Bachelor's Thesis

# Financial prediction with Neural Networks

## "Predicting Spread Prices on VIX Futures with Deep Learning"

by Thomas Leyh

University of Freiburg
Department of Computer Science
Computer Vision Group
Prof. Dr. Thomas Brox

September 12th, 2017

# Outline

1. Why do this?
   - Some financial terminology
   - The actual problem
   - Where machine learning comes in
2. Approaches
   - Data representation
   - Network architecture
3. Conclusion
   - Results
   - Why did it fail?
   - Where to go from here?

# Outline

# Some financial terminology

**VIX** *CBOE's Volatility Index* based on the *S&P 500 Index* (similar to the German *DAX*).

**Volatility** Measures the risk of sharp market movements.

**Futures** A standardized tradable contract. The holder is guaranteed the delivery of the underlying commodity at its expiration.

**Term structure** Formed by multiple futures with different expirations.

# Some financial terminology

**VIX** *CBOE's Volatility Index* based on the *S&P 500 Index* (similar to the German *DAX*).

**Volatility** Measures the risk of sharp market movements.

**Futures** A standardized tradable contract. The holder is guaranteed the delivery of the underlying commodity at its expiration.

**Term structure** Formed by multiple futures with different expirations.

# Some financial terminology

**VIX** *CBOE's Volatility Index* based on the *S&P 500 Index* (similar to the German *DAX*).

**Volatility** Measures the risk of sharp market movements.

**Futures** A standardized tradable contract. The holder is guaranteed the delivery of the underlying commodity at its expiration.

**Term structure** Formed by multiple futures with different expirations.

# Some financial terminology

**VIX** *CBOE's Volatility Index* based on the *S&P 500 Index* (similar to the German *DAX*).

**Volatility** Measures the risk of sharp market movements.

**Futures** A standardized tradable contract. The holder is guaranteed the delivery of the underlying commodity at its expiration.

**Term structure** Formed by multiple futures with different expirations.

Source: Jean Weber, INRA (Wikimedia Commons)

# Some financial terminology

**VIX** *CBOE's Volatility Index* based on the *S&P 500 Index* (similar to the German *DAX*).

**Volatility** Measures the risk of sharp market movements.

**Futures** A standardized tradable contract. The holder is guaranteed the delivery of the underlying commodity at its expiration.

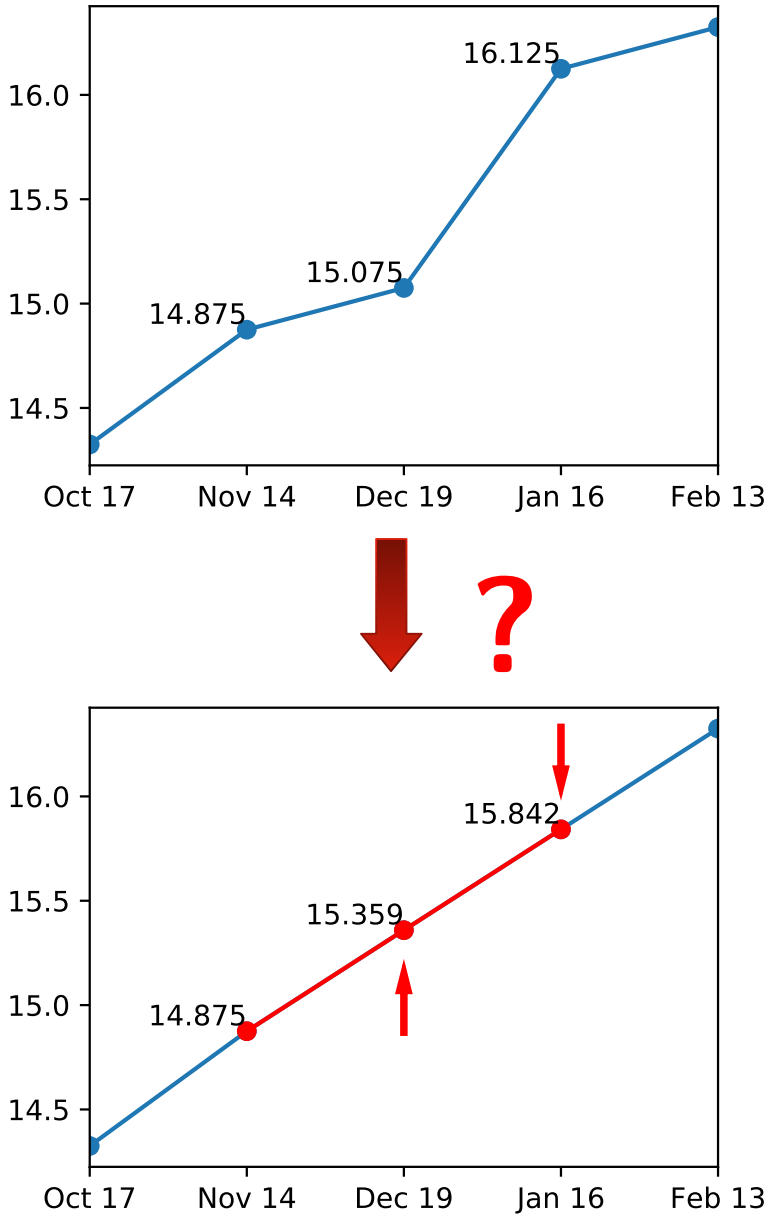
**Term structure** Formed by multiple futures with different expirations.

