



Regression-based ML Algorithms (LR and NN)

Dr Paul Yoo

Dept CSIS
24/10/19

1



Timetable

Week	Date	Lecture (G12, Torrington, UCL)	Lab (MAL 414-417)
1	03/10/19	Introduction, Workflow and Loading	Loading data and descriptive statistics
2	10/10/19	Data pre-processing	Preparing data
3	17/10/19	Feature selection and re-sampling	Selecting features and re-sampling
4	24/10/19	DT and RF	Comparing ML algorithms
5	31/10/19	LR and NN, and Eval.	Automating the process
6	07/11/19	TensorFlow, Keras and Project Briefing	MLP with Keras
7	14/11/19	Project (30%) Labs 414-417	
8	21/11/19		
9	28/11/19	Image processing	Deep learning - CNN
10	05/12/19	RNN and sequential data	Deep learning – RNN & LSTM
11	12/12/19	Real-life case	Deep learning

Autumn term: 30/09/2019 to 13/12/2019

2

Quiz



What is the C4.5 used to build?

1. Decision Tree
2. Regression analysis
3. Neural Network
4. Association Rule
5. Random Forest
6. None of above

3

Quiz



Which machine learning is most suitable for categorial variables?

1. Association Rule
2. Regression analysis
3. Neural Network
4. Statistical
5. Decision Tree
6. None of above

4

Quiz



What does decision node illustrate in a decision tree?

1. Data value description
2. Class of instance
3. Test specification
4. Data process description
5. All of above

5

Quiz



Which of the following is/are true about bagging trees?

1. In bagging trees, individual trees are independent of each other
 2. Bagging is the method for improving the performance by aggregating the results of weak learners
- A. 1
B. 2
C. 1 and 2
D. None of above

6

Quiz



What is the main weakness of Random Forest?

7

Quiz



In Random Forest you can generate hundreds of trees (say T_1, T_2, \dots, T_n) and then aggregate the results of these tree. Which of the following is true about individual tree (T_k) in Random Forest?

1. Individual tree is built on a subset of the features
2. Individual tree is built on all the features
3. Individual tree is built on a subset of observations
4. Individual tree is built on full set of observations

- A. 1 and 3
 B. 1 and 4
 C. 2 and 4
 D. None of above

Random forest is based on bagging concept, that consider part of sample and part of feature for building the individual trees.

8

Quiz



Random Forest reduces variance from Decision Tree by adopting bagging and random feature selection concepts. Is there a way to further reduce variance from Random Forest (add randomisation)?

1. Yes
2. No

9

Extra Tree (Extremely Randomized Tree)



An ensemble learning method based on decision trees but randomises
1) certain decisions and 2) subsets of data to **minimise over-learning**
from the data **and overfitting**.

Decision Tree (High Variance)

- usually overfits the data - it learns from only one pathway of decisions.

Random Forest (Medium Variance)

- reduces the risk of overfitting by introducing randomness by:
 - building multiple trees ($n_{\text{estimators}}$)
 - drawing observations **with replacement** (e.g., a bootstrapping)
 - splitting nodes on the **best split** among a random subset of the features selected at every node.

Extra Tree (Low Variance)

- builds multiple trees and splits nodes using random subsets of features, but with two key differences
 - no bootstrap observations - it samples **without replacement**
 - **nodes are split on random splits.**

10

Overview



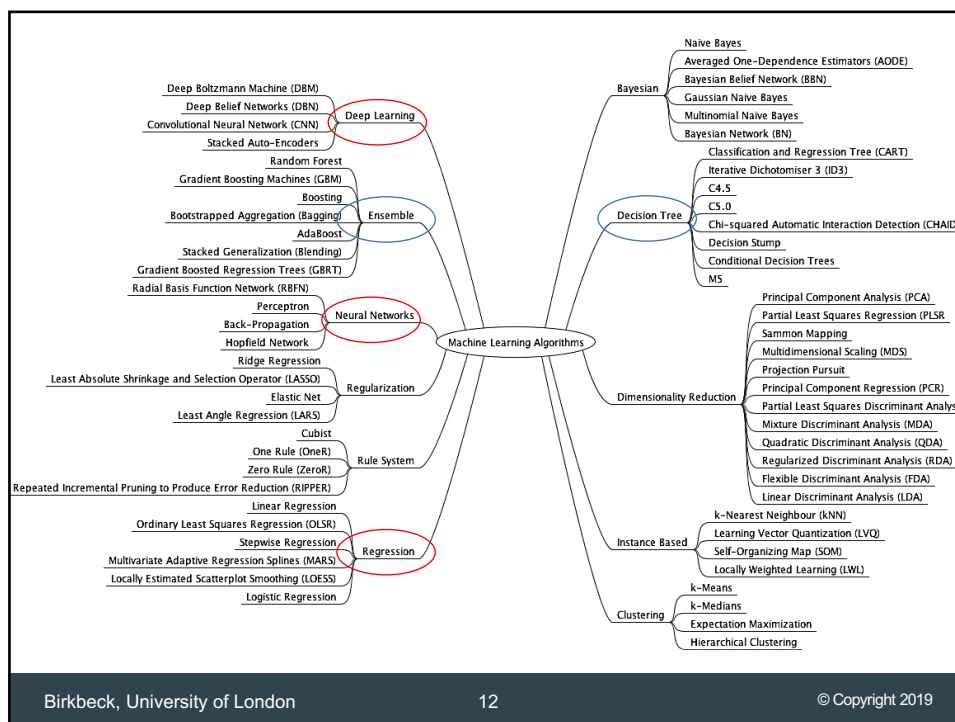
We covered:

- Decision Tree
- Information Gain
- Bagging (Boostrapping + Aggregation)
- Random Forest
- Extra Trees

We will cover:

- Linear Regression
- Logistic Regression
- Neural Network (MLP)

11



12

Model Essentials – Regressions

▶ Predict new cases.

Prediction formula

▶ Select useful inputs.

Sequential selection

▶ Optimize complexity.

Best model from sequence

// limited # of parameters vs. infinite # of parameters (e.g. nn)

Regressions, as parametric models, assume a specific **association structure** between input and target.

Trees, as predictive algorithms, simply seek to **isolate concentrations of cases** with like-valued target measurements.

13

...

13

Linear Regression Prediction Formula

$$\hat{y} = \hat{w}_0 + \hat{w}_1 \cdot x_1 + \hat{w}_2 \cdot x_2$$

prediction estimate
intercept estimate *parameter estimate* *input measurement* *prediction estimate*

// decides location

// remaining weights determine the trend strength between each input and the target.

