model Selection, Underfitting, Overfitting

- As machine learning scientists, our goal is to discover general patterns.

Model Selection, Underfitting, Overfitting

- As machine learning scientists, our goal is to discover general patterns.
- Not to memorize our training set!

Model Selection, Underfitting, Overfitting

- As machine learning scientists, our goal is to discover general patterns.
- Not to memorize our training set!
- How to discover patters that **generalize** is the fundamental problem of machine learning.

Model Selection, Underfitting, Overfitting

- As machine learning scientists, our goal is to discover general patterns.
- Not to memorize our training set!
- How to discover patters that **generalize** is the fundamental problem of machine learning.
- The danger is that when we train models, we access just a small sample of data.

Model Selection, Underfitting, Overfitting

- As machine learning scientists, our goal is to discover general patterns.
- Not to memorize our training set!
- How to discover patters that **generalize** is the fundamental problem of machine learning.
- The danger is that when we train models, we access just a small sample of data.
- The phenomena of fitting our training data more closely than we fit the underlying distribution is called **overfitting**.

Model Selection, Underfitting, Overfitting

- As machine learning scientists, our goal is to discover general patterns.
- Not to memorize our training set!
- How to discover patters that **generalize** is the fundamental problem of machine learning.
- The danger is that when we train models, we access just a small sample of data.
- The phenomena of fitting our training data more closely than we fit the underlying distribution is called **overfitting**.
- The techniques used to combat overfitting are called **generalization**.

Training Error and Generalization Error

- The training error is the error of our model as calculated on the training data set.

Training Error and Generalization Error

- The training error is the error of our model as calculated on the training data set.
- The generalization error is the expectation of our model's error were we to apply it to an infinite data points from the same underlying distribution as our original sample.

Training Error and Generalization Error

- The training error is the error of our model as calculated on the training data set.
- The generalization error is the expectation of our model's error were we to apply it to an infinite data points from the same underlying distribution as our original sample.
- In practice, we must estimate the generalization error by applying our model to an independent test set that were withheld from our training set.

Training Error and Generalization Error

- The training error is the error of our model as calculated on the training data set.
- The generalization error is the expectation of our model's error were we to apply it to an infinite data points from the same underlying distribution as our original sample.
- In practice, we must estimate the generalization error by applying our model to an independent test set that were withheld from our training set.
- Example: a student memorizing all exam questions to prepare an exam in future

Statistical Learning Theory

- In the standard supervised learning setting, we assume that both the training data and the test data are drawn *independently* from *identical distributions*.

Statistical Learning Theory

- In the standard supervised learning setting, we assume that both the training data and the test data are drawn *independently* from *identical distributions*.
- This is called the i.i.d assumption.

Statistical Learning Theory

- In the standard supervised learning setting, we assume that both the training data and the test data are drawn *independently* from *identical distributions*.
- This is called the i.i.d assumption.
- Sometimes we can get away with minor violations of the i.i.d assumption

Statistical Learning Theory

- In the standard supervised learning setting, we assume that both the training data and the test data are drawn *independently* from *identical distributions*.
- This is called the i.i.d assumption.
- Sometimes we can get away with minor violations of the i.i.d assumption
- But some violations can cause trouble

Factors that tend to influence the generalizability

- The number of tunable parameters
  - When the number of tunable parameters, sometimes called the degrees of freedom, is large, models tend to be more susceptible to overfitting.

Factors that tend to influence the generalizability

- The number of tunable parameters
  - When the number of tunable parameters, sometimes called the degrees of freedom, is large, models tend to be more susceptible to overfitting.
- The values taken by the parameters.
  - When weights can take a wider range of values, models can be more susceptible to over fitting.

Factors that tend to influence the generalizability

- The number of tunable parameters
  - When the number of tunable parameters, sometimes called the degrees of freedom, is large, models tend to be more susceptible to overfitting.
- The values taken by the parameters.
  - When weights can take a wider range of values, models can be more susceptible to over fitting.
- The number of training examples.
  - It’s trivially easy to overfit a dataset containing only one or two examples even if your model is simple

Model Selection

- In machine learning, we usually select our final model after evaluating several candidate models.

Model Selection

- In machine learning, we usually select our final model after evaluating several candidate models.
- In order to determine the best among our candidate models, we will typically employ a validation set.

