Imperial College London

April 30th, 2021

# Regularization, Bias, Variance, etc…

Olivier Dubrule

1

Imperial College London

## Objectives of the Day

- Underfitting and Overfitting in Relation to Bias and Variance
- Regularization to Control Variance/Overfitting
- Batch Normalization for Acceleration of Training and Regularization
- Machine Learning Diagnostics for Hyperparameters Optimization
- Training set, Validation Set and Test Set for Machine Learning Diagnostics
- k-Fold Validation as a Machine Learning Diagnostics
- Approaches for Hyperparameter Search

2

Imperial College London

## Regularization, Bias and Variance

1. Overfitting and Underfitting, Bias and Variance
2. Regularization
3. Batch Normalization
4. Regression and Classification Criteria
5. Training Set, Validation Set and Test Set
6. K-Fold Validation

3

Imperial College London

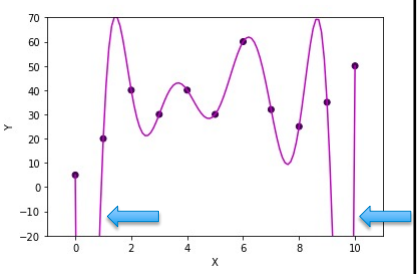
## The Training Set Error is not an indicator of how well the fitted model is going to work on other data

If we have a large number of fitting parameters, the trained model may fit the training set very well:

$$J(\theta) = \frac{1}{2m}\sum_{i=1}^{m}\left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)^2 = 0$$

But fail to generalize to new data! This is called *Overfitting.*

Example of Polynomial Overfitting

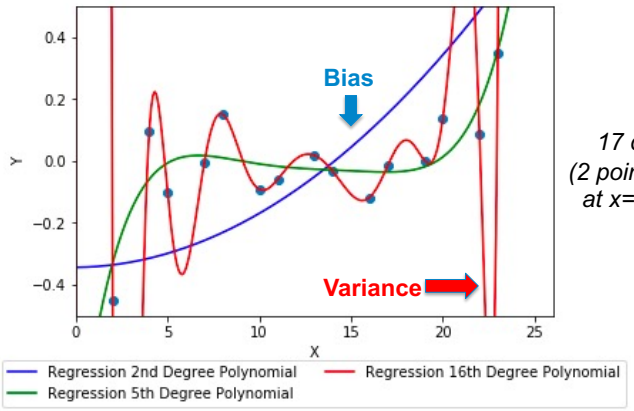



$m = 11$ data points
$10^{th}$ degree polynomial with bias

4

Imperial College London

## Bias (or Underfitting) vs Variance (or Overfitting)




*17 data points (2 points not shown at x=0 and x=26)*

5

Imperial College London

## Why the terms « Bias » and « Variance »?

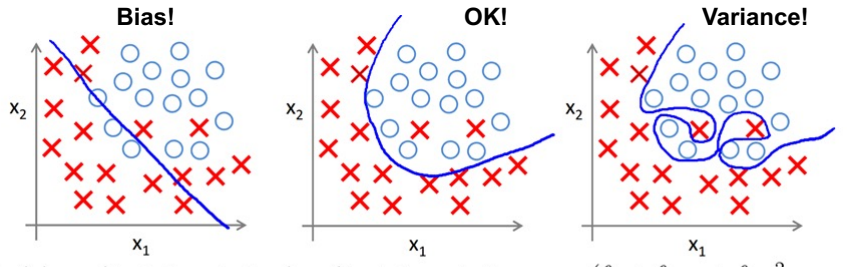
***Bias:*** *the hypothesis function $h_\theta(x)$ has a « pre-conception » or an « a priori » idea of the data variations which is too simple, which is biased from the start (for instance it is a polynomial of too low degree), considering the actual variability of the data.*

***Variance:*** *the hypothesis function $h_\theta(x)$ has too many degrees of freedom - or parameters - considering the size of the Training Set (for example $h_\theta(x)$ is a polynomial of too high degree). As a result, an estimate for a given point may change drastically following a small change in the training data..*

6

Imperial College London

## Under/Over-fitting may also apply to Classification




$h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2)$

($g$ = sigmoid function)

$g(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1^2 + \theta_4 x_2^2 + \theta_5 x_1 x_2)$

$g(\theta_0 + \theta_1 x_1 + \theta_2 x_1^2 + \theta_3 x_1^2 x_2 + \theta_4 x_1^2 x_2^2 + \theta_5 x_1^2 x_2^3 + \theta_6 x_1^3 x_2 + \dots)$

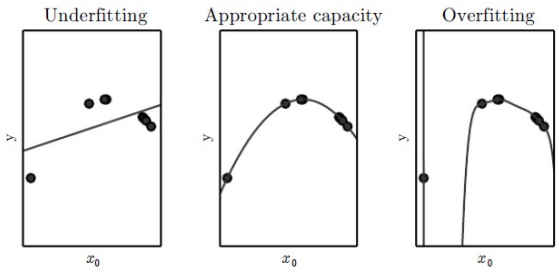
*Source: Machne Learning Course, Andrew Ng*

7

Imperial College London

## Capacity of a Model

*The **Capacity** of a model is associated with its likely underfitting or overfitting. Informally a model's capacity is its ability to fit a wide variety of functions. Models with low capacity may tend to **underfit (Bias) because they do not have enough parameters** , models with high capacity may tend to **overfit (Variance) because they have too many parameters.***




Goodfellow et al, 2017

8

Imperial College London

## The Generalization Error

*The trained model must perform well on new, previously unseen data, not just those on which the model was trained. The ability to perform well on* ***previously unseen*** *data – or* ***Test Data*** *- is called* ***Generalization.***

Goodfellow et al, 2017

9

Imperial College London

## Underfitting and Overfitting

A good Machine Learning algorithm must:

1. Make the Training Error small. If this is not the case, we have Underfitting.
2. Make the gap between Training and Test – or Generalization - Error small. If this is not the case we have Overfitting.

