Lecture 8
Basics for Machine Learning

Lin ZHANG, PhD
School of Software Engineering
Tongji University
Spring 2016
Deep Learning
With massive amounts of computational power, machines can now recognize objects and translate speech in real time. Artificial intelligence is finally getting smart.

Temporary Social Media
Messages that quickly self-destruct could enhance the privacy of online communications and make people freer to be spontaneous.

Prenatal DNA Sequencing
Reading the DNA of fetuses will be the next frontier of the genomic revolution. But do you really want to know about the genetic problems or musical aptitude of your unborn child?

Additive Manufacturing
Skeptical about 3-D printing? GE, the world's largest manufacturer, is on the verge of using the technology to make jet parts.

Baxter: The Blue-Collar Robot
Rodney Brooks's newest creation is easy to interact with, but the complex innovations behind the robot show just how hard it is to get along with people.

Memory Implants
A maverick neuroscientist believes he has deciphered the code by which the brain forms long-term memories. Next: testing a prosthetic implant for people suffering from long-term memory loss.

Smart Watches
The designers of the Pebble watch realized that a mobile phone is more useful if you don't have to take it out of your pocket.

Ultra-Efficient Solar Power
Doubling the efficiency of a solar cell would completely change the economics of renewable energy. Nanotechnology just might make it possible.

Big Data from Cheap Phones
Collecting and analyzing information from simple cell phones can provide surprising insights into how people move about and behave - and even help us understand the spread of diseases.

Supergrids
A new high-power circuit breaker could finally make highly efficient DC power grids practical.
Outline

• Basic concepts
• Linear regression
• Logistic regression
• Softmax regression
• Neural network
• Convolutional neural network (CNN)
Basic concepts

• Supervised learning
  – It will infer a function from labeled training data
  – The training data consists of a set of training examples
  – Each example is a pair consisting of an input object (typically a vector) and a desired output value (also called the supervisory signal)

• Unsupervised learning
  – Trying to find hidden structure in unlabeled data
  – Since the examples given to the learner are unlabeled, there is no error or reward signal to evaluate a potential solution
  – Such as PCA, K-means (a clustering algorithm)
Basic concepts

• Training sample and training set

\[ (x^{(i)}, y^{(i)}) \]

One training sample

feature of sample \( i \)

value of sample \( i \)

\[ \{(x^{(i)}, y^{(i)}) : i = 1, \ldots, m\} \]

Training set comprising \( m \) samples
Basic concepts

• Training sample and training set

• Test set
  – A test set is a set of data that is independent of the training data, but that follows the same probability distribution as the training data
  – Used only to assess the performance of a fully specified classifier
Basic concepts

• Training sample and training set
• Test set
• Validation set
  – In order to avoid overfitting, when any classification parameter needs to be adjusted, it is necessary to have a validation set
  – The training set is used to train the candidate algorithms, while the validation set is used to compare their performances and decide which one to take
Basic concepts

• **Underfitting**
  – The learned model cannot fit the training data very well
Basic concepts

• Underfitting

• Overfitting
  – It occurs when a statistical model describes random error or noise instead of the underlying relationship
  – It generally occurs when a model is excessively complex, such as having too many parameters relative to the number of observations
  – A model that has been overfit will generally have poor predictive performance, as it can exaggerate minor fluctuations in the data
Basic concepts

• Underfitting
• Overfitting
• Generalization
  – Refers to the performance of the learned model on new, previously unseen examples, such as the test set
Basic concepts

• Underfitting
• Overfitting
• Generalization

Example: Linear regression (housing prices)
Basic concepts

- Underfitting
- Overfitting
- Generalization

Example: Logistic regression

\[ h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2) \]
\[ g(x) = \text{sigmoid function} \]

"Underfit"

"Overfit"

http://blog.csdn.net/zouxy09
Basic concepts

- Underfitting
- Overfitting
- Generalization
- Capacity
  - Measures the complexity, expressive power, richness, or flexibility of a classification algorithm

\[ y^* = b + \omega x, \quad y^* = b + \omega_1 x + \omega_2 x^2, \quad y^* = b + \sum_{i=1}^{10} \omega_i x^i \]

higher capacity
Basic concepts

[Graphs illustrating underfitting, appropriate model complexity, and overfitting]

higher capacity
Basic concepts
Outline

• Basic concepts
• Linear regression
• Logistic regression
• Softmax regression
• Neural network
• Convolutional neural network (CNN)
Linear regression

• Our goal in linear regression is to predict a target value \( y \) from a vector of input values \( x \in \mathbb{R}^n \); we use a linear function \( h \) as the model

