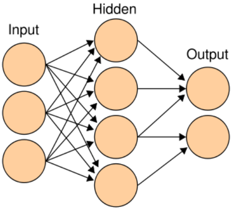
Supervised Learning: Performance Issues

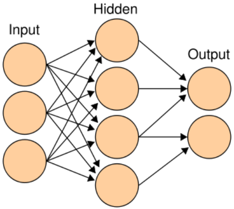
19 February 2020

Training Neural Nets
Decisions, decisions...



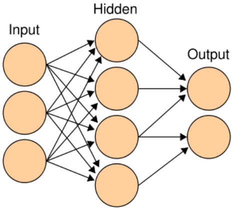
- How do we...
  - Prepare the data?
  - Initialise the weights?
  - Decide on the architecture?
  - Choose activation functions?
  - Choose a training algorithm?
  - Choose training algorithm parameters?
  - Measure model accuracy?
  - Measure model complexity?

Training Neural Nets
Decisions, decisions...



- How do we...
  - Prepare the data?
  - Initialise the weights?
  - Decide on the architecture?
  - Choose activation functions?
  - Choose a training algorithm?
  - Choose training algorithm parameters?
  - Measure model accuracy?
  - Measure model complexity?
- Trial and error

Training Neural Nets
Decisions, decisions...



- How do we...
  - Prepare the data?
  - Initialise the weights?
  - Decide on the architecture?
  - Choose activation functions?
  - Choose a training algorithm?
  - Choose training algorithm parameters?
  - Measure model accuracy?
  - Measure model complexity?
- Trial and error
- "Whatever I am familiar with!"

Data Preparation

- Correctly pre-processing your data takes you half-way to constructing a good model

Data Preparation

- Correctly pre-processing your data takes you half-way to constructing a good model

Representation

- Decide on the inputs and the outputs
- Remove obviously irrelevant inputs (names, unique IDs...)

Data Preparation

- Correctly pre-processing your data takes you half-way to constructing a good model

Representation

- Decide on the inputs and the outputs
- Remove obviously irrelevant inputs (names, unique IDs...)

Missing Values

Solutions?

Data Preparation

- Correctly pre-processing your data takes you half-way to constructing a good model

Representation

- Decide on the inputs and the outputs
- Remove obviously irrelevant inputs (names, unique IDs...)

Missing Values

Solutions?

- Remove the entire pattern if some data is missing

Data Preparation

- Correctly pre-processing your data takes you half-way to constructing a good model

Representation

- Decide on the inputs and the outputs
- Remove obviously irrelevant inputs (names, unique IDs...)

Missing Values

Solutions?

- Remove the entire pattern if some data is missing
- Replace the missing value with an average for that value over all patterns (continuous) or the most frequently occuring one (discrete)

Data Preparation

- Correctly pre-processing your data takes you half-way to constructing a good model

Representation

- Decide on the inputs and the outputs
- Remove obviously irrelevant inputs (names, unique IDs...)

Missing Values

Solutions?

- Remove the entire pattern if some data is missing
- Replace the missing value with an average for that value over all patterns (continuous) or the most frequently occuring one (discrete)
- Add an extra input unit to indicate if a parameter is missing

Data Preparation

Coding of the Inputs/Outputs

- All inputs/outputs must be numeric

Data Preparation

Coding of the Inputs/Outputs

- All inputs/outputs must be numeric
- In case of discrete values/labels, use **binary one-hot** encoding:

Data Preparation

Coding of the Inputs/Outputs

- All inputs/outputs must be numeric
- In case of discrete values/labels, use **binary one-hot** encoding:
    - A nominal input that takes n different values can be coded as n binary input parameters
    - For each pattern, input corresponding to a specific nominal value is set to 1, the rest are set to 0