Model Selection

- In machine learning, we usually select our final model after evaluating several candidate models.
- In order to determine the best among our candidate models, we will typically employ a validation set.
- Validation Set:
  - We should never rely on the test data for model selection
  - And yet we cannot rely solely on the training data for model selection
  - Solution: split data in three ways: training, testing and validation

Model Selection

- In machine learning, we usually select our final model after evaluating several candidate models.
- In order to determine the best among our candidate models, we will typically employ a validation set.
- Validation Set:
  - We should never rely on the test data for model selection
  - And yet we cannot rely solely on the training data for model selection
  - Solution: split data in three ways: training, testing and validation
- What if we cannot afford to holdout enough data
  - Solution: K-fold Cross Validation

Underfitting or Overfitting

- Underfitting:
  - when our training error and validation error are both substantial but there is a little gap between them
  - If the model is unable to reduce the training error, that could mean that our model is too simple
  - since the generalization gap between our training and validation errors is small, we have reason to believe that we could get away with a more complex model.

Underfitting or Overfitting

- Underfitting:
  - when our training error and validation error are both substantial but there is a little gap between them
  - If the model is unable to reduce the training error, that could mean that our model is too simple
  - since the generalization gap between our training and validation errors is small, we have reason to believe that we could get away with a more complex model.
- Overfitting:
  - when our training error is significantly lower than our validation error

Underfitting or Overfitting

- Underfitting:
  - when our training error and validation error are both substantial but there is a little gap between them
  - If the model is unable to reduce the training error, that could mean that our model is too simple
  - since the generalization gap between our training and validation errors is small, we have reason to believe that we could get away with a more complex model.
- Overfitting:
  - when our training error is significantly lower than our validation error
- Whether we overfit or underfit can depend both on the complexity of our model and the size of the available training datasets

Model Complexity

- An example using polynomials $$\hat{y} = \sum_{i=0}^{d} x^i w_i$$

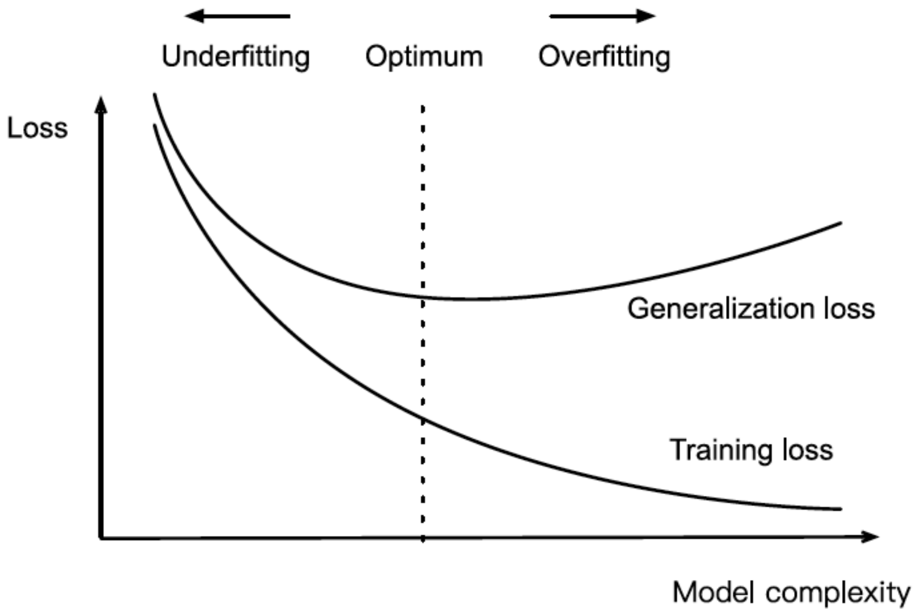

Fig. 6.4.1: Influence of Model Complexity on Underfitting and Overfitting

Data Set Size

- The fewer samples we have in the training dataset, the more likely (and more severely) we are to encounter overfitting.

Data Set Size

- The fewer samples we have in the training dataset, the more likely (and more severely) we are to encounter overfitting.
- As we increase the amount of training data, the generalization error typically decreases.

Data Set Size

- The fewer samples we have in the training dataset, the more likely (and more severely) we are to encounter overfitting.
- As we increase the amount of training data, the generalization error typically decreases.
- Given more data, we might profitably attempt to fit a more complex

model.