Choose intercept and parameter estimates to **minimize**:

squared error function

$$\sum_{\text{training data}} (y_i - \hat{y}_i)^2$$

// weights are chosen to minimize the squared error between predicted and observed target values. (least square estimation)

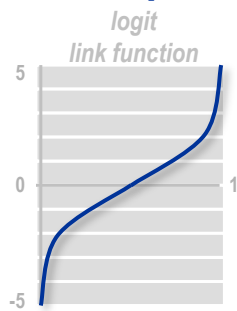
15

...

15

Logit Link Function

$$\log\left(\frac{\hat{p}}{1-\hat{p}}\right) = \hat{w}_0 + \hat{w}_1 \cdot x_1 + \hat{w}_2 \cdot x_2 \quad \text{logit scores}$$



The logit link function transforms probabilities (between 0 and 1) to logit scores (between $-\infty$ and $+\infty$).

18

...

18

Logit Link Function

$$\log\left(\frac{\hat{p}}{1-\hat{p}}\right) = \hat{w}_0 + \hat{w}_1 \cdot x_1 + \hat{w}_2 \cdot x_2 = \text{logit}(\hat{p})$$

$$\hat{p} = \frac{1}{1 + e^{-\text{logit}(\hat{p})}} \quad \text{// the logistic function is simply the inverse of the logit function.}$$

To obtain prediction estimates, the logit equation is solved for \hat{p} .

20

...

20

The Sigmoid



Let's find the inverse of the logit function:

Starting with:

$$y = \log\left(\frac{x}{1-x}\right)$$

and swapping y and x and solving for y

$$x = \log\left(\frac{y}{1-y}\right)$$

$$e^x = \frac{y}{1-y}$$

$$y = (1-y) * e^x$$

$$y = e^x - y * e^x$$

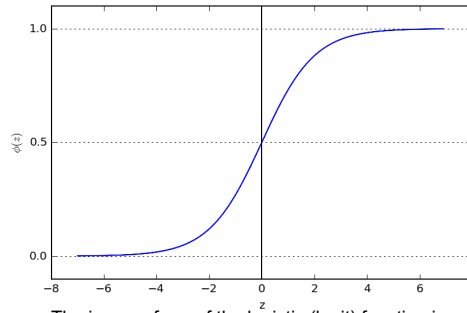
$$y + ye^x = e^x$$

$$y * (1 + e^x) = e^x$$

$$y = \frac{e^x}{1+e^x}$$

$$y = \frac{1}{\frac{1}{e^x} + 1}$$

$$y = \frac{1}{1+e^{-x}}$$



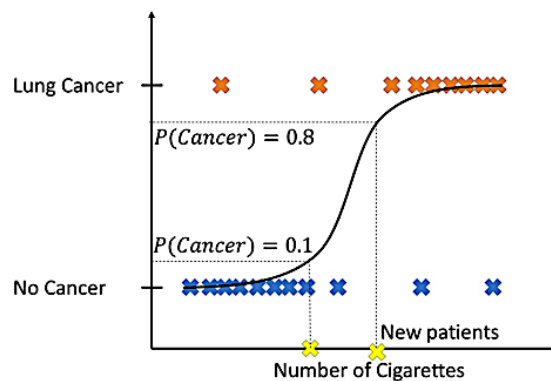
The inverse form of the logistic (logit) function is looks kind of like an S, which, I've read, is why it's called a Sigmoid function.

23

Sigmoid Function



While linear regression fits a line into the training data, logistic regression fits an S-shaped curve, called "the sigmoid function".

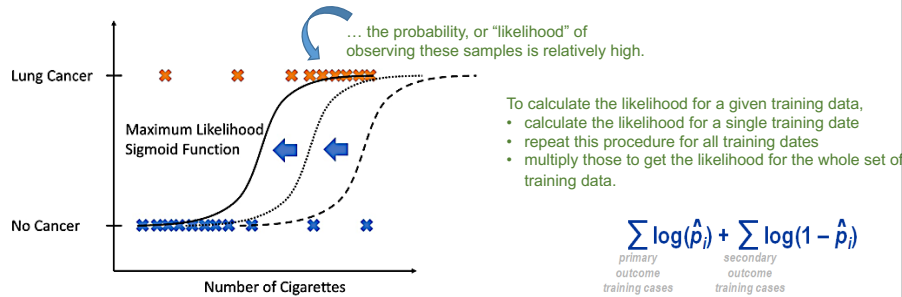


24

How to select the correct sigmoid function that best fits the training data?



- Maximum likelihood!
- Which sigmoid function would generate the observed training data with the highest probability?



25

Simple Prediction Illustration – Regressions

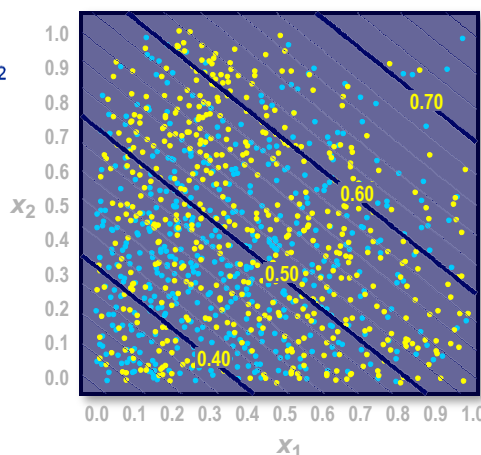
$$\text{logit}(\hat{p}) = \hat{w}_0 + \hat{w}_1 x_1 + \hat{w}_2 x_2$$

$$\hat{p} = \frac{1}{1 + e^{-\text{logit}(\hat{p})}}$$

Find parameter estimates by maximizing

$$\sum_{\text{primary outcome training cases}} \log(\hat{p}_i) + \sum_{\text{secondary outcome training cases}} \log(1 - \hat{p}_i)$$

log-likelihood function



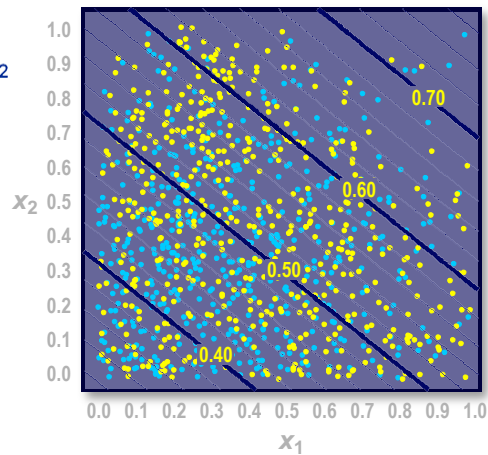
28

Simple Prediction Illustration – Regressions

$$\text{logit}(\hat{p}) = -0.81 + 0.92x_1 + 1.11x_2$$

$$\hat{p} = \frac{1}{1 + e^{-\text{logit}(\hat{p})}}$$

Using the maximum likelihood estimates, the prediction formula assigns a logit score to each x_1 and x_2 .



