MNIST在Keras下的幾種學習架構

Presented by David Tung

Outlines

Machine learning basic

- underfitting/overfitting
- stochastic gradient descent
- softmax/cross-entropy

Program structure

MNIST and its network models

- linear based classifier
- multiple layer perceptron
- convolutional neural network

General resource

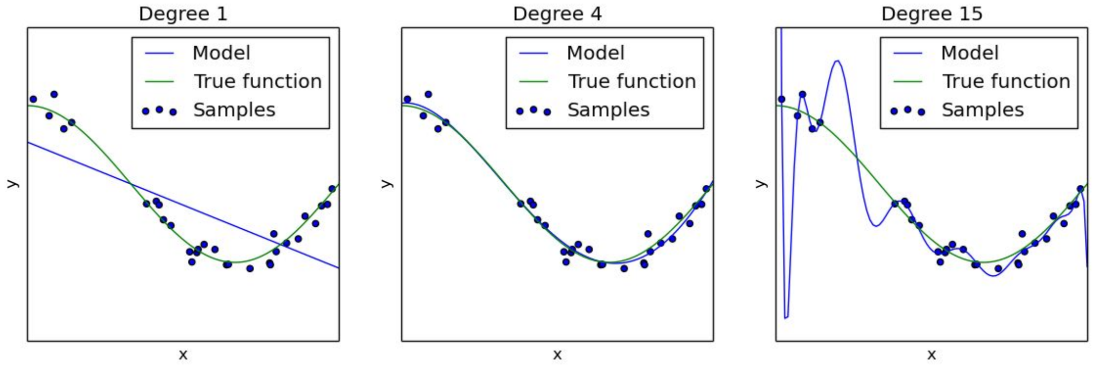
Tensorflow : https://www.tensorflow.org/get_started/

Keras: https://keras.io/getting-started/sequential-model-guide/

Deep learning book: http://www.deeplearningbook.org/

Machine learning basic - capacity

underfitting/overfitting



Source:http://scikit-learn.org/stable/auto_examples/model_selection/plot_underfitting_overfitting.html

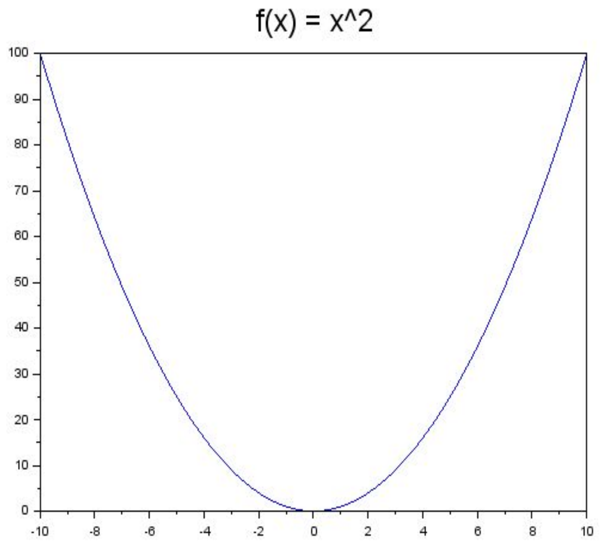
Machine learning basic - stochastic gradient descent

Gradient descent

$$\mathbf{x} = \mathbf{x} - \eta \cdot \nabla_{\mathbf{x}} f(\mathbf{x})$$

Example:

$$f(x) = x^2$$

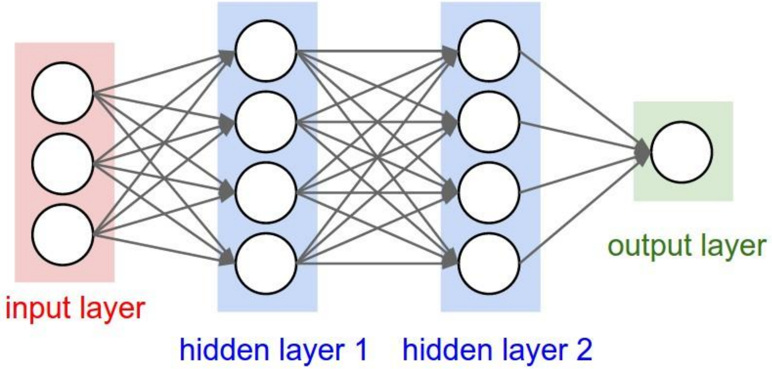


Stochastic gradient descent

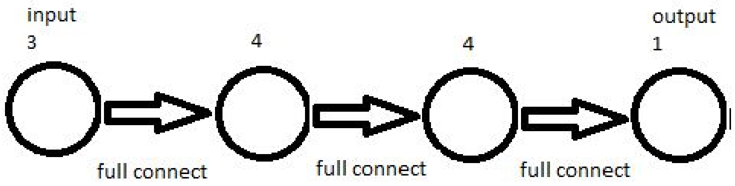
$$\omega = \omega - \eta \cdot \sum_{i=1}^{n} \nabla_{\omega} f(\omega; \mathbf{x}_i)$$