Data Preparation

Coding of the Inputs/Outputs

- All inputs/outputs must be numeric
- In case of discrete values/labels, use **binary one-hot** encoding:
    - A nominal input that takes n different values can be coded as n binary input parameters
    - For each pattern, input corresponding to a specific nominal value is set to 1, the rest are set to 0
- Alternatively, use a single continuous value where every nominal value corresponds to a specific continuous value

Data Preparation

Coding of the Inputs/Outputs

- All inputs/outputs must be numeric
- In case of discrete values/labels, use **binary one-hot** encoding:
  - A nominal input that takes *n* different values can be coded as *n* binary input parameters
  - For each pattern, input corresponding to a specific nominal value is set to 1, the rest are set to 0
- Alternatively, use a single continuous value where every nominal value corresponds to a specific continuous value
  - Discrete characteristic is lost
  - Distance between categories: what does it represent?
  - The representation is more dense (less sparse)

Data Preparation

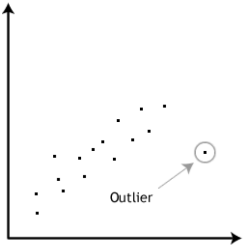
Coding of the Inputs/Outputs

- All inputs/outputs must be numeric
- In case of discrete values/labels, use **binary one-hot** encoding:
    - A nominal input that takes *n* different values can be coded as *n* binary input parameters
    - For each pattern, input corresponding to a specific nominal value is set to 1, the rest are set to 0
- Alternatively, use a single continuous value where every nominal value corresponds to a specific continuous value
    - Discrete characteristic is lost
    - Distance between categories: what does it represent?
    - The representation is more dense (less sparse)
- Which one is better: **dense or sparse**?

Data Preparation

Outliers

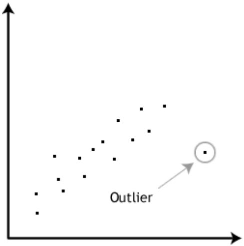
Outlier patterns produce large errors and divert the search



Data Preparation

Outliers

Outlier patterns produce large errors and divert the search



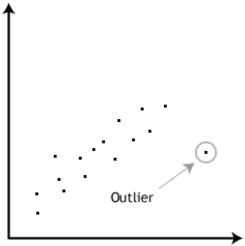
Solutions:

- Remove outliers using statistical techniques

Data Preparation

Outliers

Outlier patterns produce large errors and divert the search



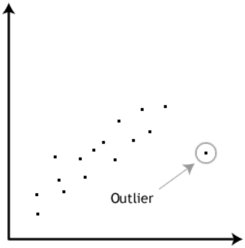
Solutions:

- Remove outliers using statistical techniques
- Use a robust objective function that is not influenced by outliers

Data Preparation

Outliers

Outlier patterns produce large errors and divert the search



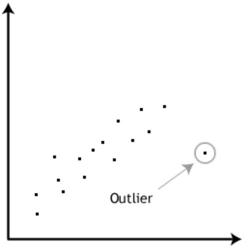
Solutions:

- Remove outliers using statistical techniques
- Use a robust objective function that is not influenced by outliers
  - if $|E| > \epsilon$, set E to a constant value

Data Preparation

Outliers

Outlier patterns produce large errors and divert the search



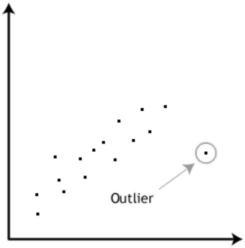
Solutions:

- Remove outliers using statistical techniques
- Use a robust objective function that is not influenced by outliers
  - if $|E| > \epsilon$, set E to a constant value
  - Problems with this approach?