30

30

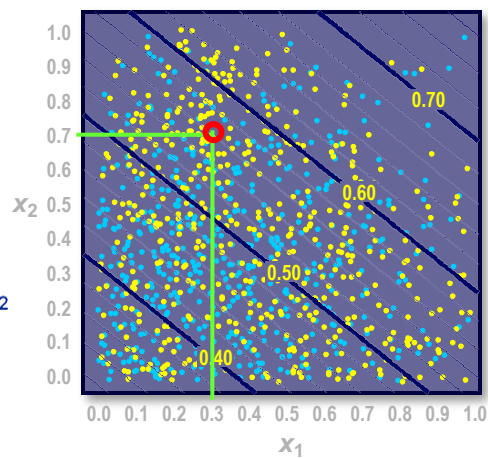
Quiz – Correct Answer

What is the logistic regression prediction for the indicated point?

- a. 0.243 // logit score - ranking
- b. 0.56 // prediction estimate from logistic function - estimate
- c. yellow // classification
- d. It depends.**

$$\text{logit}(\hat{p}) = -0.81 + 0.92x_1 + 1.11x_2$$

$$\hat{p} = \frac{1}{1 + e^{-\text{logit}(\hat{p})}}$$



32

32

Regressions: Beyond the Prediction Formula

- ▶ Manage missing values.
- ▶ Interpret the model.
- ▶ Handle extreme or unusual values.
- ▶ Use nonnumeric inputs.
- ▶ Account for nonlinearities.

33

...

33

Quiz – Correct Answer

The three sequential selection methods for building regression models can never lead to the same model for the same set of data.

- True
- False

35

35

Model Essentials – Regressions

▶ Predict new cases.

Prediction formula

▶ Select useful inputs.

Sequential selection

▶ **Optimize complexity.**

Best model from sequence

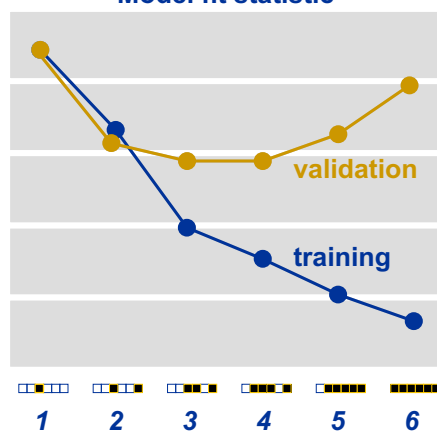
36

...

36

Model Fit versus Complexity

Model fit statistic



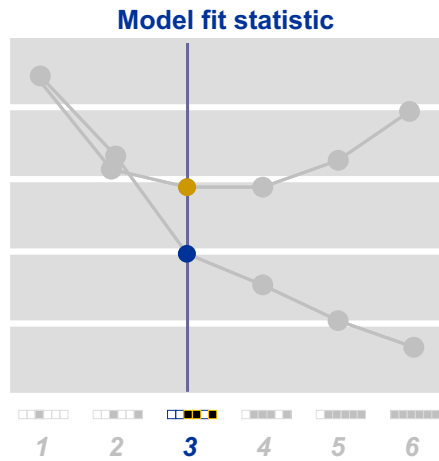
Evaluate each sequence step.

37

...

37

Select Model with Optimal Validation Fit



Evaluate each sequence step.

Choose simplest optimal model.

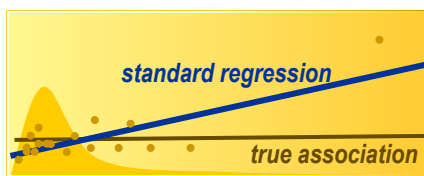
38

...

38

Extreme Distributions and Regressions

Original Input Scale



skewed input distribution

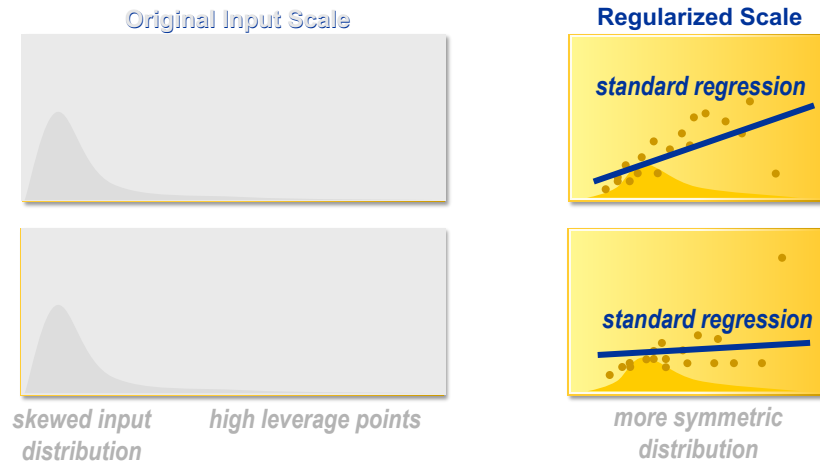
high leverage points

39

...

39

Regularizing Input Transformations



41

41

Quiz – Correct Answer

Which statement below is true about transformations of input variables in a regression analysis?

- a. They are never a good idea.
- b. They help model assumptions match the assumptions of maximum likelihood estimation.
- c. They are performed to reduce the bias in model predictions.
- d. They typically are done on nominal (categorical) inputs.

45

45

Coding Redundancy

Level	D_A	D_B	D_C	D_D	D_E	D_F	D_G	D_H	D_I
A	1	0	0	0	0	0	0	0	0
B	0	1	0	0	0	0	0	0	0
C	0	0	1	0	0	0	0	0	0
D	0	0	0	1	0	0	0	0	0
E	0	0	0	0	1	0	0	0	0
F	0	0	0	0	0	1	0	0	0
G	0	0	0	0	0	0	1	0	0
H	0	0	0	0	0	0	0	1	0
I	0	0	0	0	0	0	0	0	1

// the total # of variables is required is one less than # of inputs.

// a single categorical input can vastly increase a model's complexity – overfitting.

47

...