Goodfellow et al, 2017

10

Imperial College London

## Regularization, Bias and Variance

1. Overfitting and Underfitting, Bias and Variance
2. Regularization
3. Batch Normalization
4. Regression and Classification Criteria
5. Training Set, Validation Set and Test Set
6. K-Fold Validation

11

Imperial College London

## One Way to Avoid Overfitting: Regularization

*Regularization is any modification we make to a learning algorithm that is intended to reduce its Generalization Error but not its Training Error…*

*An effective regularizer is one that reduces Variance significantly while not increasing the Bias.*

Goodfellow et al, 2017

12

Imperial College London

## (L1 or L2) Regularization for Regression ML

L1 and L2 Regularizations consist of adding a new term to the objective function in order to control the variations of the parameters:

$$J(\theta) = \frac{1}{2m}\left(\sum_{i=1}^{m}\left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)^2 + \lambda \sum_{j=1}^{n} \theta_j^2\right)$$ if L2 norm is used

$\lambda$ Regularization Parameter, controlling the "Weight Decay" which makes the weights smaller

$$J(\theta) = \frac{1}{2m}\left(\sum_{i=1}^{m}\left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)^2 + \lambda \sum_{j=1}^{n} |\theta_j|\right)$$ if L1 norm is used

13

Imperial College London

## The well-known Ridge & Lasso Regressions

$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \ldots + \theta_n x_n$$

**Ridge Regression (preferred because math derivations simpler)**

$$J(\theta) = \frac{1}{2m}\left(\sum_{i=1}^{m}\left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)^2 + \lambda \sum_{j=1}^{n} \theta_j^2\right)$$ if L2 norm is used

**Lasso (Least Absolute Shrinkage Selector Operator) Regression**

$$J(\theta) = \frac{1}{2m}\left(\sum_{i=1}^{m}\left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)^2 + \lambda \sum_{j=1}^{n} |\theta_j|\right)$$ if L1 norm is used

*Lasso & Ridge Regressions, call $\lambda$ the shrinkage parameter because it shrinks the weights.*

14

Imperial College London

## Classification with L2 Regularization: Exercise

*Take the Case of Binary Linear Logistic Regression in 2-D.*

*Write* $h_\theta(x)$ *as a function of* $x$ *and* $\theta$. $h_\theta(x) = \sigma(\theta_0 + \theta_1 x_1 + \theta_2 x_2) = \frac{1}{1 + e^{-(\theta_0 + \theta_1 x_1 + \theta_2 x_2)}}$

*Assume there are* $m$ *data points* $(x^{(i)}, y^{(i)})$ *for* $i = 1, \dots, m$

*Express* $J(\theta)$*using L2 regularization and a regularization parameter* $\lambda$.

$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}\left[y^{(i)}\log\left(h_\theta(x^{(i)})\right) + (1 - y^{(i)})\log\left(1 - h_\theta(x^{(i)})\right)\right] + \frac{\lambda}{2m}(\theta_0^2 + \theta_1^2 + \theta_2^2)$$

*Calculate* $\frac{\partial J(\theta)}{\partial \theta_j}$ *for one of the* $\theta_j$ *parameters.* $\frac{\partial J(\theta)}{\partial \theta_2} = \frac{1}{m}\sum_{i=1}^{m}[h_\theta(x^{(i)}) - y^{(i)}]x_2^{(i)} + \frac{\lambda}{m}\theta_2$

15

Imperial College London

## Gradient Descent for Regularized Logistic Regression

Gradient Descent Approach without Regularization Term

$$\theta_j := \theta_j - \alpha\frac{\partial J}{\partial \theta_j} = \theta_j - \alpha\frac{1}{m}\sum_{i=1}^{m}\left(h_\theta(x^{(i)}) - y^{(i)}\right)x_j^{(i)}\Big)$$

Gradient Descent with Regularization Term (if L2 norm is used)

$$\theta_j := \theta_j - \alpha\frac{\partial J}{\partial \theta_j} = \theta_j - \alpha\frac{1}{m}\sum_{i=1}^{m}\left(h_\theta(x^{(i)}) - y^{(i)}\right)x_j^{(i)}\Big) - \alpha\frac{\lambda}{m}\theta_j$$

$$\theta_j := \theta_j\left(1 - \alpha\frac{\lambda}{m}\right) - \alpha\frac{1}{m}\sum_{i=1}^{m}\left(h_\theta(x^{(i)}) - y^{(i)}\right)x_j^{(i)}\Big)$$

16

Imperial College London

## Gradient Descent with Regularized Regression

Consider in more detail the Gradient Descent term in case of Regularization:

$$\theta_j := \theta_j \left(1 - \alpha \frac{\lambda}{m}\right) - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right) x_j^{(i)}$$

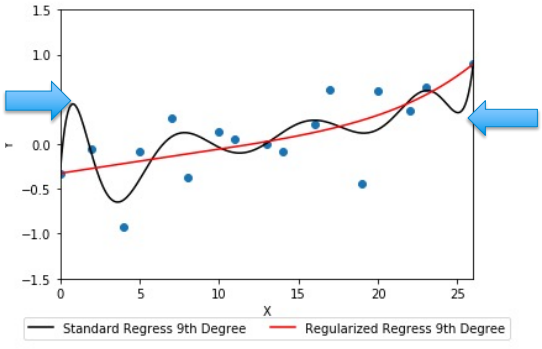
Systematic decrease of absolute value $\theta_j$ at each iteration

Same term as for optimization without regularization

17

Imperial College London

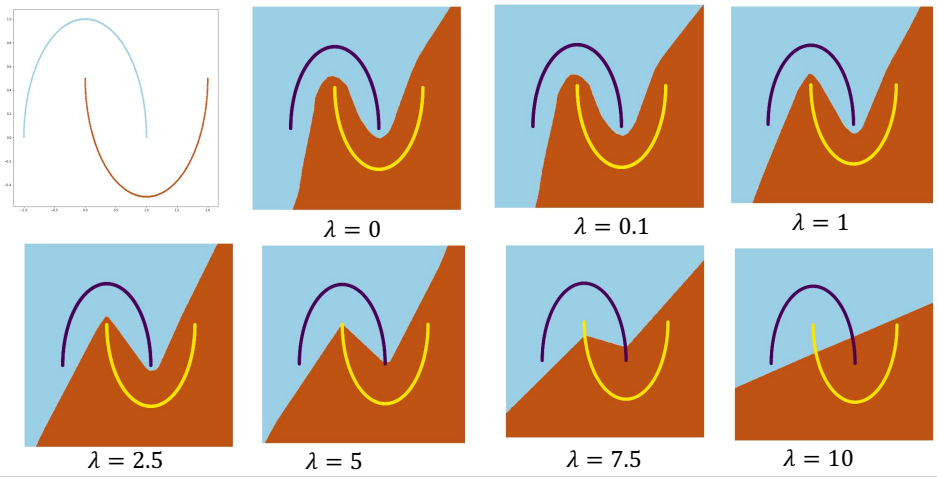
## The Impact of Applying Regularization to Regression




18

Imperial College London

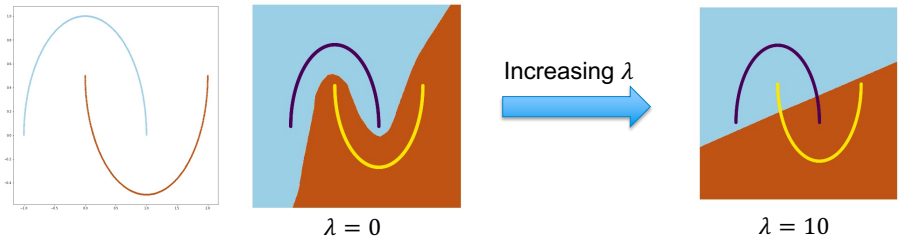
## A Simple Neural Network Regularization Example (1)




19

Imperial College London

## A Simple Neural Network Regularization Example (2)




As $\lambda$ increases, the values of the weights tend to zero, and the Decision Boundary first becomes a broken Line, then a single Line.