• At the training stage, we aim to find \( h(x) \) so that we have \( h(x^{(i)}) \approx y^{(i)} \) for each training sample

• We suppose that \( h \) is a linear function, so

\[
h_\theta(x) = \sum_j \theta_j x_j = \theta^T x
\]
Linear regression

• Then, our task is to find a choice of $\theta$ so that $h_{\theta}(x^{(i)})$ is as close as possible to $y^{(i)}$

The cost function can be written as,

$$J(\theta) = \frac{1}{2} \sum_i \left( \theta^T x^{(i)} - y^{(i)} \right)^2$$

Then, the task at the training stage is to find

$$\theta^* = \arg \min_{\theta} \frac{1}{2} \sum_i \left( \theta^T x^{(i)} - y^{(i)} \right)^2$$

In general case, it can be solved by gradient descent method
Linear regression

• Gradient descent
  – It is a first-order optimization algorithm
  – To find a local minimum of a function, one takes steps proportional to the negative of the gradient of the function at the current point
  – One starts with a guess $\theta_0$ for a local minimum of $J(\theta)$ and considers the sequence such that

$$\theta_{n+1} := \theta_n - \alpha \nabla_{\theta} J(\theta)_{|\theta=\theta_n}$$

where $\alpha$ is called as learning rate
Linear regression

- Gradient descent
Linear regression

• Gradient descent

Repeat until convergence ($J(\theta)$ will not reduce anymore)

\[
\theta_{n+1} := \theta_n - \alpha \nabla_{\theta} J(\theta) \big|_{\theta = \theta_n}
\]

GD is a general optimization solution; for a specific problem, the key step is how to compute gradient
Linear regression

• Gradient of the cost function of linear regression

\[ J(\theta) = \frac{1}{2} \sum_i \left( \theta^T x^{(i)} - y^{(i)} \right)^2 \]

The gradient is,

\[ \nabla_\theta J(\theta) = \sum_i \left( \theta^T x^{(i)} - y^{(i)} \right) x^{(i)} \]
Linear regression

- Some variants of gradient descent
  - The ordinary gradient descent algorithm looks at every sample in the **entire** training set on every step; it is also called as **batch gradient descent**
  - **Stochastic gradient descent (SGD)** repeatedly run through the training set, and each time when we encounter a training sample, we update the parameters according to the gradient of the error w.r.t that single training sample only

Repeat until convergence
{
  for \( i = 1 \) to \( m \) (\( m \) is the number of training samples)
  {
    \[ \theta_{n+1} := \theta_n - \alpha \left( \theta_n^T x^{(i)} - y^{(i)} \right) x^{(i)} \]
  }
}
Linear regression

• Some variants of gradient descent
  – The ordinary gradient descent algorithm looks at every sample in the entire training set on every step; it is also called as batch gradient descent
  – **Stochastic gradient descent (SGD)** repeatedly run through the training set, and each time when we encounter a training sample, we update the parameters according to the gradient of the error w.r.t that single training sample only
  – **Minibatch SGD**: it works identically to SGD, except that it uses more than one training samples to make each estimate of the gradient
Linear regression

• More concepts
  – $m$ Training samples can be divided into $N$ minibatches
  – When the training sweeps all the batches, we way we complete one **epoch** of training process; for a typical training process, several epochs are usually required

```plaintext
ePOCHS  = 10;
nUmBATCHES  = m/N;
WHILE epochIndex < epochs AND NOT converged
{
  FOR minibatchIndex = 1 TO numBATCHES
  {
    UPDATE the model parameters based on this batch
  }
}
```
Outline

- Basic concepts
- Linear regression
- Logistic regression
- Softmax regression
- Neural network
- Convolutional neural network (CNN)
Logistic regression

- Logistic regression is used for binary classification.
- It squeezes the linear regression $\theta^T x$ into the range (0, 1), thus the prediction result can be interpreted as probability.
- We will try to learn a function of the form

$$P(y = 1 | x) = h_\theta(x) = \frac{1}{1 + \exp(-\theta^T x)}$$

$$P(y = 0 | x) = 1 - P(y = 1 | x) = 1 - h_\theta(x)$$

Function $\sigma(z) = \frac{1}{1 + \exp(-z)}$ is called as sigmoid or logistic function.
Logistic regression

One property of the sigmoid function

$$\sigma'(z) = \sigma(z)(1 - \sigma(z))$$
Logistic regression

• The hypothesis model can be written neatly as

$$P(y \mid x; \theta) = (h_\theta(x))^y (1 - h_\theta(x))^{1-y}$$

• Our goal is to search for a value $\theta$ so that $h_\theta(x)$ is large when $x$ belongs to “1” class and small when $x$ belongs to “0” class