47

Coding Consolidation

Level	D_A	D_B	D_C	D_D	D_E	D_F	D_G	D_H	D_I
A	1	0	0	0	0	0	0	0	0
B	0	1	0	0	0	0	0	0	0
C	0	0	1	0	0	0	0	0	0
D	0	0	0	1	0	0	0	0	0
E	0	0	0	0	1	0	0	0	0
F	0	0	0	0	0	1	0	0	0
G	0	0	0	0	0	0	1	0	0
H	0	0	0	0	0	0	0	1	0
I	0	0	0	0	0	0	0	0	1

// use domain knowledge to reduce # of levels of the categorical input.

48

...

48

Coding Consolidation

<i>Level</i>	<i>D_{ABCD}</i>	<i>D_B</i>	<i>D_C</i>	<i>D_D</i>	<i>D_{EF}</i>	<i>D_F</i>	<i>D_{GH}</i>	<i>D_H</i>	<i>D_I</i>
A	1	0	0	0	0	0	0	0	0
B	1	1	0	0	0	0	0	0	0
C	1	0	1	0	0	0	0	0	0
D	1	0	0	1	0	0	0	0	0
E	0	0	0	0	1	0	0	0	0
F	0	0	0	0	1	1	0	0	0
G	0	0	0	0	0	0	1	0	0
H	0	0	0	0	0	0	1	1	0
I	0	0	0	0	0	0	0	0	1

49

49

Model Essentials – Neural Networks (MLP)

- ▶ Predict new cases.
- ▶ Select useful inputs.
- ▶ Optimize complexity.

Prediction
formula

None

Stopped
training

50

50

Model Essentials – Neural Networks

- | | |
|---|---|
| <ul style="list-style-type: none"> ▶ Predict new cases. ▶ Select useful inputs. ▶ Optimize complexity. | <div style="border: 1px solid gray; background-color: yellow; padding: 5px; display: inline-block; margin-bottom: 10px;">Prediction formula</div> <p>None</p> <p>Stopped training</p> |
|---|---|

52

...

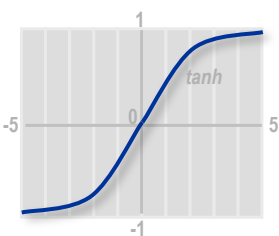
52

Neural Network Prediction Formula

▶ *prediction estimate* $\hat{y} = \hat{w}_{00} + \hat{w}_{01} \cdot H_1 + \hat{w}_{02} \cdot H_2 + \hat{w}_{03} \cdot H_3$ *hidden unit*

bias estimate *weight estimate*

// flexible addition to regression. *// the hidden unit can be thought of as regressions on the original inputs.*



$H_1 = \tanh(\hat{w}_{10} + \hat{w}_{11} x_1 + \hat{w}_{12} x_2)$
 $H_2 = \tanh(\hat{w}_{20} + \hat{w}_{21} x_1 + \hat{w}_{22} x_2)$
 $H_3 = \tanh(\hat{w}_{30} + \hat{w}_{31} x_1 + \hat{w}_{32} x_2)$

activation function *// flexibility comes at a price.*

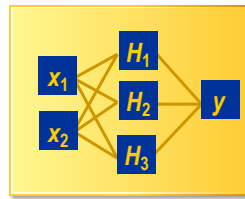
53

...

53

Neural Network Diagram

$$\log\left(\frac{\hat{p}}{1-\hat{p}}\right) = \hat{w}_{00} + \hat{w}_{01} H_1 + \hat{w}_{02} H_2 + \hat{w}_{03} H_3$$



input layer hidden layer target layer

$$H_1 = \tanh(\hat{w}_{10} + \hat{w}_{11} x_1 + \hat{w}_{12} x_2)$$

$$H_2 = \tanh(\hat{w}_{20} + \hat{w}_{21} x_1 + \hat{w}_{22} x_2)$$

$$H_3 = \tanh(\hat{w}_{30} + \hat{w}_{31} x_1 + \hat{w}_{32} x_2)$$

56

...

56

Prediction Illustration – Neural Networks

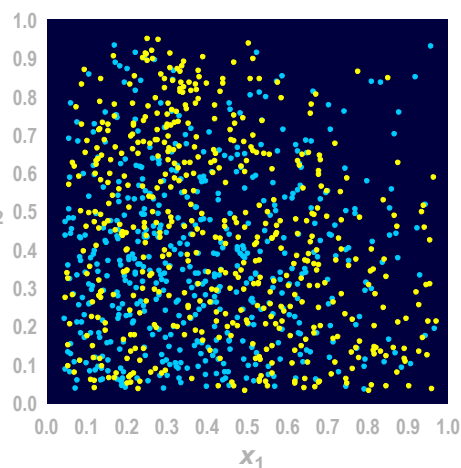
logit equation

$$\text{logit}(\hat{p}) = \hat{w}_{00} + \hat{w}_{01} H_1 + \hat{w}_{02} H_2 + \hat{w}_{03} H_3$$

$$H_1 = \tanh(\hat{w}_{10} + \hat{w}_{11} x_1 + \hat{w}_{12} x_2)$$

$$H_2 = \tanh(\hat{w}_{20} + \hat{w}_{21} x_1 + \hat{w}_{22} x_2)$$

$$H_3 = \tanh(\hat{w}_{30} + \hat{w}_{31} x_1 + \hat{w}_{32} x_2) x_2$$

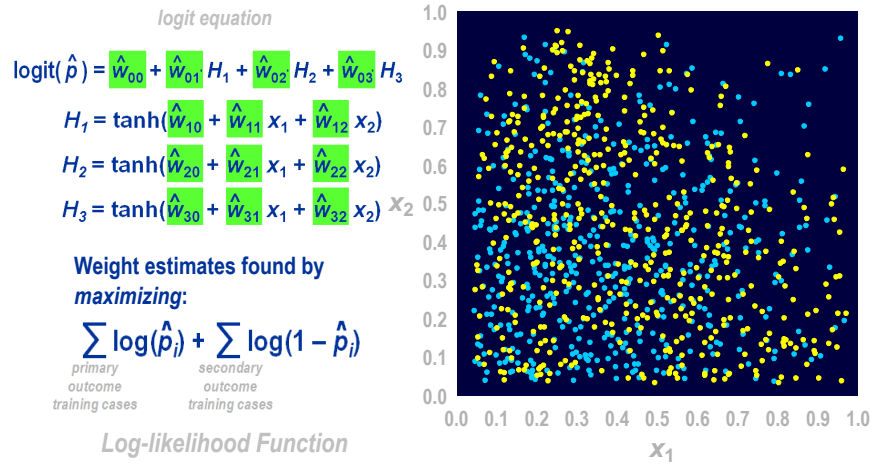


58

...