Machine learning basic - feedforward network

FNN (resource: http://cs231n.github.io/neural-networks-1/)

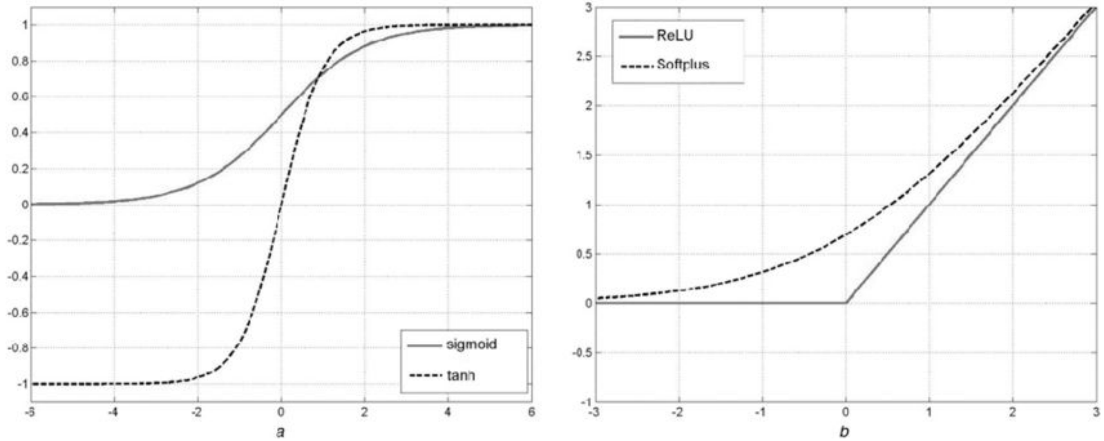


SImplified graph



Machine linearing basic - activation functions

sigmoid/tanh/relu/softplus



Resource:

https://www.researchgate.net/publication/283433254_Driving_posture_recognition_by_convolutional_neural_networks

Machine learning basic - categorical loss function

Sigmoid and negative log-likelihood

sigmoid:

$$y = \sigma(z) = \frac{1}{1+e^{-z}}$$

$$\frac{d\sigma(z)}{dz} = \frac{e^{-z}}{(1+e^{-z})^2} = \sigma(z)(1-\sigma(z))$$

negative log-likelihood:

$$L(z) = -\hat{y}\log y - (1-\hat{y})\log(1-y)$$

if $\hat{y} = 1$

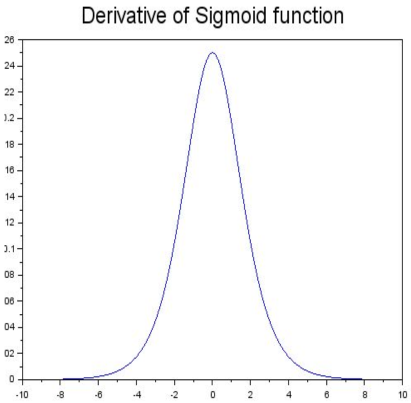
$$L(z) = -\log y = -\log\sigma(z) = \log(1+e^{-z})$$

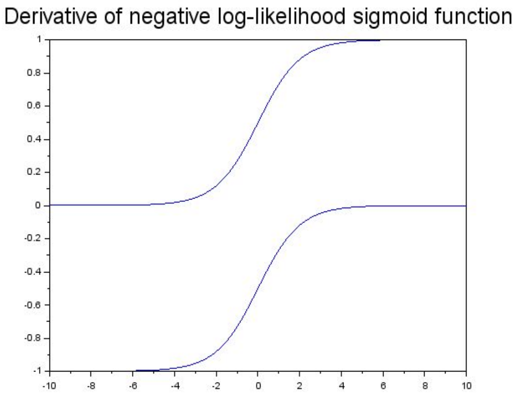
$$\frac{dL(z)}{dz} = \frac{-e^{-z}}{1+e^{-z}} = \sigma(z) - 1$$