Thus, given a training set with binary labels $\{(x^{(i)}, y^{(i)}): i = 1, \ldots, m\}$, we want to maximize,

$$\prod_{i=1}^{m} \left( h_\theta(x^{(i)}) \right)^{y^{(i)}} (1 - h_\theta(x^{(i)}))^{1-y^{(i)}}$$

Equivalent to maximize,

$$\sum_{i=1}^{m} y^{(i)} \log(h_\theta(x^{(i)})) + (1 - y^{(i)}) \log(1 - h_\theta(x^{(i)}))$$
Logistic regression

• Thus, the cost function for the logistic regression is (we want to minimize),

\[
J(\theta) = -\sum_{i=1}^{m} y^{(i)} \log(h_{\theta}(x^{(i)})) + (1 - y^{(i)}) \log(1 - h_{\theta}(x^{(i)}))
\]

To solve it with gradient descent, gradient needs to be computed,

\[
\nabla_{\theta} J(\theta) = \sum_{i=1}^{m} x^{(i)} \left( h_{\theta}(x^{(i)}) - y^{(i)} \right)
\]

Can you verify?
Logistic regression

- Exercise
  - Use logistic regression to perform digital classification
Outline

• Basic concepts
• Linear regression
• Logistic regression
• Softmax regression
• Neural network
• Convolutional neural network (CNN)
Softmax regression

- Softmax operation
  - It squashes a $K$-dimensional vector $\mathbf{z}$ of arbitrary real values to a $K$-dimensional vector $\sigma(\mathbf{z})$ of real values in the range $(0, 1)$. The function is given by,

$$
\sigma(\mathbf{z})_j = \frac{\exp(\mathbf{z}_j)}{\sum_{k=1}^{K} \exp(\mathbf{z}_k)}
$$

- Since the components of the vector $\sigma(\mathbf{z})$ sum to one and are all strictly between 0 and 1, they represent a categorical probability distribution.
Softmax regression

• For multiclass classification, given a test input $x$, we want our hypothesis to estimate $p(y = k \mid x)$ for each value $k=1,2,\ldots,K$

• The hypothesis should output a $K$-dimensional vector giving us $K$ estimated probabilities. It takes the form,

$$h_\theta(x) = \begin{bmatrix}
  p(y = 1 \mid x; \theta) \\
  p(y = 2 \mid x; \theta) \\
  \vdots \\
  p(y = K \mid x; \theta)
\end{bmatrix} = \frac{1}{\sum_{j=1}^{K} \exp\left((\theta^{(j)})^T x\right)} \begin{bmatrix}
  \exp\left((\theta^{(1)})^T x\right) \\
  \exp\left((\theta^{(2)})^T x\right) \\
  \vdots \\
  \exp\left((\theta^{(K)})^T x\right)
\end{bmatrix}$$
Softmax regression

- In softmax regression, for each training sample we have,

\[
p(y^{(i)} = k \mid x^{(i)}; \theta) = \frac{\exp\left((\theta^{(k)})^T x^{(i)}\right)}{\sum_{j=1}^{K} \exp\left((\theta^{(j)})^T x^{(i)}\right)}
\]

At the training stage, we want to maximize \( p(y^{(i)} = k \mid x^{(i)}; \theta) \) for each training sample for the correct label \( k \)}
Softmax regression

• Cost function for softmax regression

\[ J(\theta) = -\sum_{i=1}^{m} \sum_{k=1}^{K} 1\{y^{(i)} = k\} \log \frac{\exp \left( \left( \theta^{(k)} \right)^T x^{(i)} \right)}{\sum_{j=1}^{K} \exp \left( \left( \theta^{(j)} \right)^T x^{(i)} \right)} \]

where \(1\{.\}\) is an indicator function

• Gradient of the cost function

\[ \nabla_{\theta^{(k)}} J(\theta) = -\sum_{i=1}^{m} \left[ x^{(i)} \left( 1\{y^{(i)} = k\} - p \left( y^{(i)} = k \mid x^{(i)}; \theta \right) \right) \right] \]