Data Preparation

Outliers

Outlier patterns produce large errors and divert the search



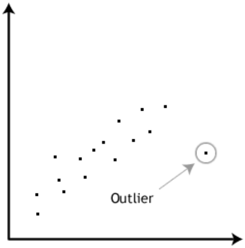
Solutions:

- Remove outliers using statistical techniques
- Use a robust objective function that is not influenced by outliers
  - if $|E| > \epsilon$, set E to a constant value
  - Problems with this approach?
  - if $|E| > \epsilon$, minimise $|E|$ instead of E^2

Data Preparation

Outliers

Outlier patterns produce large errors and divert the search



Solutions:

- Remove outliers using statistical techniques
- Use a robust objective function that is not influenced by outliers
  - if $|E| > \epsilon$, set E to a constant value
  - Problems with this approach?
  - if $|E| > \epsilon$, minimise $|E|$ instead of E^2
  - This approach is called “Huber loss” in statistics

Data Preparation
Scaling and Normalization

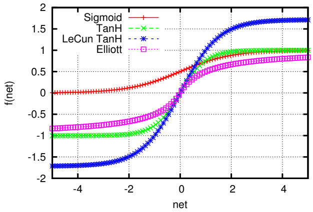
Scaling Inputs

- Inputs outside the active domain of the chosen activation function may cause saturation.

Data Preparation
Scaling and Normalization

Scaling Inputs

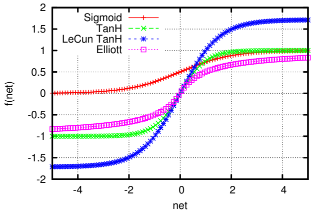
- Inputs outside the active domain of the chosen activation function may cause saturation.
- What is saturation and why is it bad?



Data Preparation
Scaling and Normalization

Scaling Inputs

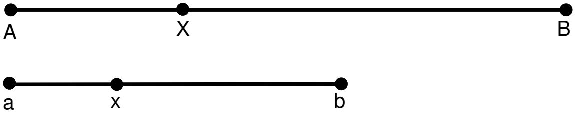
- Inputs outside the active domain of the chosen activation function may cause saturation.
- What is saturation and why is it bad?



Saturation

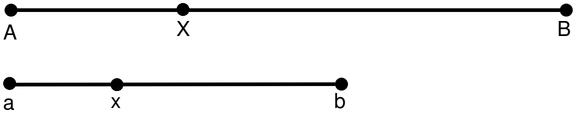
- Derivatives near assymptotes are close to 0 => slow learning
- A saturated output unit does not indicate the “confidence” level of the NN: all patterns, even the ones not fitted very well by the NN, will be classified with the same “strength”

Data Preparation
Linear (Min-Max) Scaling



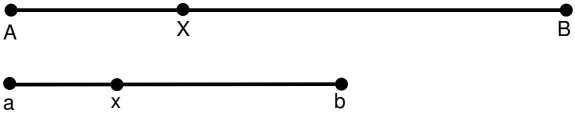
- Scale inputs/outputs to the necessary range linearly

Data Preparation
Linear (Min-Max) Scaling



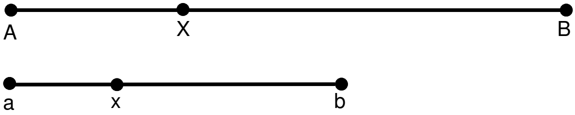
- Scale inputs/outputs to the necessary range linearly
- $\frac{X-A}{B-A} = \frac{x-a}{b-a}$
- $x = \frac{X-A}{B-A}(b-a) + a$
- x - scaled, X - unscaled, A, B - unscaled min and max, a, b - scaled min and max

Data Preparation
Linear (Min-Max) Scaling



- Scale inputs/outputs to the necessary range linearly
- $\frac{X-A}{B-A} = \frac{x-a}{b-a}$
- $x = \frac{X-A}{B-A}(b-a) + a$
- x - scaled, X - unscaled, A, B - unscaled min and max, a, b - scaled min and max
- Obvious disadvantage: you need to know the current range of values before you scale them to the desired range

Data Preparation
Linear (Min-Max) Scaling



- Scale inputs/outputs to the necessary range linearly
- $\frac{X-A}{B-A} = \frac{x-a}{b-a}$
- $x = \frac{X-A}{B-A}(b-a) + a$
- x - scaled, X - unscaled, A, B - unscaled min and max, a, b - scaled min and max
- Obvious disadvantage: you need to know the current range of values before you scale them to the desired range
- What if there is an outlier?