58

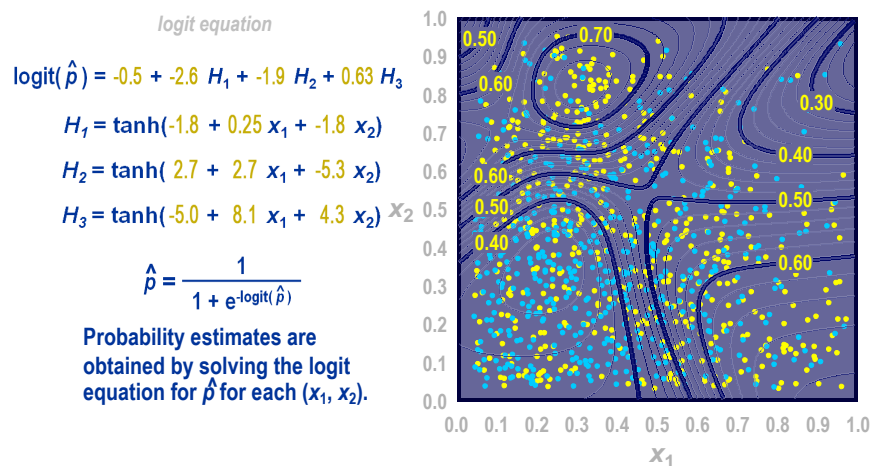
Prediction Illustration – Neural Networks



60

60

Prediction Illustration – Neural Networks



61

61

Activation Function



keeps values forward to subsequent layers **within an acceptable and useful range**, and forwards the output.

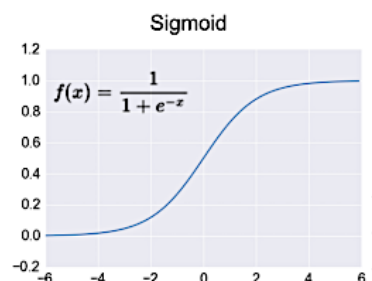
// allows
backpropagate the
model's error.

- To be useful, must be **nonlinear** and continuously **differentiable**.
- Nonlinearity allows the neural network to be a universal approximator.
- The efficient **back propagation of errors** throughout the network.
- Inside the neuron:
 - an activation function is assigned to the neuron or entire layer of neurons
 - weighted sum of input values are added up
 - the activation function is applied to weighted sum of input values and transformation takes place
 - the output to the next layer consists of this transformed value

Sigmoid



- The sigmoid function "squashes" values to the **range 0 and 1**
- The sigmoid function does not center output around **zero**.
- The sigmoid is rarely used any longer, except in the particular case of binary classification, and then only in the output layer. **Why?**



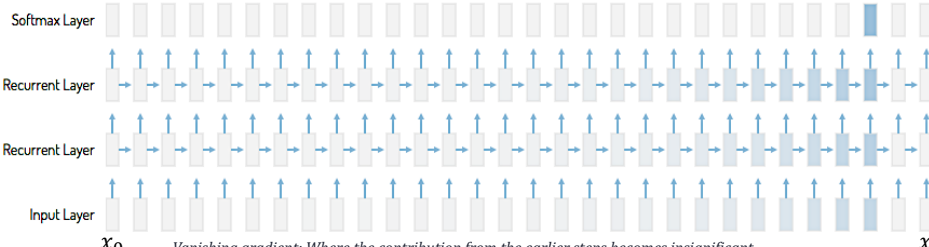
Sigmoid (RNN)

$$x_n = W_n x_0$$

$$W_n x_0 \rightarrow \begin{cases} \infty; & W > 1 \\ 0; & W < 1 \end{cases}$$

$x_i, W \in \mathbb{R}$
 $i \in [0, n]$

$x_i \in \mathbb{R}^D$
 $W \in \mathbb{R}^{D \times D}$



x_0 Vanishing gradient: Where the contribution from the earlier steps becomes insignificant. x_n

- This arrow means that *long-term information* has to sequentially travel through all cells before getting to the current processing cell.
- This means it can be easily corrupted by being multiplied many times by small numbers < 0 . This is the cause of vanishing gradients.

Birkbeck, University of London
64
© Copyright 2019

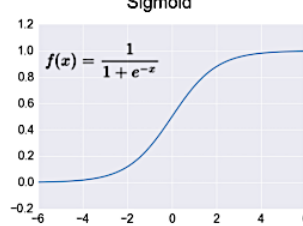
64

Hyperbolic Tangent (TanH)

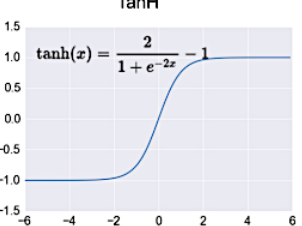
- outputs values between **-1.0** and **1.0** - centered around **zero**.
- can be thought of as a scaled, or shifted sigmoid.
- the tanh function was preferred over the sigmoid
 - easier to train and often had better performance.

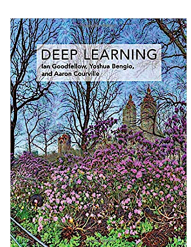
"... the hyperbolic tangent activation function typically performs better than the logistic sigmoid."

Sigmoid



TanH





Ian Goodfellow
and Yoshua Benjio

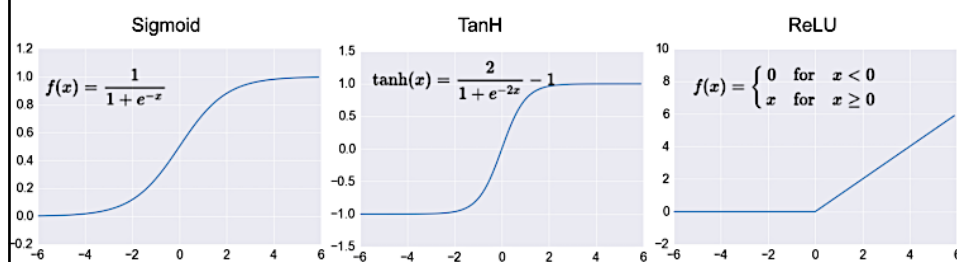
Birkbeck, University of London
65
© Copyright 2019

65

Rectified Linear Unit (ReLU)



- The range of ReLU is $[0, \text{inf})$.
- It gives an output x if x is positive and 0 otherwise.
 - almost 50% of the network yields 0 activation
 - a fewer neurons are firing and the network is lighter.
- ReLU is nonlinear in nature - any function can be approximated with combinations of ReLU.
- Some neurons stop responding to variations in error/input (simply because gradient is 0, nothing changes) – called **dying ReLU**
- There are variations in ReLU to mitigate this issue by simply making the horizontal line into **non-horizontal component**.
- Adoption of ReLU may easily be considered one of the few milestones in the deep learning revolution



66

Neural Nets: Beyond the Prediction Formula

► Manage missing values.