Can you verify?
Softmax regression

• Redundancy of softmax regression parameters

Subtract a fixed vector $\psi$ from every $\theta^{(j)}$, we have

$$p\left( y^{(i)} = k \mid x^{(i)}; \theta \right) = \frac{\exp\left( (\theta^{(k)} - \psi)^T x^{(i)} \right)}{\sum_{j=1}^{K} \exp\left( (\theta^{(j)} - \psi)^T x^{(i)} \right)}$$

$$= \frac{\exp\left( \left( \theta^{(k)} \right)^T x^{(i)} \right) \exp\left( -\psi^T x^{(i)} \right)}{\sum_{j=1}^{K} \exp\left( \left( \theta^{(j)} \right)^T x^{(i)} \right) \exp\left( -\psi^T x^{(i)} \right)}$$

$$= \frac{\exp\left( \left( \theta^{(k)} \right)^T x^{(i)} \right)}{\sum_{j=1}^{K} \exp\left( \left( \theta^{(j)} \right)^T x^{(i)} \right)}$$
Softmax regression

• Redundancy of softmax regression parameters

• So, in most cases, instead of optimizing $K \cdot n$ parameters, we can set $\theta^{(K)} = 0$ and optimize only w.r.t the $(K - 1) \cdot n$ remaining parameters
Outline

• Basic concepts
• Linear regression
• Logistic regression
• Softmax regression
• Neural network
• Convolutional neural network (CNN)
Neural networks

• It is one way to solve a supervised learning problem given labeled training examples \( \{x^{(i)}, y^{(i)}\} (i = 1, \ldots, m) \)

• Neural networks give a way of defining a complex, non-linear form of hypothesis \( h_{W,b}(x) \), where \( W \) and \( b \) are the parameters we need to learn from training samples
Neural networks

- A single neuron

- $x_1$, $x_2$, and $x_3$ are the inputs, +1 is the intercept term, $h_{W,b}(x)$ is the output of this neuron

$$h_{W,b}(x) = f \left( W^T x \right) = f \left( \sum_{i=1}^{3} W_i x_i + b \right)$$

where $f(\cdot)$ is the activation function
Neural networks

- Commonly used activation functions
  - Sigmoid function
    \[ f(z) = \frac{1}{1 + \exp(-z)} \]
  - Tanh function
    \[ f(z) = \tanh(z) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}} \]
  - Rectified linear unit (ReLU)
    \[ f(z) = \max(0, z) \]
Neural networks

• A neural network is composed by hooking together many simple neurons
• The output of a neuron can be the input of another
• Example, a three layers neural network,
Neural networks

• Terminologies about the neural network
  – The circle labeled +1 are called bias units
  – The leftmost layer is called the input layer
  – The rightmost layer is the output layer
  – The middle layer of nodes is called the hidden layer
    » In our example, there are 3 input units, 3 hidden units, and 1 output unit
  – We denote the activation (output value) of unit $i$ in lay $l$ as $a_i^{(l)}$
Neural networks

\[ a_1^{(2)} = f \left( W_{11}^{(1)} x_1 + W_{12}^{(1)} x_2 + W_{13}^{(1)} x_3 + b_1^{(1)} \right) \]

\[ a_2^{(2)} = f \left( W_{21}^{(1)} x_1 + W_{22}^{(1)} x_2 + W_{23}^{(1)} x_3 + b_2^{(1)} \right) \]

\[ a_3^{(2)} = f \left( W_{31}^{(1)} x_1 + W_{32}^{(1)} x_2 + W_{33}^{(1)} x_3 + b_3^{(1)} \right) \]

\[ h_{W,b}(x) = a_1^{(3)} = f \left( W_{11}^{(2)} a_1^{(2)} + W_{12}^{(2)} a_2^{(2)} + W_{13}^{(2)} a_3^{(2)} + b_1^{(2)} \right) \]
Neural networks

- Neural networks can have multiple outputs
- Usually, we can add a softmax layer as the output layer to perform multiclass classification
Neural networks

- At the testing stage, given a test input $x$, it is straightforward to evaluate its output.
- At the training stage, given a set of training samples, we need to train $W$ and $b$.
  - The key problem is how to compute the gradient.
  - Back propagation algorithm.
Outline

• Basic concepts
• Linear regression
• Logistic regression
• Softmax regression
• Neural network
• Convolutional neural network (CNN)
Convolutional neural network

• Specially designed for data with grid-like structures (LeCun et al. 98)
  – 1D grid: sequential data
  – 2D grid: image
  – 3D grid: video, 3D image volume

• Beat all the existing computer vision technologies on object recognition on ImageNet challenge with a large margin in 2012
• Problems of fully connected networks
  – Every output unit interacts with every input unit
  – The number of weights grows largely with the size of the input image
  – Pixels in distance are less correlated
Convolutional neural network