20

Imperial College London

## Model Averaging for Reducing Generalization Error

***Approach***:

Given a Training Set of $m$ data, train not just one but a number $k$ of different models and predict the Test Set by averaging the results of the $k$ models.

The $k$ models can consist of the same algorithm trained on different subsets of the initial Training Set (this is called Bagging), or different algorithms trained on the same Training Set (this is Model Averaging).

***Bagging*** *can consist of constructing $k$ datasets from the initial Training Set of $m$ examples, building each dataset by sampling with replacement m examples from the Training Set.*

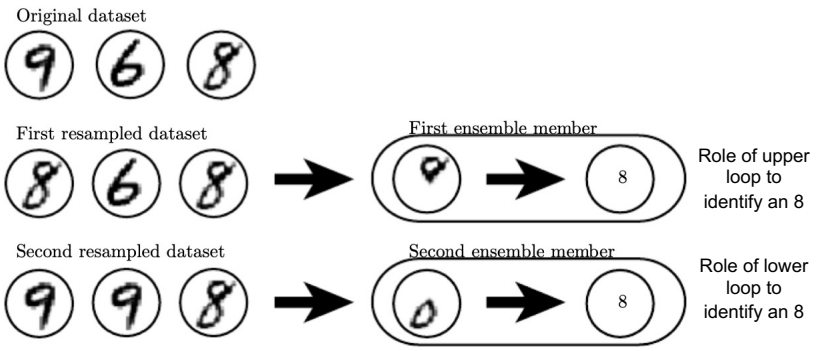
***Model Averaging Methods are also called Ensemble Methods***

https://www.youtube.com/watch?v=kAwF--GJ-ek

21

Imperial College London

## MNIST- Based Example of Sampling with Replacement



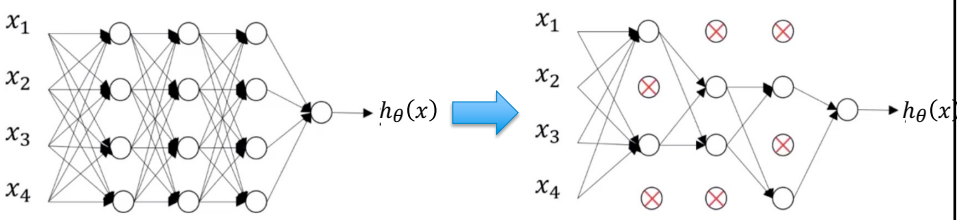

From Goodfellow et al, 2016

22

Imperial College London

## Some Sort of Model Averaging with Drop-Out (1)

***At Training time***:




For each Training example… Drop hidden layers units with 0.5 (or $p$) probability

*Same approach for input layer but with $p$ closer to 1 (say 0.8) . No change in output layer.*

***We are averaging the results over a set of network configurations!***

https://www.youtube.com/watch?v=kAwF--GJ-ek

23

Imperial College London

May 1st, 2020

## Classic Dropout Paper (>18000 citations!)

Journal of Machine Learning Research 15 (2014) 1929-1958 Submitted 11/13; Published 6/14

**Dropout: A Simple Way to Prevent Neural Networks from Overfitting**

**Nitish Srivastava** NITISH@CS.TORONTO.EDU
**Geoffrey Hinton** HINTON@CS.TORONTO.EDU
**Alex Krizhevsky** KRIZ@CS.TORONTO.EDU
**Ilya Sutskever** ILYA@CS.TORONTO.EDU
**Ruslan Salakhutdinov** RSALAKHU@CS.TORONTO.EDU
*Department of Computer Science*
*University of Toronto*
*10 Kings College Road, Rm 3302*
*Toronto, Ontario, M5S 3G4, Canada.*

**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

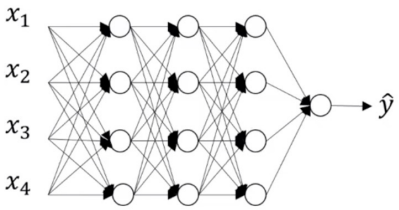
http://jmlr.csail.mit.edu/papers/volume15/srivastava14a/srivastava14a.pdf

24

Imperial College London

## Another Regularization Approach: Drop-Out (2)

***At Test time***: no drop-out, use all hidden units but multiply their output by $p = 0.5$




*Drop-out can also be used to quantify Model Uncertainty in Bayesian Neural Networks (Gal and Ghahramani, 2016)*

https://www.youtube.com/watch?v=kAwF--GJ-ek

25

Imperial College London

## Another Regularization Approach: Data Augmentation

From Andrew Ng, CL2W1L08

26

Imperial College London

## Software Tools for Data Augmentation

A number of basic geometrical image transformations are likely to change the appearance of the image, but not its class:

- Rotation (to some reasonable degree)
- Translation (up, down, left, right)
- Zooming
- Cropping
- Added noise
- Changing the Brightness level

…

These are readily available in Python code (*imgaug, Albumentation* packages…).

27

Imperial College London

## Regularization, Bias and Variance

1. Overfitting and Underfitting, Bias and Variance
2. Regularization
3. Batch Normalization
4. Regression and Classification Criteria
5. Training Set, Validation Set and Test Set
6. K-Fold Validation

28

Imperial College London

## Normalization vs Standardization Common Definitions

### Normalization

$x$

$xmin = 50$ $xmax = 500$

$x'$

0 1

$$x' = \frac{x - xmin}{xmax - xmin}$$

### Standardization

$$x' = \frac{x - \mu_x}{\sigma_x}$$

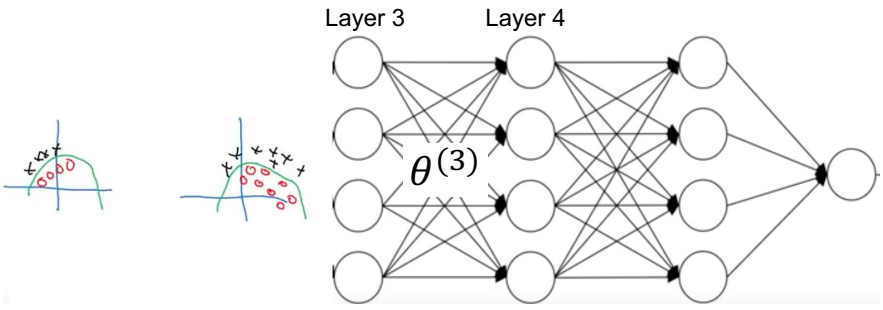
(where $\mu_x$ and $\sigma_x$ are mean and standard deviation of array $x$)