► Handle extreme or unusual values.

// activation functions squash extreme input values between 0/-1 and 1.

► Use non-numeric inputs.

► Account for nonlinearities.

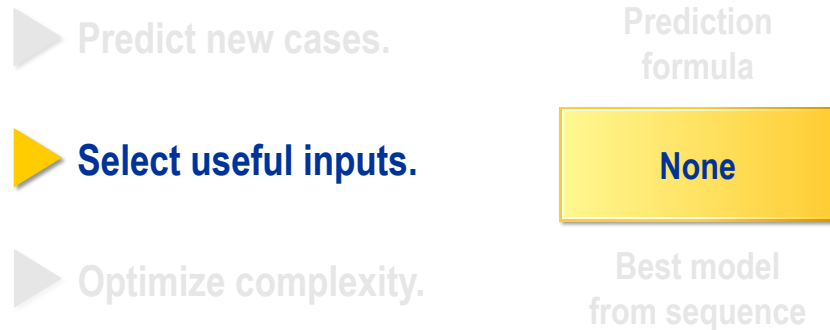
► Interpret the model.

67

...

67

Model Essentials – Neural Networks



68

68

Quiz – Correct Answers

Which of the following are true about neural networks?

- a. Neural networks are universal approximators.
- b. Neural networks have no internal, automated process for selecting useful inputs.
- c. Neural networks are easy to interpret and thus are very useful in highly regulated industries.
- d. Neural networks cannot model nonlinear relationships.

70

70

Model Essentials – Neural Networks

▶ Predict new cases.

Prediction
formula

▶ Select useful inputs.

Sequential
selection

▶ **Optimize complexity.**

Stopped training

71

...

71

Fit Statistic versus Optimization Iteration

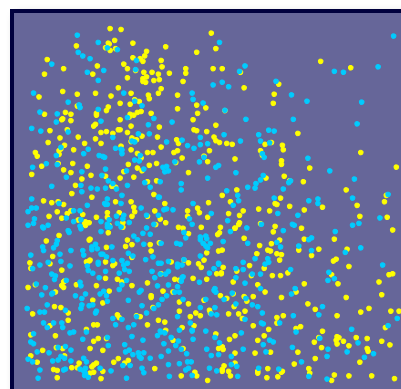
$$\text{logit}(\hat{p}) = 0 + 0 \cdot H_1 + 0 \cdot H_2 + 0 \cdot H_3$$

$$H_1 = \tanh(-1.5 - .03x_1 - .07x_2)$$

$$H_2 = \tanh(.79 - .17x_1 - .10x_2)$$

$$H_3 = \tanh(.57 + .05x_1 + .35x_2)$$

random initial
input weights and biases

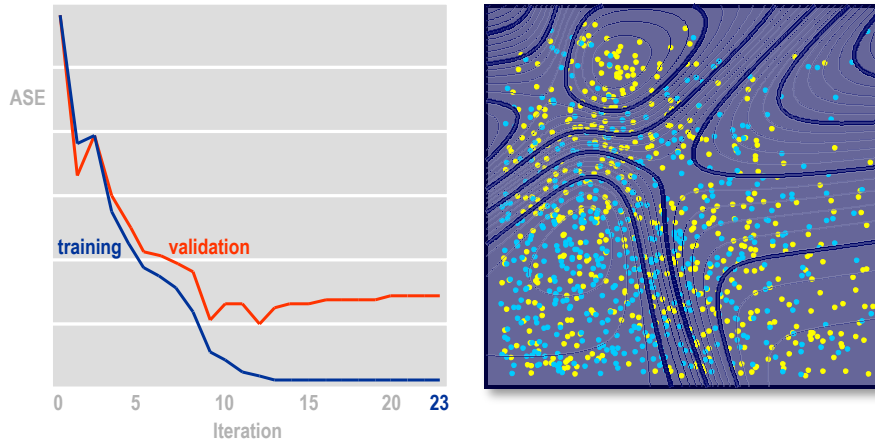


74

...

74

Fit Statistic versus Optimization Iteration

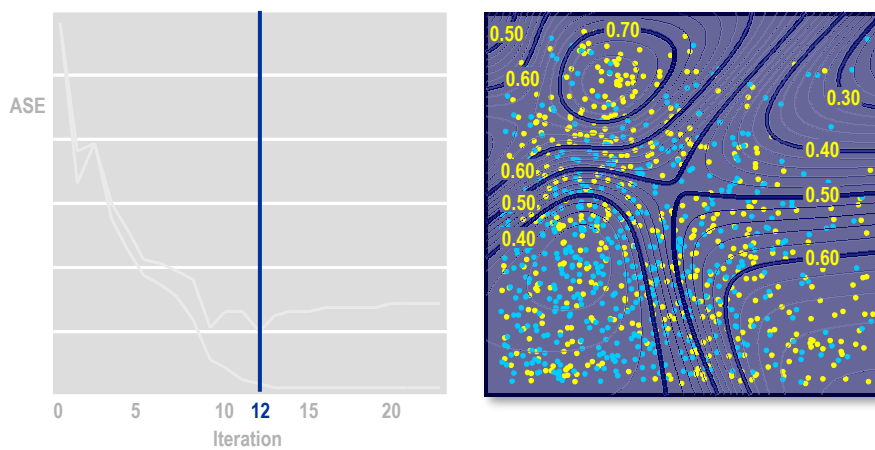


98

...

98

Fit Statistic versus Optimization Iteration



99

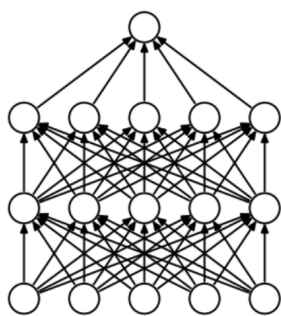
...

99

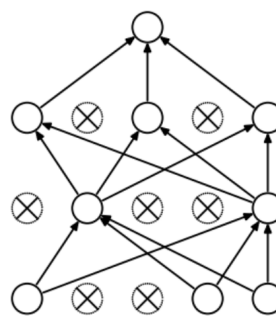
Dropout – network regularization



Dropout works by randomly setting the outgoing edges of hidden units to 0 at each update of the training phase.