- Problems of fully connected networks
Convolutional neural network

• One simple solution is locally connected neural networks
  – Sparse connectivity: a hidden unit is only connected to a local patch; (weights connected to the patch are called filter or kernel)
  – It is inspired by biological systems, where a cell is sensitive to a small sub-region of the input space, called a receptive field. Many cells are tiled to cover the entire visual field
Convolutional neural network

- One simple solution is locally connected neural networks

Example: 1000x1000 image
1M hidden units
Filter size: 10x10
100M parameters

Ranzato CVPR’13
Convolutional neural network

• One simple solution is locally connected neural networks
  – The learned filter is a spatially local pattern
  – A hidden node at a higher layer has a larger receptive field in the input
  – Stacking many such layers leads to “filters” (not anymore linear) which become increasingly “global”
Convolutional neural network

- Convolution
  - Computing the responses at hidden nodes is equivalent to convoluting the input image $x$ with a learned filter $w$

$$\text{net}[i, j] = (x \ast w)[i, j] = \sum_m \sum_n x[m, n]w[i - m, j - n]$$
Convolutional neural network

• Downsampled convolution layer (optional)
  – To reduce computational cost, we may want to skip some positions of the filter and sample only every $s$ pixels in each direction. A downsampled convolution function is defined as

  $\text{net}(i, j) = (x \ast w)[i \times s, j \times s]$

  – $s$ is referred as the stride of this downsampled convolution
Convolutional neural network

• Multiple filters
  – Multiple filters generate multiple feature maps
  – Detect the spatial distributions of multiple visual patterns
Convolutional neural network

- 3D filtering when input has multiple feature maps
Convolutional neural network

- Convolutional layer

Ranzato CVPR’13
Convolutional neural network

• To the convolution responses, we then perform nonlinear activation
  – ReLU
  – Tanh
  – Sigmoid
Convolutional neural network

• Local contrast normalization (optional)
  – Normalization can be done within a neighborhood along both spatial and feature dimensions

\[
h_{i+1,x,y,k} = \frac{h_{i,x,y,k} - m_{i,N(x,y,k)}}{\sigma_{i,N(x,y,k)}}
\]
Convolutional neural network

• Then, we perform pooling
  – Max-pooling partitions the input image into a set of rectangles, and for each sub-region, outputs the maximum value
  – Non-linear down-sampling
  – The number of output maps is the same as the number of input maps, but the resolution is reduced
  – Reduce the computational complexity for upper layers and provide a form of translation invariance
  – Average pooling can also be used
Convolutional neural network

• Then, we perform pooling
Convolutional neural network

• Typical architecture of CNN
  – Convolutional layer increases the number of feature maps
  – Pooling layer decreases spatial resolution
  – LCN and pooling are optional at each stage
Convolutional neural network

• Typical architecture of CNN

Example with only two filters.

Ranzato CVPR’13
Convolutional neural network

• Typical architecture of CNN

One stage (zoom)

A hidden unit in the first hidden layer is influenced by a small neighborhood (equal to size of filter).

Ranzato CVPR’13
Convolutional neural network

• Typical architecture of CNN

After a few stages, residual spatial resolution is very small. We have learned a descriptor for the whole image. Ranzato CVPR’13
Convolutional neural network

- Typical architecture of CNN
Convolutional neural network

• Different CNN structures for image classification
  – AlexNet
  – Clarifai
  – Overfeat
  – VGG
  – DeepImage of Baidu
  – Network-in-network
  – GoogLeNet
Convolutional neural network

- AlexNet: CNN for object recognition on ImageNet challenge
  - Krizhevsky, Sutskever, and Hinton, NIPS 2012
  - Trained on one million images of 1000 categories collected from the web with two GPU. 2GB RAM on each GPU. 5GB of system memory
  - Training lasts for one week
  - Google and Baidu announced their new visual search engines with the same technology six months after that
  - Google observed that the accuracy of their visual search engine was doubled
Convolutional neural network

- ImageNet
  - http://www.image-net.org/
Convolutional neural network

• Architecture of AlexNet
  – 5 convolutional layers and 2 fully connected layers for learning features
  – Max-pooling layers follow first, second, and fifth convolutional layers
Convolutional neural network

- Architecture of AlexNet
  - The first time deep model is shown to be effective on large scale computer vision task
  - The first time a very large scale deep model is adopted
  - GPU is shown to be very effective on this large deep model
Convolutional neural network

- Opensource platforms for CNN
  - CAFFE (support GPU), http://caffe.berkeleyvision.org/
  - Cxxnet (support GPU), https://github.com/dmlc/cxxnet
  - MatConvNet (support GPU), http://www.vlfeat.org/matconvnet/
  - Theano (support GPU), http://deeplearning.net/software/theano/
Thanks for your attention