***This is the basis for "Batch Normalization"***

29

Imperial College London

## Batch Normalization: Covariate Shift



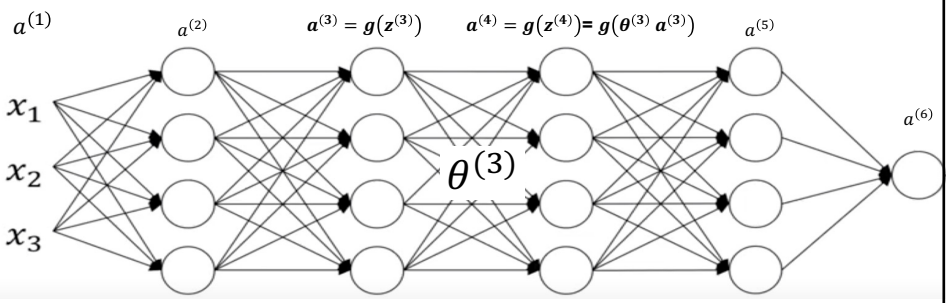

*Each time the distribution of the input of a neural network becomes different from the input used for training, the weights become invalid (this is like Training a network on Black Cats and applying it to Coloured Cats). When the input distribution of a training system changes, this is called Covariate Shift.*

https://www.youtube.com/watch?v=tNIpEZLv_eg

30

Imperial College London

## Batch Normalization for Hidden Layers




*Can we normalize $z^{(3)}$ (or $a^{(3)}$) in order to stabilize the calculation of weights $\boldsymbol{\theta}^{(3)}$?*

https://www.youtube.com/watch?v=tNIpEZLv_eg

31

Imperial College London

## Batch Normalization on $z^{(l)}$ vector (could be on $a^{(l)}$)

1. Normalize $z^{(l)}$ vector at layer $l$ :

$$z^{(l)}_{norm} = \frac{z^{(l)} - \mu_z}{\sqrt{\sigma_z^2 + \varepsilon^2}}$$

(with $\mu_z$ and $\sigma_z^2$ mean and variance of $z^{(l)}$ calculated separately on each mini-batch, and $\varepsilon$ small parameter useful if $\sigma_z^2 = 0$ )

2. Apply Linear Transform to $z^{(l)}_{norm}$, with trainable parameters $\beta^{(l)}$ and $\gamma^{(l)}$ . For each component $j$ of $z^{(l)}_{norm}$

$$\tilde{z}_j^{(l)} = \gamma_j^{(l)}\, z_{j\,norm}^{(l)} + \beta_j^{(l)}$$ ⟹ Optimize mean and variance of $\tilde{z}^{(l)}$

(note that if $\gamma_j^{(l)} = \sqrt{\sigma_z^2 + \varepsilon^2}$ and $\beta_j^{(l)} = \mu_z$ then $\tilde{z}_j^{(l)} = z_{j\,norm}^{(l)}$)

3. Use $\tilde{z}^{(l)}$ as input to the activation function: $a^{(l)} = g(\tilde{z}^{(l)})$

https://www.youtube.com/watch?v=tNIpEZLv_eg

32

Imperial College London

## Batch Normalization

- Performs pre-processing at each layer of the network, instead of just on input layer
- Accelerates Training by stabilizing the layer changes through the iterations
- Provides some kind of Regularization, as Drop-Out, but not as strong.

33

Imperial College London

## Classic Batch Norm Paper (>15000 citations!)

Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift

Sergey Ioffe
Google Inc., *sioffe@google.com*

Christian Szegedy
Google Inc., *szegedy@google.com*

**Abstract**

Training Deep Neural Networks is complicated by the fact that the distribution of each layer's inputs changes during training, as the parameters of the previous layers change. This slows down the training by requiring lower learning rates and careful parameter initialization, and makes it notoriously hard to train models with saturating nonlinearities. We refer to this phenomenon as *internal covariate shift*, and address the problem by normalizing layer inputs. Our method draws its strength from making normalization a part of the model architecture and performing the normalization *for each training mini-batch*. Batch Normalization allows us to use much higher learning rates and be less careful about initialization. It also acts as a regularizer, in some cases eliminating the need for Dropout. Applied to a state-of-the-art image classification model, Batch Normalization achieves the same accuracy with 14 times fewer training steps, and beats the original model by a significant margin. Using an ensemble of batch-normalized networks, we improve upon the best published result on ImageNet classification: reaching 4.9% top-5 validation error (and 4.8% test error), exceeding the accuracy of human raters.

**1 Introduction**

Deep learning has dramatically advanced the state of the art in vision, speech, and many other areas. Stochastic

Using mini-batches of examples, as opposed to one example at a time, is helpful in several ways. First, the gradient of the loss over a mini-batch is an estimate of the gradient over the training set, whose quality improves as the batch size increases. Second, computation over a batch can be much more efficient than $m$ computations for individual examples, due to the parallelism afforded by the modern computing platforms.

While stochastic gradient is simple and effective, it requires careful tuning of the model hyper-parameters, specifically the learning rate used in optimization, as well as the initial values for the model parameters. The training is complicated by the fact that the inputs to each layer are affected by the parameters of all preceding layers – so that small changes to the network parameters amplify as the network becomes deeper.

The change in the distributions of layers' inputs presents a problem because the layers need to continuously adapt to the new distribution. When the input distribution to a learning system changes, it is said to experience *covariate shift* (Shimodaira, 2000). This is typically handled via domain adaptation (Jiang, 2008). However, the notion of covariate shift can be extended beyond the learning system as a whole, to apply to its parts, such as a sub-network or a layer. Consider a network computing

$$\ell = F_2(F_1(\mathrm{u}, \Theta_1), \Theta_2)$$

where $F_1$ and $F_2$ are arbitrary transformations, and the parameters $\Theta_1, \Theta_2$ are to be learned so as to minimize

https://www.youtube.com/watch?v=tNIpEZLv_eg

34

Imperial College London

## Regularization, Bias and Variance

1. Overfitting and Underfitting, Bias and Variance
2. Regularization
3. Batch Normalization
4. Regression and Classification Criteria
5. Training Set, Validation Set and Test Set
6. K-Fold Validation

35

Imperial College London

## L2-Norm: the Standard Regression Loss Function

If $x^{(i)}$ is a feature data vector and $y^{(i)}$ the associated real vector target, the Regression algorithm usually minimizes the average (squared) L2 norm, or Euclidian norm:

$$J(\theta) = \frac{1}{2m}\sum_{i=1}^{m}\|y^{(i)} - h_\theta(x^{(i)})\|_2^2$$

This is because the derivatives of $J(\theta)$ are easy to calculate for back-propagation.

This norm is also used to evaluate the quality of the model after Training.

Other criteria for evaluating the model after Training could be the L1 norm or the number of non-zero elements (sometimes wrongly called the L0 norm).

36

36

Imperial College London

## Cross-Entropy and Accuracy for Classification

If $x^{(i)}$ is a feature data vector and $y^{(i)}$ the associated hot-encoded class vector, the Classification algorithm usually minimizes the average cross-entropy.