(a) Standard Neural Net



(b) After applying dropout.

// during the forward pass of your network
randomly set some of the neurons to zero.

Dropout cont.



There are some debate as to whether the dropout should be placed before or after the activation function.

- A rule of thumb – place the dropout **after** the activate function for all activation functions other than **relu**.
- The dropout rate is set to 20%, meaning one in 20% of number of inputs will be randomly excluded (set to 0) from each update cycle.
- For intermediate layers, **0.5** for large networks works well. For the input layer, it should be kept about **0.2** or lower.

```
# dropout in the input layer with weight constraint
def create_model():
    # create model
    model = Sequential()
    model.add(Dropout(0.2, input_shape=(60,)))
    model.add(Dense(60, activation='relu')) // each neuron receives input from all the neurons
    model.add(Dense(30, activation='relu')) // in the previous layer, thus densely connected.
    model.add(Dense(1, activation='sigmoid'))

    # compile model
    sgd = SGD(lr=0.1, momentum=0.9)
    model.compile(loss='binary_crossentropy', optimizer=sgd, metrics=['accuracy'])
    return model
```

Dropout: A Simple Way to Prevent Neural Networks from Overfitting

Nitish Srivastava
 Geoffrey Hinton
 Alex Krizhevsky
 Ilya Sutskever
 Ruslan Salakhutdinov
*Department of Computer Science
 University of Toronto
 10 Kings College Road, Rm 3302
 Toronto, Ontario, M5S 3G4, Canada.*

NITISH@CS.TORONTO.EDU
 HINTON@CS.TORONTO.EDU
 KRIZ@CS.TORONTO.EDU
 ILYA@CS.TORONTO.EDU
 RSALAKHU@CS.TORONTO.EDU

Editor: Yoshua Bengio

Abstract

Deep neural nets with a large number of parameters are very powerful machine learning systems. However, overfitting is a serious problem in such networks. Large networks are also slow to use, making it difficult to deal with overfitting by combining the predictions of many different large neural nets at test time. Dropout is a technique for addressing this problem. The key idea is to randomly drop units (along with their connections) from the neural network during training. This prevents units from co-adapting too much. During training, dropout samples from an exponential number of different "thinned" networks. At test time, it is easy to approximate the effect of averaging the predictions of all these thinned networks by simply using a single unthinned network that has smaller weights. This significantly reduces overfitting and gives major improvements over other regularization methods. We show that dropout improves the performance of neural networks on supervised learning tasks in vision, speech recognition, document classification and computational biology, obtaining state-of-the-art results on many benchmark data sets.

Keywords: neural networks, regularization, model combination, deep learning

102

Cost Function



A loss function maps decisions to their associated costs.

- In supervised learning, we want to **minimise the error** for each training example during the learning process.
- This is done using some optimisation strategies like **gradient descent**. And this error comes from the loss function.
- Cost function Vs Loss function?
 - **cost function** and **loss function** are synonymous and used interchangeably but
 - **A loss function is for a single training example** (also called an **error function**).
 - A cost function, is **the average loss** over the entire training dataset.
- The optimisation strategies aim at minimising the cost function (average loss).

103

Common Loss Functions



Six common loss functions used in machine learning

Regression Loss Functions

- Squared Error Loss (a.k.a L2 Loss)

$$L = \left(\overset{\text{// actual}}{y} - \underset{\text{// predicted}}{f(x)} \right)^2$$

// the corresponding cost function is the **Mean** of these **Squared Errors (MSE)**.

- Absolute Error Loss (a.k.a L1 Loss)

$$L = |y - f(x)|$$

// is the **distance** between the predicted and the actual values, irrespective of the sign. // the cost is the **Mean** of these **Absolute Errors (MAE)**.

- Huber Loss

$$L_{\delta} = \begin{cases} \frac{1}{2}(y - f(x))^2, & \text{if } |y - f(x)| \leq \delta \\ \delta|y - f(x)| - \frac{1}{2}\delta^2, & \text{otherwise} \end{cases}$$

Quadratic // combines the best properties of MSE and MAE.
Linear // quadratic for smaller errors and is linear otherwise (identified by its *delta* parameter)

Common Loss Functions



Binary Classification Loss Functions

- Binary Cross Entropy
- Hinge Loss
 - Hinge loss is primarily used with SVM Classifiers with class labels -1 and 1.
 - So make sure you change the label of the 'positive' target class in the dataset from 0 to -1.
 - Hinge loss for an input-output pair (x, y) is given as:

$$L = \max(0, 1 - y * f(x))$$

```
def Hinge(yHat, y):
    return np.max(0, 1 - yHat * y)
```

Common Loss Functions



Multi-class Classification Loss Functions

- Multi-class Cross Entropy Loss
- Kullback Leibler (KL) Divergence Loss (a.k.a Relative Entropy)
 - measures how a probability distribution differs from another distribution.
 - A KL-divergence of **zero** indicates that the distributions are **identical**.
 - Can be seen as combination of **Cross-Entropy and Entropy**.
 - KL-Divergence is used more commonly to approximate complex functions than in multi-class classification.

$$D_{KL}(P||Q) = \sum_i P(i) \log \frac{P(i)}{Q(i)}$$

```
def KLDivergence(yHat, y):
    """
    :param yHat:
    :param y:
    :return: KLDiv(yHat || y)
    """
    return np.sum(yHat * np.log((yHat / y)))
```

106

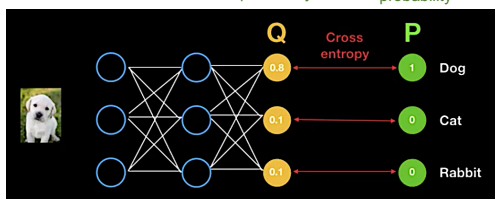
Cross Entropy (Logloss) – NN's default cost function



CE is used to quantify the difference between two probability distributions.