if $\hat{y} = 0$

$$L(z) = -\log(1-y) = -\log(1-\sigma(z)) = \log(1+e^{-z}) + z$$

$$\frac{dL(z)}{dz} = \frac{-e^{-z}}{1+e^{-z}} + 1 = \sigma(z)$$





Machine learning basic - categorical loss function

Softmax and cross-entropy

softmax:

$$y_i = s_i(\mathbf{z}) = \frac{e^{z_i}}{\sum_{k=1}^{n} e^{z_k}}$$

cross-entropy

$$L(\mathbf{z}) = -\sum_{k=1}^{n} \hat{y}_k \log y_k$$

one-hot code of $\hat{\mathrm{y}}$, let one to be index d:

$$\frac{\partial L(\mathbf{z})}{\partial z_i} = \frac{\partial - \log y_d}{\partial z_i}$$

if $i = d$

$$\frac{\partial L(\mathbf{z})}{\partial z_d} = s_d(\mathbf{z}) - 1$$

if $i \neq d$

$$\frac{\partial L(\mathbf{z})}{\partial z_i} = s_i(\mathbf{z})$$

Machine learning basic - backpropagation

Gradient + chain rule

Reference:
https://www.facebook.com/groups/1948109392087807/permalink/1980996765465736/

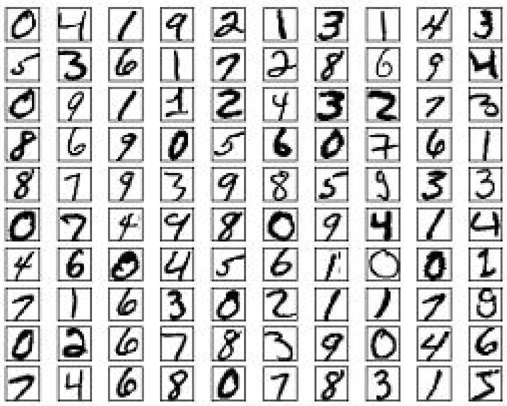
Program structure

- Dataset preprocessing
- Model construction
- Loss function/Optimization
- Training/Validation/Test
- Display

MNIST data set

MNIST is a data set of images of handwritten digits.

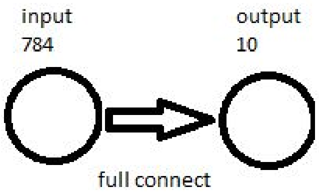
Every digit is represented by 28x28 pixels; each pixel is 256 gray levels.



Resource: http://neuralnetworksanddeeplearning.com/chap1.html

Linear based classifier

Linear based classifier model:



Modified from:
https://github.com/fchollet/keras/blob/master/examples/mnist_mlp.py

Reference: https://www.tensorflow.org/get_started/mnist/beginners

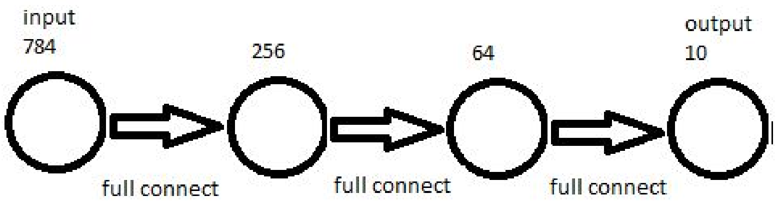
Key technology: softmax/cross-entropy/stochastic gradient descent

Keras model:

```
# model
model = Sequential()
model.add(Dense(10, activation='softmax', input_shape=(784,)))
```

Multiple layer perceptron

MLP model:



Modified from:

https://github.com/fchollet/keras/blob/master/examples/mnist_mlp.py

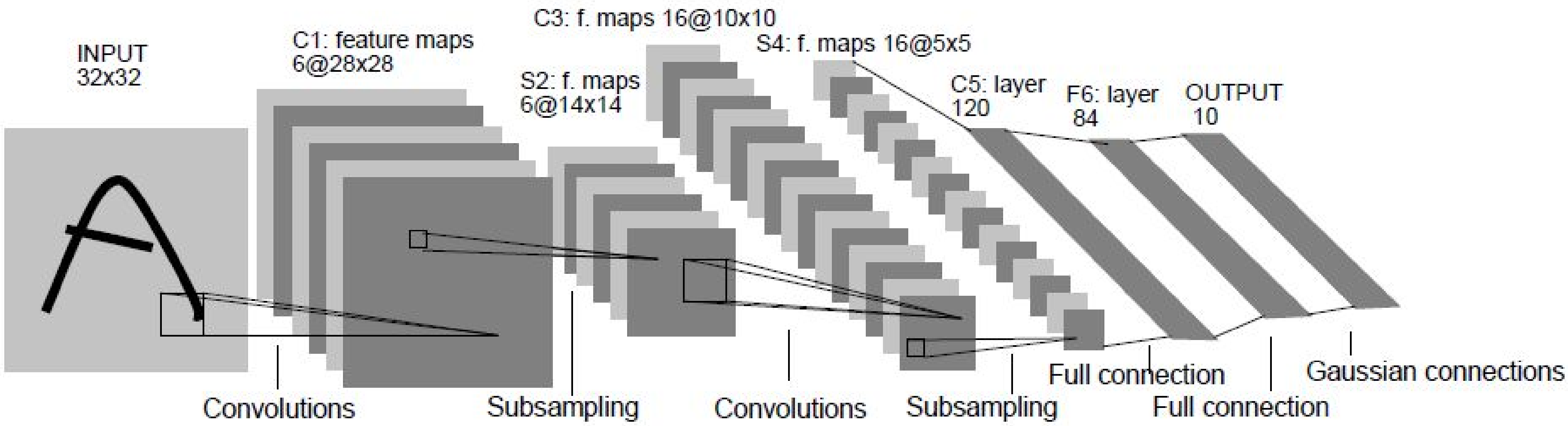
Key technology: backpropagation/ReLU

Keras model:

```
# model
model = Sequential()
model.add(Dense(256, activation='relu', input_shape=(784,)))
model.add(Dense(64, activation='relu'))
model.add(Dense(10, activation='softmax'))
```

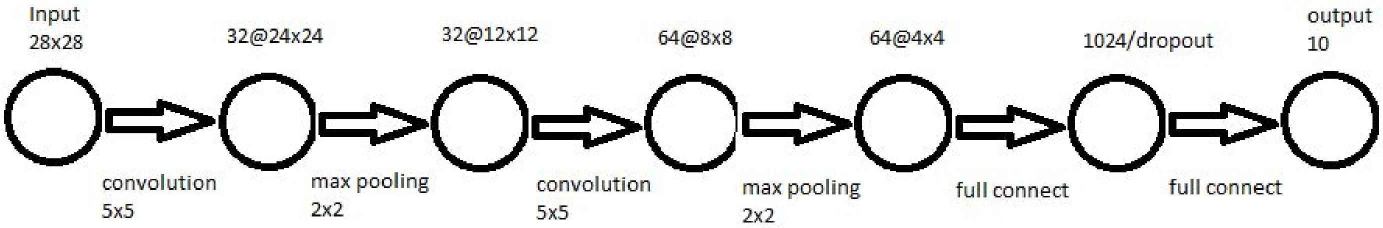
Convolutional neural network - LeNet5

Reference:
http://www.iro.umontreal.ca/~lisa/bib/pub_subject/finance/pointeurs/lecun-98.pdf



CNN complexity

Our CNN model:



Complexity:

$$
\begin{aligned}
1 \cdot (5^2 + 1) \cdot 24^2 \cdot 32 &= 479,232 \\
32 \cdot (5^2 + 1) \cdot 8^2 \cdot 64 &= 3,407,872 \\
(4^2 \cdot 64 + 1) \cdot 1024 &= 1,049,600 \\
(1024 + 1) \cdot 10 &= 10,250 \\
total &= 4,946,964
\end{aligned}
$$

Convolutional neural network

Modified from:
https://github.com/fchollet/keras/blob/master/examples/mnist_cnn.py

Reference: https://www.tensorflow.org/get_started/mnist/pros

Key technology: convolutional kernel/padding/stride/pooling/features/dropout

Kears model:

```
# model
model = Sequential()
model.add(Conv2D(32, kernel_size=(5, 5),
                 activation='relu',
                 input_shape=input_shape))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(64, (5, 5), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Flatten())
model.add(Dense(1024, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(num_classes, activation='softmax'))
```

Appendix - more study notes

Backpropagation by Computational Graph:
https://www.facebook.com/groups/1948109392087807/permalink/1980996765465736/

Feedforword Networks:
https://www.facebook.com/groups/1948109392087807/permalink/1982550641977015/

Convolutional Neural Network:
https://www.facebook.com/groups/1948109392087807/permalink/1987843988114347/

Recurrent Neural Networks (RNN) and Its Applications:
https://www.facebook.com/groups/1948109392087807/permalink/2004972959734783/

Generative Adversarial Network (GAN) and Its Applications:
https://www.facebook.com/groups/1948109392087807/permalink/2025433097688769/