This is because the derivatives of the cross-entropy are easy to calculate for back-propagation.

But the quality of the model after Training is usually evaluated using the Accuracy:

$$Accuracy = \frac{Number\ of\ Correctly\ Classified\ Examples}{Total\ Number\ of\ Examples}$$

But we may need to analyze the results more deeply…

37

37

Imperial College London

## A Classification Criteria: The Confusion Matrix

**MNIST Example**

| Predict / Actual | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 585 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 2 | 1 |
| 1 | 0 | 666 | 2 | 2 | 0 | 0 | 0 | 0 | 2 | 2 |
| 2 | 1 | 10 | 569 | 2 | 1 | 0 | 3 | 6 | 3 | 1 |
| 3 | 3 | 2 | 6 | 578 | 0 | 13 | 0 | 2 | 6 | 3 |
| 4 | 1 | 2 | 7 | 0 | 561 | 1 | 1 | 1 | 2 | 8 |
| 5 | 1 | 0 | 0 | 7 | 2 | 521 | 3 | 0 | 6 | 2 |
| 6 | 8 | 0 | 1 | 0 | 8 | 2 | 569 | 1 | 3 | 0 |
| 7 | 2 | 5 | 6 | 3 | 3 | 2 | 0 | 595 | 1 | 10 |
| 8 | 3 | 8 | 5 | 7 | 2 | 9 | 5 | 0 | 539 | 7 |
| 9 | 5 | 2 | 0 | 4 | 9 | 3 | 0 | 8 | 7 | 557 |

38

38

Imperial College London

## Rewriting the Confusion Matrix for a Binary Problem

| | | Predicted: Negative | Predicted: Positive |
|---|---|---|---|
| Actual | Negative | True Negative | False Positive |
| | Positive | False Negative | True Positive |



https://towardsdatascience.com/accuracy-precision-recall-or-f1-331fb37c5cb9



Imperial College London

## Precision and Recall for Unbalanced Classes

| | | Predicted/Classified: Negative | Predicted/Classified: Positive |
|---|---|---|---|
| Actual | Negative | 998 | 0 |
| | Positive | 1 | 1 |

*What is the accuracy of this model?* $\frac{998+1}{1000} = 0.999$

*What if the false negative corresponds to a person sick with COVID, to a major fraud case or to a dangerous terrorist?* ***The impact may be big!!***



https://towardsdatascience.com/accuracy-precision-recall-or-f1-331fb37c5cb9

Imperial College London

## The Recall criteria

$$\text{Recall} = \frac{True\ Positive}{True\ Positive + False\ Negative}$$

| | | Predicted: Negative | Predicted: Positive |
|---|---|---|---|
| Actual | Negative | True Negative | False Positive |
| | Positive | False Negative | True Positive |

*Recall is the proportion of true positive which are properly predicted. Good if you want to minimize the number of false negative (example of contagious patient or terrorist detection).*


https://towardsdatascience.com/accuracy-precision-recall-or-f1-331fb37c5cb9



Imperial College London

## The Precision criteria

$$\text{Precision} = \frac{True\ Positive}{True\ Positive + False\ Positive}$$

| | | Predicted: Negative | Predicted: Positive |
|---|---|---|---|
| Actual | Negative | True Negative | False Positive |
| | Positive | False Negative | True Positive |

*Precision is the proportion of predicted positive which are properly predicted. Good if you want to minimize the number of false positives (example of spam detection).*


https://towardsdatascience.com/accuracy-precision-recall-or-f1-331fb37c5cb9

Imperial College London

## The F1 Score

*The F1 Score is the harmonic mean of Precision and Recall*

$$\frac{1}{F1} = \frac{1}{2}\left(\frac{1}{Precision} + \frac{1}{Recall}\right)$$

*The highest possible value of $F_1$ is 1, indicating perfect precision and recall, and the lowest possible value is 0, if either the precision or the recall is zero.*

43

43

Imperial College London

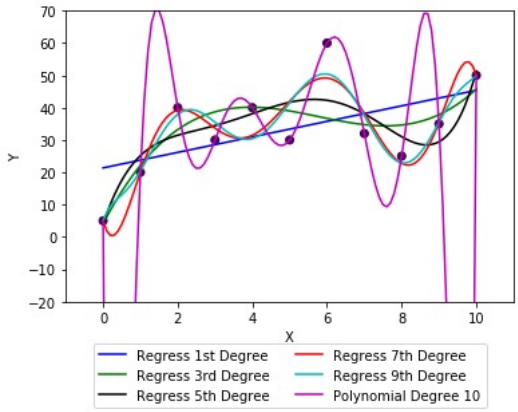
## Regularization, Bias and Variance

1. **Overfitting and Underfitting, Bias and Variance**
2. **Regularization**
3. **Batch Normalization**
4. **Regression and Classification Criteria**
5. **Training Set, Validation Set and Test Set**
6. **K-Fold Validation**

44

Imperial College London

## Change of Interpolating Function as Degree Increases

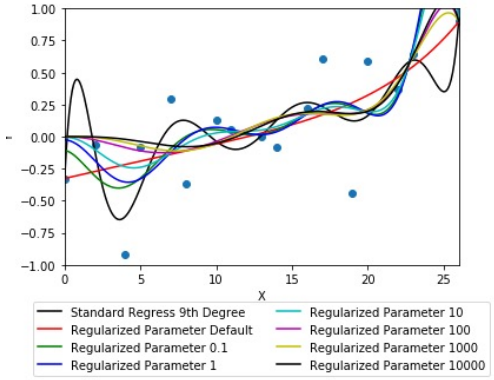



*How can we choose the "Best" Polynomial Degree?*

45

Imperial College London

## Changing the Regularization Parameter




*How can we choose the "Best" Regularization Parameter?*

46

Imperial College London

## What is a Hyperparameter?

A **hyperparameter** is a neural network parameter whose value is set before the Training process begins. It does not change during Training. By contrast, the values of parameters $\theta$ are derived via Training.

*Exercise: Give Examples of Hyper-Parameters of a Neural Network*

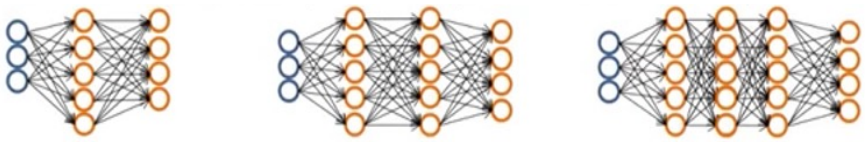
*Batch Size, Learning Rate, Optimization Approach used, Number of Layers, Number of Hidden Units, Regularization Parameter…*

47

Imperial College London

## Change of Network Architecture

*From one to three Hidden Layers*



*How can we choose the "Best" Network Architecture?*

*Two issues to consider: which criteria to apply (this has already been discussed) , and **on which data**?*

48

48

Imperial College London

## The Need for a Validation Set

*In a Supervised Learning context, the Neural Network parameters (or weights) are trained on the Training Set, but tested on the Test Set.*

*The Test Set constitutes the unseen data, it should not be used to calculate the parameters of the Neural Network … or to optimize the Hyperparameters.*

Hence, to optimize the Hyperparameters, we need a third Set, the Validation Set!