To calculate the probability (binary classification), use the sigmoid function $S(z) = \frac{1}{1 + e^{-z}}$

```
// softmax layer
// a way to squash Q into probability
// estimated probability // true probability
```



In binary classification, where $M=2$, cross-entropy can be calculated as:

$$-\sum_{n=1}^N p_n \cdot \log(q_n) + (1 - p_n) \cdot \log(1 - q_n)$$

If $M>2$, we calculate a separate loss for each class label per observation and sum the result.

$$H(p, q) = -\sum_{c=1}^M \log(q_c) \cdot p_c$$

// amount of info // probability distribution

- $H = -(\ln(0.0) * 1.0 + \ln(1.0) * 0.0 + \ln(0.0) * 0.0) = \text{infinity}$ // totally wrong prediction
- $H = -(\ln(0.8) * 1.0 + \ln(0.1) * 0.0 + \ln(0.1) * 0.0) = 0.321$ // similar prediction
- $H = -(\ln(1.0) * 1.0 + \ln(0.0) * 0.0 + \ln(0.0) * 0.0) = 0.0$ // perfect prediction (Q=P)

107

Kullback-Leibler (KL) Divergence Loss

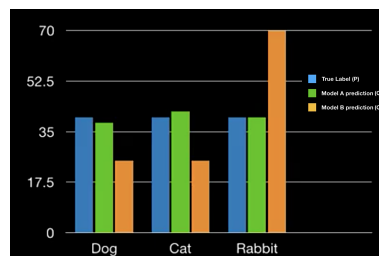


Relative Entropy

Which model has better prediction?

If prediction = true label, no difference.

- Smaller difference between P and Q_A .
- Larger difference between P and Q_B .
- Could we 'measure' the difference?



KL divergence can be denoted as $D_{KL}(P \parallel Q)$

- $D_{KL}(P \parallel P) = 0.0$
- $D_{KL}(P \parallel Q_A) = 0.25$
- $D_{KL}(P \parallel Q_B) = 1.85$

// additional amount of information required to specify the value of x as a result of using $q(x)$ instead of true distribution of $p(x)$.

$$\begin{aligned}
 D_{KL}(p|q) &= \sum_i p_i \log \frac{p_i}{q_i} \\
 &= \sum_i (-p_i \log q_i + p_i \log p_i) \\
 &= -\sum_i p_i \log q_i + \sum_i p_i \log p_i \\
 &= -\sum_i p_i \log q_i - \sum_i p_i \log \frac{1}{p_i} \\
 &= -\sum_i p_i \log q_i - H(p) \\
 &\quad \text{// CE} \\
 &= \sum_i p_i \log \frac{1}{q_i} - H(p)
 \end{aligned}$$

Cross Entropy Vs KL Divergence



$D_{KL}(P \parallel Q)$ shows how much Q is relatively differ from P .

// measure for uncertainty (=impurity)

- $D(P \parallel Q) = \text{Cross Entropy}(P, Q) - \text{Entropy}(P)$

// this does not change as the parameters of the model change so can be ignored in the loss function.

$$-\sum_{n=1}^N p_n \cdot \log(q_n) - \left(-\sum_{n=1}^N p_n \cdot \log(p_n) \right)$$

- $D(P \parallel Q) \geq 0$ // additional amount of information required to specify the value of x as a result of using $q(x)$ instead of true distribution of $p(x)$.

- $D(P \parallel Q) \neq D(Q \parallel P)$

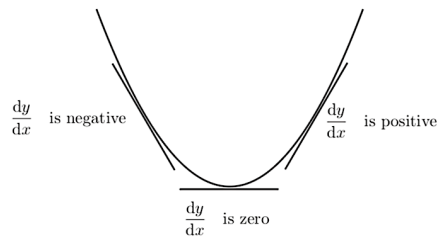
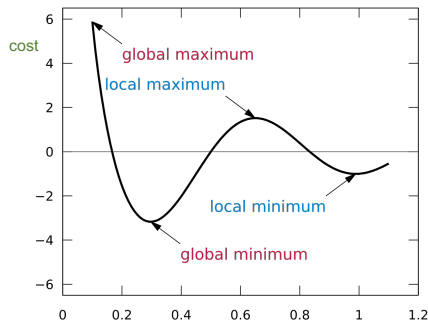
Gradient Descent



Minimising the difference between expected value and model output is done using gradient descent (a. k. a. backpropagation)

// rate of change w.r.t x

Because **the derivative** provides information about the **gradient or slope** of the graph of a function we can use it to **locate points on a graph where the gradient is zero** (e.g. if $x = 1$ then $f'(x)$ becomes zero).



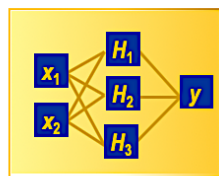
$\frac{dy}{dx}$ goes from negative through zero to positive as x increases.

111

Stochastic Gradient Descent



$$\log\left(\frac{\hat{p}}{1-\hat{p}}\right) = \hat{w}_{00} + \hat{w}_{01} \cdot H_1 + \hat{w}_{02} \cdot H_2 + \hat{w}_{03} \cdot H_3$$



input layer hidden layer target layer

$$H_1 = \tanh(\hat{w}_{10} + \hat{w}_{11} x_1 + \hat{w}_{12} x_2)$$

$$H_2 = \tanh(\hat{w}_{20} + \hat{w}_{21} x_1 + \hat{w}_{22} x_2)$$

$$H_3 = \tanh(\hat{w}_{30} + \hat{w}_{31} x_1 + \hat{w}_{32} x_2)$$

112

SGD

Too low

A small learning rate requires many updates before reaching the minimum point

Just right

The optimal learning rate swiftly reaches the minimum point

Too high

Too large of a learning rate causes drastic updates which lead to divergent behaviors

min

gradient = -2.9999999999999996

- 1) calculate gradients of the loss function.
- 2) updating existing parameters in response to the gradients.

$$x_0 = x_0 - \eta \frac{\partial f}{\partial x_0}$$

Birkbeck, University of London

113

© Copyright 2019

113

SGD

$$\log\left(\frac{\hat{p}}{1-\hat{p}}\right) = \hat{w}_{00} + \hat{w}_{01} H_1 + \hat{w}_{02} H_2 + \hat{w}_{03} H_3$$

$H_1 = \tanh(\hat{w}_{10} + \hat{w}_{11} x_1 + \hat{w}_{12} x_2)$
 $H_2 = \tanh(\hat{w}_{20} + \hat{w}_{21} x_1 + \hat{w}_{22} x_2)$
 $H_3 = \tanh(\hat{w}_{30} + \hat{w}_{31} x_1 + \hat{w}_{32} x_2)$

$$w_0 = w_0 - \eta \frac{\partial f}{\partial w_0}$$

$$w_1 = w_1 - \eta \frac{\partial f}{\partial w_1}$$

$$w_2 = w_2 - \eta \frac{\partial f}{\partial w_2}$$

...

$$w_n = w_n - \eta \frac{\partial f}{\partial w_n}$$

Birkbeck, University of London

114

© Copyright 2019

114

Questions?

paul@dcs.bbk.ac.uk