Data Preparation
Scaling and Normalization

Mean Centering: Mean of 0

- Convert the existing distribution to a “gaussian” one
- Average value of variable Z_i for all P: $\bar{Z}_i = \sum_{p=1}^{P} Z_{i,p}/P$
- Scale:
  - $Z_{i,p}^{M} = Z_{i,p} - \bar{Z}_i$

Data Preparation
Scaling and Normalization

Mean Centering: Mean of 0

- Convert the existing distribution to a “gaussian” one
- Average value of variable Z_i for all P: $\bar{Z}_i = \sum_{p=1}^{P} Z_{i,p}/P$
- Scale:
  - $Z_{i,p}^M = Z_{i,p} - \bar{Z}_i$

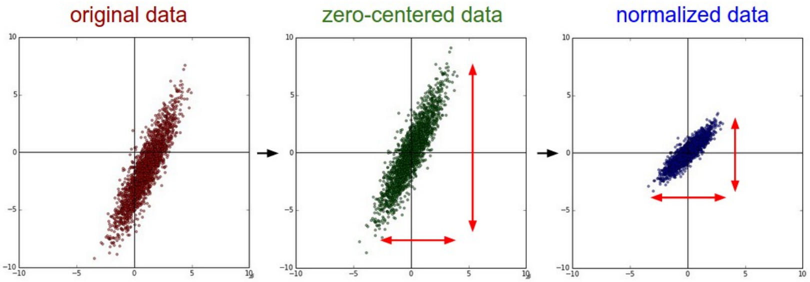
Variance Scaling: Variance of 1

- Let σ_{z_i} be the standard deviations of $Z_{i,p}$. Then:
  - $Z_{i,p}^V = \frac{Z_{i,p}}{\sigma_{z_i}}$

Data Preparation
Combine mean centering and variance scaling

Z-score (“standard score”) normalization

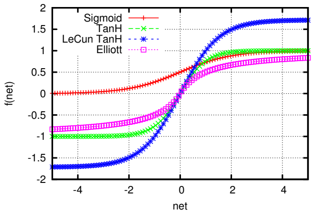
- Combine mean centering and variance scaling to normalize the data:
  - $Z^{MV}_{i,p} = \frac{Z_{i,p} - \bar{Z}_i}{\sigma_{z_i}}$



Data Preparation
Scaling and Normalization

Scaling Outputs

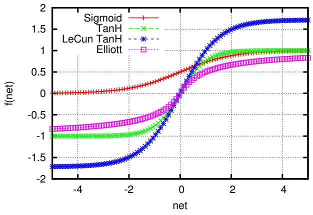
- Outputs outside of the activation function range will be unreachable
- NN will always produce a large error



Data Preparation
Scaling and Normalization

Scaling Outputs

- Outputs outside of the activation function range will be unreachable
- NN will always produce a large error



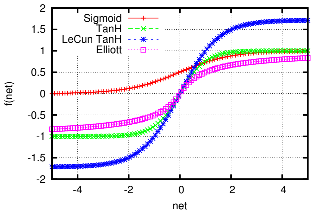
Outputs for bounded functions

- Make outputs reachable: scale to $[0, 1]$ for Sigmoid, $[-1, 1]$ for TanH, etc.

Data Preparation
Scaling and Normalization

Scaling Outputs

- Outputs outside of the activation function range will be unreachable
- NN will always produce a large error



Outputs for bounded functions

- Make outputs reachable: scale to $[0, 1]$ for Sigmoid, $[-1, 1]$ for TanH, etc.
- Trick of the trade: scale to $[0.1, 0.9]$ (Sigmoid) and $[-0.9, 0.9]$ (TanH) instead (why?)

Data Preparation
Do you have enough data?

- Many diverse examples are better than a few similar examples

Data Preparation
Do you have enough data?