49

Imperial College London

## Create Validation Set to Optimize Hyperparameters

Split Data Into Three Sets:
*Training Set* (~70%) , *Validation Set* (~15%) , *Test Set* (~15%)

| ***Training Set*** | ***Validation Set*** | ***Test Set*** |
|---|---|---|

***Example of MNIST:*** *the Training Set contains 60,000 images, the « official » Test Set contains 10,000 images. This ratio is usually appropriate for this size of datasets. For very large datasets (say 100.000's to millions), split between Training and Test Set sizes can be smaller and be as low as 90%-10%.*
*Note that MNIST does not have a pre-defined Validation Set. This has to be chosen and extracted from the Training Set by the user.*

50

Imperial College London

## Use Validation Set to Optimize Hyperparameters

For each possible choice of Neural Network Hyperparameters:

1. Train Neural Network Parameters on Training Set
2. Evaluate Performance on the Validation Set

Then:

4. Pick the Hyperparameters that give the best performance on the Validation Set.
5. Re-train the new Neural Network on Training+Validation Set.
6. Test the Neural Network on the Test Set

*The Test Set is the final measure of performance but must never be used in the Training!*

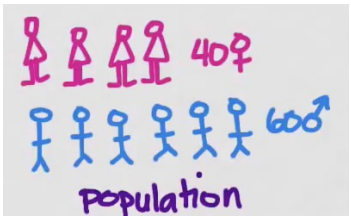
51

Imperial College London

## Sampling the Validation Set (and possibly theTest Set)

*In Classification problems, make sure the proportions of each class are the same for Training, Validation and possibly Test Sets. Take the example of a 20% Validation Set.*

*How to create a 20% Validation Set from this population?*




*If I randomly select 20 individuals (that is 20%) from this group of 40 women and 60 men, it is very unlikely that the group of 20 will be exactly composed of 8 (that is 20%) women and 12 (that is 20%) men!*

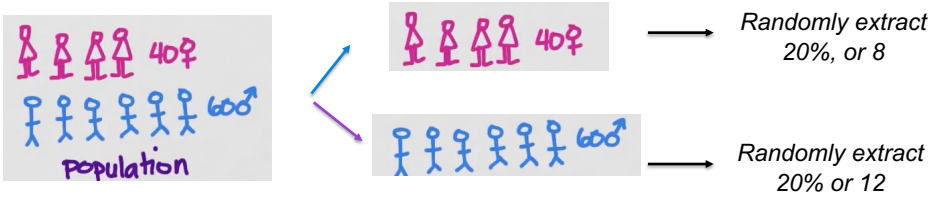
*For smaller data sets, use Stratified Random Sampling (class by class) instead of global sampling over the whole set.*

52

Imperial College London

## Stratified Sampling: *Create Validation Set of 20%*

*Sort Data into two "Strata"*



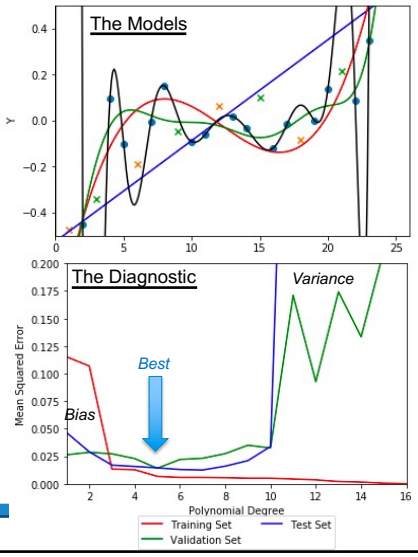

*First sort the population into "strata", then sample from each strata!*

53

Imperial College London

## Different Datasets for Polynomial Degree Choice


17 Training data
5 Validation data
5 Test Data




54

Imperial College London

## $J_{train}(\theta)$ and $J_{val}(\theta)$ for picking the Regularization Parameter

The Training Set has $m$ data points. The optimized Loss Function is (if L2 norm):

$$J(\theta) = \frac{1}{2m}\left(\sum_{i=1}^{m}\left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)^2 + \lambda \sum_{j=1}^{n}\theta_j^2\right)$$

For each tested value of $\lambda$, train the network on the Training Set, and evaluate the errors:

$$J_{train}(\theta) = \frac{1}{2m}\sum_{i=1}^{m}\left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)^2$$

And on the Validation Set: $J_{val}(\theta) = \frac{1}{2m_{val}}\sum_{i=1}^{m_{val}}\left(h_\theta\left(x_{val}^{(i)}\right) - y_{val}^{(i)}\right)^2$

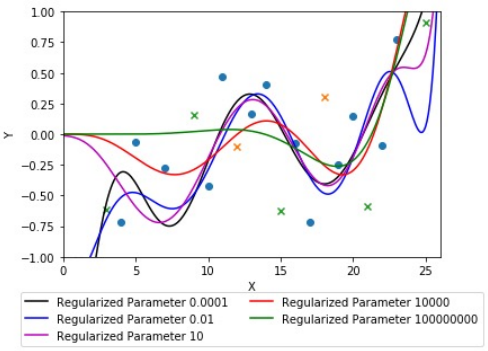
On the Test Set: $J_{test}(\theta) = \frac{1}{2m_{test}}\sum_{i=1}^{m_{test}}\left(h_\theta\left(x_{test}^{(i)}\right) - y_{test}^{(i)}\right)^2$

55

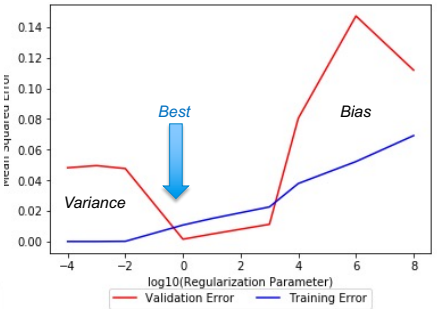
Imperial College London

## Optimizing the Regularization Parameter

Interpolating Function



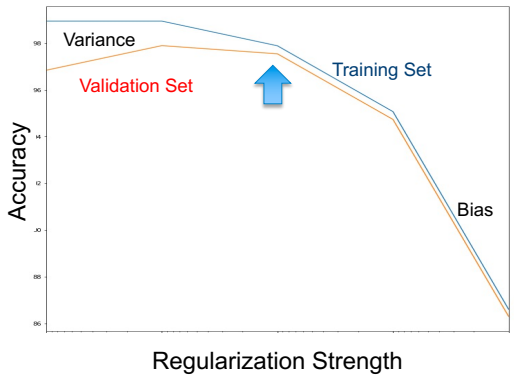

Mean Squared Error




56

Imperial College London

## Typical Regularization Behaviour in Classification




57

Imperial College London

## The Role of the Validation Set: Back to MNIST

The “official” MNIST Training Set is 60000 data. The “official” Test Set is 10000 and unknown to the user.