Data Set Size

- The fewer samples we have in the training dataset, the more likely (and more severely) we are to encounter overfitting.
- As we increase the amount of training data, the generalization error typically decreases.
- Given more data, we might profitably attempt to fit a more complex

model.

- In part, the current success of deep learning owes to the current abundance of massive datasets due to internet companies, cheap storage, connected devices, and the broad digitization of the economy.

Weight Decay (ℓ_2 regularization)

- Probably the most widely-used technique for regularization

Weight Decay (ℓ_2 regularization)

- Probably the most widely-used technique for regularization
- Intuition: among all functions f, the function $f = 0$ is the simplest.

Weight Decay (ℓ_2 regularization)

- Probably the most widely-used technique for regularization
- Intuition: among all functions f, the function $f = 0$ is the simplest.
- We can measure functions by their proximity to 0.

Weight Decay (ℓ_2 regularization)

- Probably the most widely-used technique for regularization
- Intuition: among all functions f, the function $f = 0$ is the simplest.
- We can measure functions by their proximity to 0.
- Consider a linear function $f(\mathbf{x}) = \mathbf{w}^T \mathbf{x}$

Weight Decay (ℓ_2 regularization)

- Probably the most widely-used technique for regularization
- Intuition: among all functions f, the function $f = 0$ is the simplest.
- We can measure functions by their proximity to 0.
- Consider a linear function $f(\mathbf{x}) = \mathbf{w}^T\mathbf{x}$
- f is simple if its weight vector is small

Weight Decay (ℓ_2 regularization)

- Probably the most widely-used technique for regularization
- Intuition: among all functions f, the function $f = 0$ is the simplest.
- We can measure functions by their proximity to 0.
- Consider a linear function $f(\mathbf{x}) = \mathbf{w}^T\mathbf{x}$
- f is simple if its weight vector is small
- We can measure this via $||\mathbf{w}||^2$

Weight Decay (ℓ_2 regularization)

- Probably the most widely-used technique for regularization
- Intuition: among all functions f, the function $f = 0$ is the simplest.
- We can measure functions by their proximity to 0.
- Consider a linear function $f(\mathbf{x}) = \mathbf{w}^T\mathbf{x}$
- f is simple if its weight vector is small
- We can measure this via $||\mathbf{w}||^2$
- We can add this as penalty term to our loss function

Weight Decay (ℓ_2 regularization)

- Probably the most widely-used technique for regularization
- Intuition: among all functions f, the function $f = 0$ is the simplest.
- We can measure functions by their proximity to 0.
- Consider a linear function $f(\mathbf{x}) = \mathbf{w}^T\mathbf{x}$
- f is simple if its weight vector is small
- We can measure this via $||\mathbf{w}||^2$
- We can add this as penalty term to our loss function
- For linear regression problem, our loss becomes:

$$l(\mathbf{w}, b) + \frac{\lambda}{2}\|w\|^2$$

Regularization constant $\lambda \geq 0$ governs the amount of regularization

Weight Decay (ℓ_2 regularization)

- Why not other norms?

Weight Decay (ℓ_2 regularization)

- Why not other norms?
- In fact, several other choices are valid and popular in statistics.
  - L2 regularized regression is called ridge regression
  - L1 regularized regression is called lasso regression

Weight Decay (ℓ_2 regularization)

- Why not other norms?
- In fact, several other choices are valid and popular in statistics.
  - L2 regularized regression is called ridge regression
  - L1 regularized regression is called lasso regression
- What is the SGD update for L2 regularized regression?

Dropout

- What constitutes to simple model?
  - Small number of features
  - Small norm of the basis functions
  - The function should be robust under small changes in the input

Dropout

- What constitutes to simple model?
  - Small number of features
  - Small norm of the basis functions
  - The function should be robust under small changes in the input
- In 1995, Christopher Bishop showed that training with input noise is equivalent to Tikhonov regularization

Dropout

- What constitutes to simple model?
  - Small number of features
  - Small norm of the basis functions
  - The function should be robust under small changes in the input
- In 1995, Christopher Bishop showed that training with input noise is equivalent to Tikhonov regularization
- In 2014, Siravastana applied Bishop’s idea to internal layers of the network
  - Dropout – widely used in neural networks
  - On each iteration, drop out zeroes out some fraction of the nodes in each layer before calculating the subsequent layer in training

Dropout

- The key challenge: how to inject noise without introducing bias?
  - i.e. we want to perturb the inputs to each layer during training in such a way that the expected value of the layer is equal to the value it would have taken had we not introduced any noise at all.