- Many diverse examples are better than a few similar examples
- The existing data set can always be artificially expanded
    - Add noise sampled from a normal distribution with a small variance and zero mean to the existing patterns

Data Preparation
Do you have enough data?

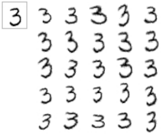
- Many diverse examples are better than a few similar examples
- The existing data set can always be artificially expanded
  - Add noise sampled from a normal distribution with a small variance and zero mean to the existing patterns
  - Image data: rotate, distort, zoom in and out, etc.

Data Preparation
Do you have enough data?

- Many diverse examples are better than a few similar examples
- The existing data set can always be artificially expanded
  - Add noise sampled from a normal distribution with a small variance and zero mean to the existing patterns
  - Image data: rotate, distort, zoom in and out, etc.
- Adding random noise at every epoch helps prevent overfitting - it becomes harder to memorise the exact patterns, so the NN learns the bigger picture instead

Data Preparation
Do you have enough data?

- Many diverse examples are better than a few similar examples
- The existing data set can always be artificially expanded
  - Add noise sampled from a normal distribution with a small variance and zero mean to the existing patterns
  - Image data: rotate, distort, zoom in and out, etc.
- Adding random noise at every epoch helps prevent overfitting - it becomes harder to memorise the exact patterns, so the NN learns the bigger picture instead
- Deep learning: first record-breaking MNIST-recognizing NNs modified the data set by artificial expansion (rotations, distortions, etc.)
- `http://yann.lecun.com/exdb/mnist/`



Weight Initialisation
Choosing the starting point

- Gradient descent starts on a single point: choosing a “bad” starting point can be fatal

Weight Initialisation
Choosing the starting point

- Gradient descent starts on a single point: choosing a “bad” starting point can be fatal
- Population-based algorithms such as PSO and GA use a population of points: not distributing them adequately can cause stagnation/premature convergence

Weight Initialisation

Choosing the starting point

- Gradient descent starts on a single point: choosing a “bad” starting point can be fatal
- Population-based algorithms such as PSO and GA use a population of points: not distributing them adequately can cause stagnation/premature convergence
- Can you set all weights to zero?

Weight Initialisation
Choosing the starting point

- Gradient descent starts on a single point: choosing a “bad” starting point can be fatal
- Population-based algorithms such as PSO and GA use a population of points: not distributing them adequately can cause stagnation/premature convergence
- Can you set all weights to zero?
- Can you set weights to random numbers?

Weight Initialisation
Choosing the starting point

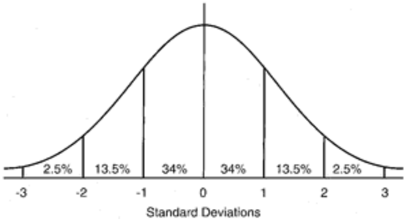
- Gradient descent starts on a single point: choosing a “bad” starting point can be fatal
- Population-based algorithms such as PSO and GA use a population of points: not distributing them adequately can cause stagnation/premature convergence
- Can you set all weights to zero?
- Can you set weights to random numbers?
- Random weights in a small range around 0:
  - Small net input signals
  - Midrange values produced by activation functions
  - I.e. no instant saturation => better learning potential

Weight Initialisation
Choosing the starting point

- Gradient descent starts on a single point: choosing a “bad” starting point can be fatal
- Population-based algorithms such as PSO and GA use a population of points: not distributing them adequately can cause stagnation/premature convergence
- Can you set all weights to zero?
- Can you set weights to random numbers?
- Random weights in a small range around 0:
  - Small net input signals
  - Midrange values produced by activation functions
  - I.e. no instant saturation => better learning potential

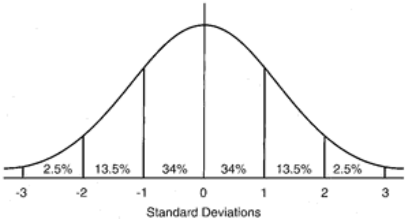
Weight Initialisation
Choosing the starting point