The user first needs to split the official Training Set into a new Training and Validation Sets, for example with 54000 and 6000 data points in each.

Then for a number of possible Hyperparameters:

1. Train Neural Network on Training Set
2. Test its Performance on Validation Set

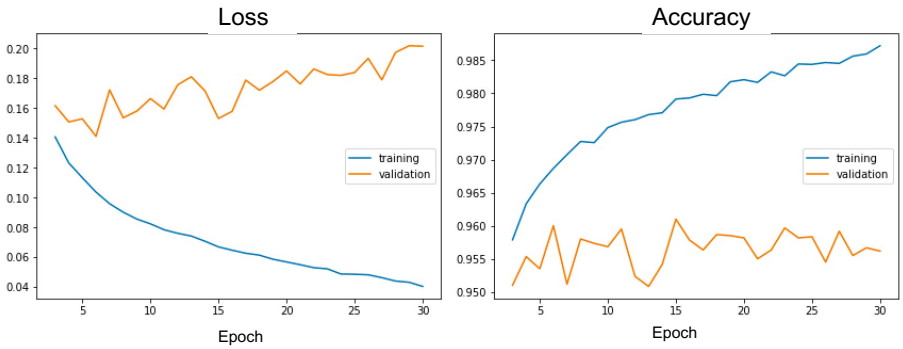
Choose the Hyperparameters that give the optimal performance on the Validation Set. Retrain the network on the official Training Set and finally evaluate its performance on the Test Set.



58

Imperial College London

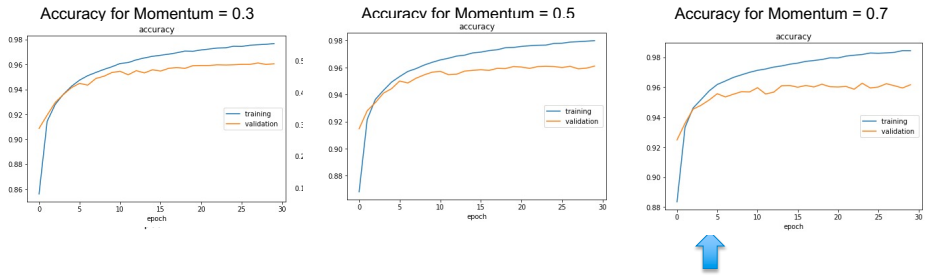
## MNIST: Using the Validation Set during Training




59

Imperial College London

## MNIST: Using the Validation Set during Training



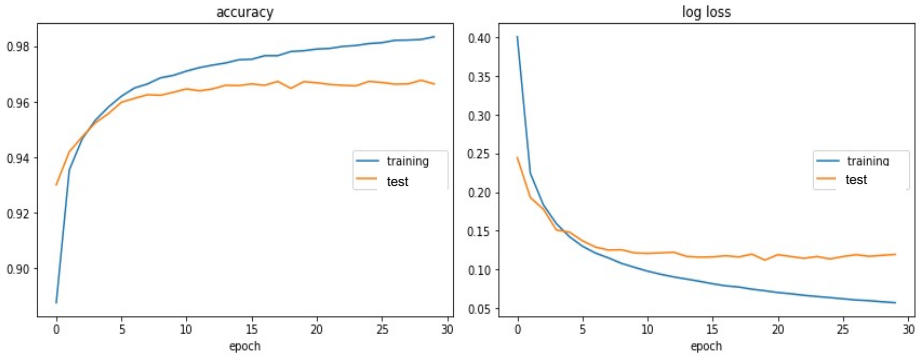

*Slightly Better Best Performance on Validation Set (all three Validation Accuracies close to .96)*

*"Grid Search" evaluates Validation Accuracy for a set of Hyperparameter(s)* values

60

Imperial College London

## MNIST: Final Training using Full Training Set

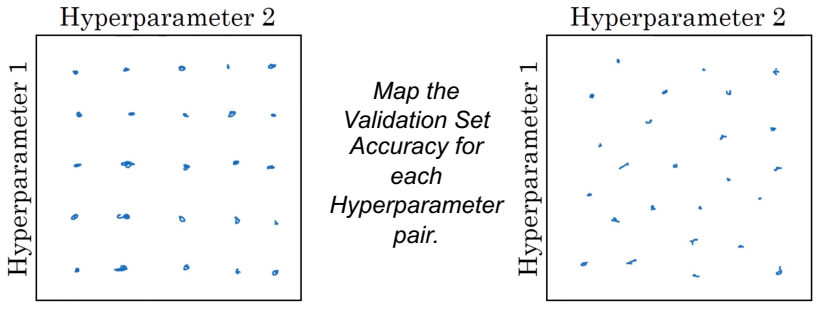



*Test Set accuracy at about .97!*

61

Imperial College London

## Grid Search to Optimize Multiple Hyperparameters




*Map the Validation Set Accuracy for each Hyperparameter pair.*

***Systematic Grid Search*** ***Random Grid Search***

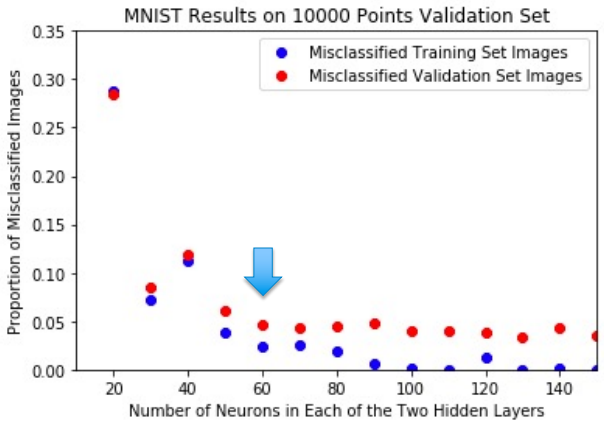
*If one parameter is less sensitive than the other, the random search provides a richer exploration of the values of the sensitive parameter.*

https://www.youtube.com/watch?v=AXDByU3D1hA

62

Imperial College London

## MNIST: Optimizing Number of Neurons in Layers



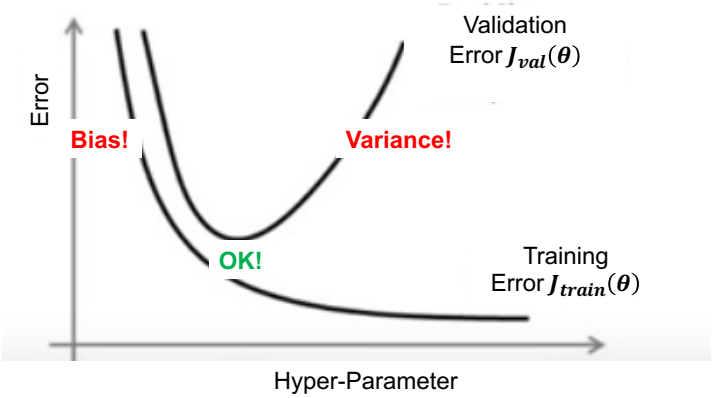

*Optimal value seems to be about 60 neurons in each of the two hidden layers.*

*This hyperparameter value gives a proportion of 0.04 misclassified images in the Test Set (same as Validation Set!)*