Dropout

- The key challenge: how to inject noise without introducing bias?
  - i.e. we want to perturb the inputs to each layer during training in such a way that the expected value of the layer is equal to the value it would have taken had we not introduced any noise at all.
- Dropout at each node with probability p is defined as:

$$h' = \begin{cases} 0 & \text{with probability } p \\ \frac{h}{1-p} & \text{otherwise} \end{cases}$$

Dropout

- The key challenge: how to inject noise without introducing bias?
  - i.e. we want to perturb the inputs to each layer during training in such a way that the expected value of the layer is equal to the value it would have taken had we not introduced any noise at all.
- Dropout at each node with probability p is defined as:

$$h' = \begin{cases} 0 & \text{with probability } p \\ \frac{h}{1-p} & \text{otherwise} \end{cases}$$

- Show the expectations remain unchanged!

Dropout in Practice

- In the image below h_2 and h_5 are removed
- This way, calculation of the output cannot be overly dependent on any one element of h_1, h_2, h_3, h_4, h_5.

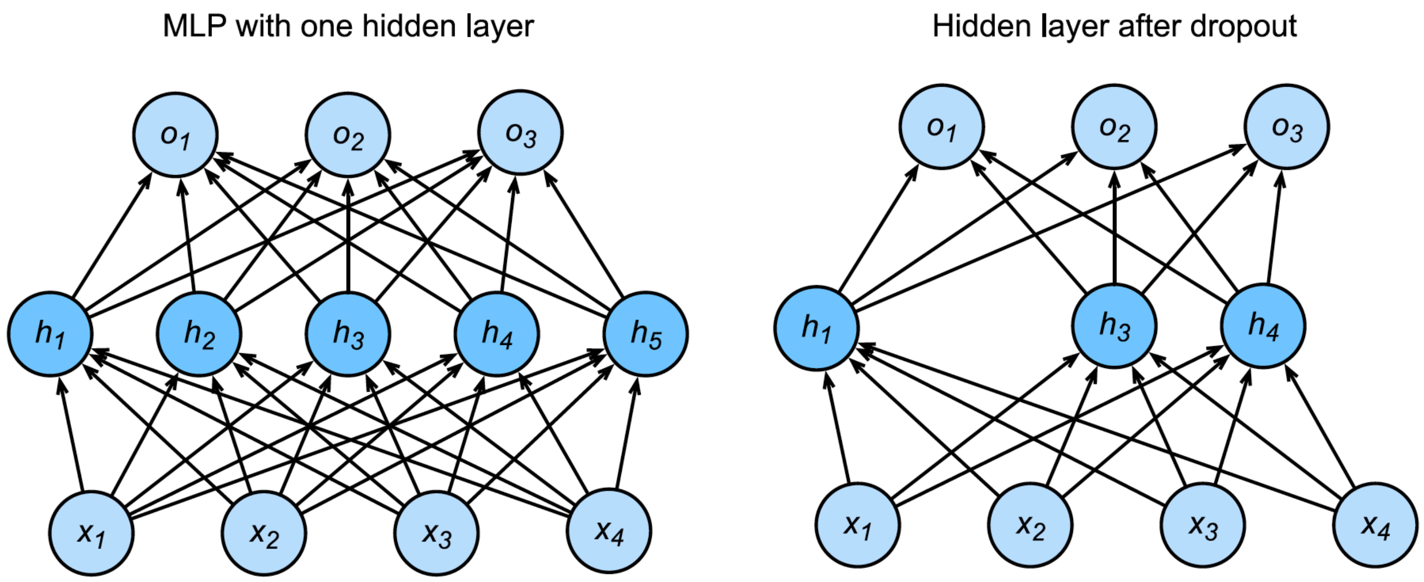

Fig. 6.6.1: MLP before and after dropout

- At test time, we typically do not use dropout.

Implementation of dropout

- From scratch
- concise