- Gaussian distribution can be used instead of uniform to sample the weights:



Weight Initialisation
Choosing the starting point

- Gaussian distribution can be used instead of uniform to sample the weights:



- Wessels and Barnard: choose random weights in range $\left[\frac{-1}{\sqrt{fanin}}, \frac{1}{\sqrt{fanin}}\right]$, where *fanin* is the number of incoming connections for the given unit.
  - The larger the architecture, the smaller interval will be used

Weight Initialisation
Choosing the starting point

- Xavier and Glorot (2010): sample random weights from a distribution with $\mu = 0$ and $\sigma = \frac{\sqrt{6}}{\sqrt{fanin+fanout}}$.
  - Derived to normalise (maintain variance of) the backpropagated gradients

Weight Initialisation
Choosing the starting point

- Xavier and Glorot (2010): sample random weights from a distribution with $\mu = 0$ and $\sigma = \frac{\sqrt{6}}{\sqrt{fanin+fanout}}$.
  - Derived to normalise (maintain variance of) the backpropagated gradients
- He Normal (He et al., 2015) [for rectified activations]:
  1. Choose weights randomly from a standard normal distribution.
  2. Multiply each weight by $\frac{\sqrt{2}}{\sqrt{fanin}}$.
  3. Bias weights are initialized to zero.

Weight Initialisation
Choosing the starting point

- Xavier and Glorot (2010): sample random weights from a distribution with $\mu = 0$ and $\sigma = \frac{\sqrt{6}}{\sqrt{fanin+fanout}}$.
    - Derived to normalise (maintain variance of) the backpropagated gradients
- He Normal (He et al., 2015) [for rectified activations]:
    1. Choose weights randomly from a standard normal distribution.
    2. Multiply each weight by $\frac{\sqrt{2}}{\sqrt{fanin}}$.
    3. Bias weights are initialized to zero.
- Are smaller weights always better?

Data set

Training, Generalisation, Validation

- Supervised training: data patterns with known target values are presented to the NN
- Data set has to be subdivided into three parts:

Training set

Data that the training algorithm will use to iteratively adjust the weights

Data set

Training, Generalisation, Validation

- Supervised training: data patterns with known target values are presented to the NN
- Data set has to be subdivided into three parts:

Training set

Data that the training algorithm will use to iteratively adjust the weights

Generalisation/Test set

Data that will be used to calculate the generalisation error during hyperparameter optimization

Data set

Training, Generalisation, Validation

- Supervised training: data patterns with known target values are presented to the NN
- Data set has to be subdivided into three parts:

Training set

Data that the training algorithm will use to iteratively adjust the weights

Generalisation/Test set

Data that will be used to calculate the generalisation error during hyperparameter optimization

Validation set

Data that will be used to calculate the generalisation error of the optimized model

Data set

Training, Generalisation, Validation

Separating the data

Using training/optimization data for testing is not fair: generalisation error will not be an objective measure

Data set
Training, Generalisation, Validation

Separating the data

Using training/optimization data for testing is not fair: generalisation error will not be an objective measure

Data normalisation

How would you scale the data in training, testing, validation subsets?

Data Scaling

Which one is correct?

Data Scaling

Which one is correct?

1. Scale the whole data set before splitting:

$$T_i = \frac{T_i - \bar{T}_{all}}{\sigma_{all}}$$

Data Scaling

Which one is correct?

1. Scale the whole data set before splitting:

$$T_i = \frac{T_i - \bar{T}_{all}}{\sigma_{all}}$$

2. Scale training and testing sets separately:

$$T_{i,train} = \frac{T_{i,train} - \bar{T}_{train}}{\sigma_{train}}, \; T_{i,test} = \frac{T_{i,test} - \bar{T}_{test}}{\sigma_{test}}$$

Data Scaling

Which one is correct?