63

Imperial College London

## How to Identify Bias vs Variance Problems




64

Imperial College London

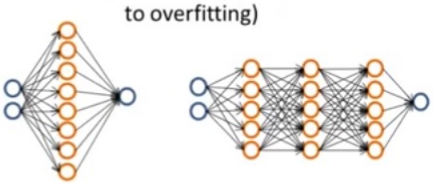
## Bias/Underfitting and Variance/Overfitting in NNs

**Neural networks and overfitting**

"Small" neural network (fewer parameters; more prone to underfitting)

Computationally cheaper

"Large" neural network (more parameters; more prone to overfitting)



Computationally more expensive.

Use regularization ($\lambda$) to address overfitting.

Andrew NG 10.7

65

Imperial College London

## General Guidelines for Improving Training

***To address Bias problems***

Try using more input features in order to increase the number of parameters
Try increasing the number of neurons
Try decreasing the regularization

***To address Variance problems***

Try decreasing the number of features
Try decreasing the number of neurons
Try increasing the regularization

66

Imperial College London

## MNIST : Impact of Number of Training Data


The larger the number of Training Data, the less chance of Overfitting!

67

Imperial College London

## Regularization, Bias and Variance

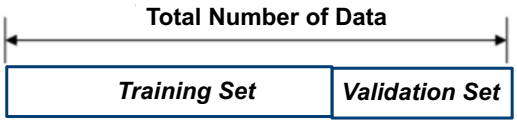
1. Overfitting and Underfitting, Bias and Variance
2. Regularization
3. Batch Normalization
4. Regression and Classification Criteria
5. Training Set, Validation Set and Test Set
6. K-Fold Validation

68

Imperial College London

## Limitation of using just one Validation Set (Hold-Out Method)

If Data Set is not very big, the role played by Training and Validation sets is not symmetrical. The approach used for the split may significantly affect the results.



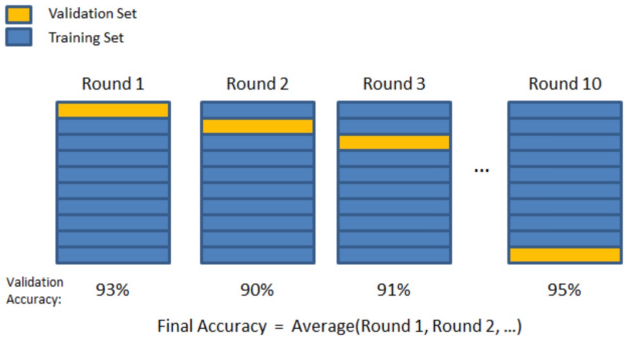

Possible approach: permutate between Validation and Training Sets and recalculate. Can do it if size of both is similar.

https://towardsdatascience.com/cross-validation-70289113a072

69

Imperial College London

## What is k-Fold Validation?




https://towardsdatascience.com/cross-validation-70289113a072

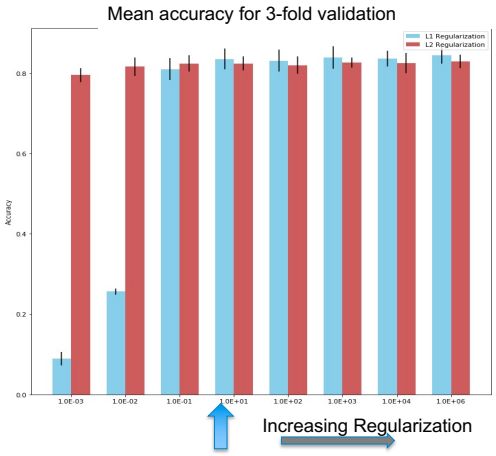
70

Imperial College London

## 3-Fold Validation on MNIST Example

| 1 | 2 | 3 |
|---|---|---|
| Validation | Train | Train |

*For each value of the Regularization Parameter, a 3-fold validation is run: three validation sets are created in turn and predicted using the rest of the data as training set.*




71

Imperial College London

## Summary for Regularization, Bias and Variance

- **Bias and Variance to characterize Underfitting versus Overfitting.**
- **L1 or L2 Regularizations as a cure against Overfitting.**
- **Model Averaging, Drop-out, Data Augmentation as Regularization techniques.**
- **Batch Normalization for accelerating Training (and Regularization)**
- **Importance of Training, Validation and Test Set for Hyper-parameters choice.**
- **K-Fold Validation for Hyper-Parameters Optimization.**

72

Imperial College London

## Exercise 1: Introduce Regularization into https://playground.tensorflow.org

## Exercise 2: Simple calculations of Precision and Recall on Dataset:

| | | Predicted | |
|---|---|---|---|
| | | Negative | Positive |
| Actual | Negative | 94,000 | 500 |
| | Positive | 3,500 | 2,000 |



Imperial College London

## Paper associated with Exercise 3

**Model Evaluation, Model Selection, and Algorithm Selection in Machine Learning**

**Sebastian Raschka**
University of Wisconsin–Madison
Department of Statistics
November 2018
sraschka@wisc.edu

**Abstract**

The correct use of model evaluation, model selection, and algorithm selection techniques is vital in academic machine learning research as well as in many industrial settings. This article reviews different techniques that can be used for each of these three subtasks and discusses the main advantages and disadvantages of each technique with references to theoretical and empirical studies. Further, recommendations are given to encourage best yet feasible practices in research and applications of machine learning. Common methods such as the holdout method for model evaluation and selection are covered, which are not recommended when working with small datasets. Different flavors of the bootstrap technique are introduced for estimating the uncertainty of performance estimates, as an alternative to confidence intervals via normal approximation if bootstrapping is computationally feasible. Common cross-validation techniques such as leave-one-out cross-validation and $k$-fold cross-validation are reviewed, the bias-variance trade-off for choosing $k$ is discussed, and practical tips for the optimal choice of $k$ are given based on empirical evidence. Different statistical tests for algorithm comparisons are presented, and strategies for dealing with multiple comparisons such as omnibus tests and multiple-comparison corrections are discussed. Finally, alternative methods for algorithm selection, such as the combined $F$-test 5x2 cross-validation and nested cross-validation, are recommended for comparing machine learning algorithms when datasets are small.