1. Scale the whole data set before splitting:

$$T_i = \frac{T_i - \bar{T}_{all}}{\sigma_{all}}$$

2. Scale training and testing sets separately:

$$T_{i,train} = \frac{T_{i,train} - \bar{T}_{train}}{\sigma_{train}}, T_{i,test} = \frac{T_{i,test} - \bar{T}_{test}}{\sigma_{test}}$$

3. Apply training data transformation across the board:

$$T_{i,train} = \frac{T_{i,train} - \bar{T}_{train}}{\sigma_{train}}, T_{i,test} = \frac{T_{i,test} - \bar{T}_{train}}{\sigma_{train}}$$

Data Scaling

Which one is correct?

1. Scale the whole data set before splitting:

$$T_i = \frac{T_i - \bar{T}_{all}}{\sigma_{all}}$$

2. Scale training and testing sets separately:

$$T_{i,train} = \frac{T_{i,train} - \bar{T}_{train}}{\sigma_{train}}, \; T_{i,test} = \frac{T_{i,test} - \bar{T}_{test}}{\sigma_{test}}$$

3. Apply training data transformation across the board:

$$T_{i,train} = \frac{T_{i,train} - \bar{T}_{train}}{\sigma_{train}}, \; T_{i,test} = \frac{T_{i,test} - \bar{T}_{train}}{\sigma_{train}}$$

Option (3) is the correct approach.

Data set

Training, Generalisation, Validation

- Training set + generalisation set make up the training set for the optimized model
- How do we subdivide the data set?

Training set

Over 50%, up to 90% of the entire data set

Data set
Training, Generalisation, Validation

- Training set + generalisation set make up the training set for the optimized model
- How do we subdivide the data set?

Training set

Over 50%, up to 90% of the entire data set

Generalisation set

About 10% to 30% of the training set

Data set
Training, Generalisation, Validation

- Training set + generalisation set make up the training set for the optimized model
- How do we subdivide the data set?

Training set

Over 50%, up to 90% of the entire data set

Generalisation set

About 10% to 30% of the training set

Validation set

About 10% to 30% of the data set. Should never be used for training.

Training the NN
Stochastic, batch, mini-batch

Stochastic (on-line) training

- Update weights after each pattern
- Backprop each gradient
- Bound to fluctuate: is a single pattern representative?

Training the NN
Stochastic, batch, mini-batch

Stochastic (on-line) training

- Update weights after each pattern
- Backprop each gradient
- Bound to fluctuate: is a single pattern representative?

Batch training

- Update weights after all patterns were considered
- Backprop average gradient
- Stable, but computationally heavy

Training the NN
Stochastic, batch, mini-batch

Stochastic (on-line) training

- Update weights after each pattern
- Backprop each gradient
- Bound to fluctuate: is a single pattern representative?

Batch training

- Update weights after all patterns were considered
- Backprop average gradient
- Stable, but computationally heavy

Mini-Batch training: Best of both

- Calculate average gradient over a subset of patterns
- Very popular in deep learning

Training the NN
Stochastic, batch, mini-batch

Shuffling the data

- Training set must be shuffled at every epoch (iteration)

Training the NN
Stochastic, batch, mini-batch

Shuffling the data

- Training set must be shuffled at every epoch (iteration)
- If patterns are presented in the same order, the NN may infer the order of patterns and learn it

Training the NN
Stochastic, batch, mini-batch

Shuffling the data

- Training set must be shuffled at every epoch (iteration)
- If patterns are presented in the same order, the NN may infer the order of patterns and learn it
- If subsets of the data set are used, the subsets may not be representative unless the data set is shuffled

Training the NN
Stochastic, batch, mini-batch

Shuffling the data

- Training set must be shuffled at every epoch (iteration)
- If patterns are presented in the same order, the NN may infer the order of patterns and learn it
- If subsets of the data set are used, the subsets may not be representative unless the data set is shuffled
- How does this apply to stochastic, batch, mini-batch training?