Source: Jean Weber, INRA (Wikimedia Commons)

# The actual problem

We are interested in the correction of the term structure:

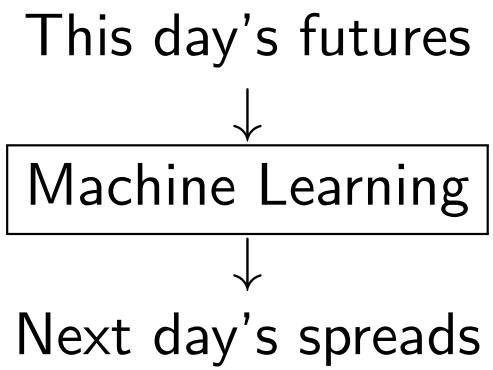
- Will it get corrected?
  Do we invest in the first place?
- At what position?
  Where to place the spread?
- When will the correction occur?
  How long to hold the spread?



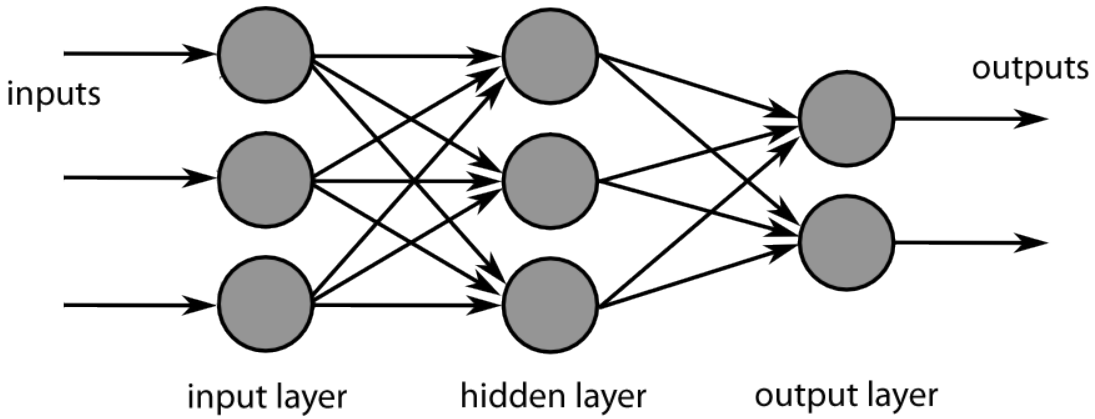
# Where machine learning comes in

We want some working model for this prediction.
But we are too lazy to build it ourselves.
$\rightarrow$ Let the computer do it.

1. Select appropriate model family
2. Optimize hyperparameters
3. Throw data at it
4. ...
5. Profit



# Where machine learning comes in: Neural Networks




Source: Chrislb (Wikimedia Commons)

Equivalent to:

$$f^{(1)}\left(\underbrace{\begin{bmatrix} y_1^{(0)} \\ y_2^{(0)} \\ y_3^{(0)} \end{bmatrix}^\top}_{\text{inputs}} \cdot \underbrace{\begin{bmatrix} w_{1,1}^{(1)} & w_{1,2}^{(1)} & w_{1,3}^{(1)} \\ w_{2,1}^{(1)} & w_{2,2}^{(1)} & w_{2,3}^{(1)} \\ w_{3,1}^{(1)} & w_{3,2}^{(1)} & w_{3,3}^{(1)} \end{bmatrix}}_{\textit{weights}}\right) = \underbrace{y^{(1)\top}}_{\text{hidden layer}} \tag{1}$$

$$f^{(2)}\left(y^{(1)\top} W^{(2)}\right) = \underbrace{y^{(2)\top}}_{\text{outputs}} \tag{2}$$

with often very simple activation function $f$ working elementwise.

# Outline

# Data representation

- There were 2656 samples used
  - From October 23th, 2006 to May 11th, 2017
  - Split with ratio 70:15:15 for training, validation and testing
  - Optionally using minutely data
- Preprocessing of the input data – today's futures –
  - The difference between the term structures' legs was used
  - A combination of additional inputs was used, mostly *days until expiration*
  - Optionally centered and normalized
- Choosing the target data – tomorrow's spreads –
  - Predict all spreads at once
    or
  - Predict one spread at a time
    or
  - Just predict if prices will fall, rise or stay the same

# Data representation

- There were 2656 samples used
  - From October 23th, 2006 to May 11th, 2017
  - Split with ratio 70:15:15 for training, validation and testing
  - Optionally using minutely data
- Preprocessing of the input data – today's futures –
  - The difference between the term structures' legs was used
  - A combination of additional inputs was used, mostly *days until expiration*
  - Optionally centered and normalized
- Choosing the target data – tomorrow's spreads –
  - Predict all spreads at once
    or
  - Predict one spread at a time
    or
  - Just predict if prices will fall, rise or stay the same

# Data representation

- There were 2656 samples used
  - From October 23th, 2006 to May 11th, 2017
  - Split with ratio 70:15:15 for training, validation and testing
  - Optionally using minutely data
- Preprocessing of the input data – today's futures –
  - The difference between the term structures' legs was used
  - A combination of additional inputs was used, mostly *days until expiration*
  - Optionally centered and normalized
- Choosing the target data – tomorrow's spreads –
  - Predict all spreads at once

    or
  - Predict one spread at a time

    or
  - Just predict if prices will fall, rise or stay the same

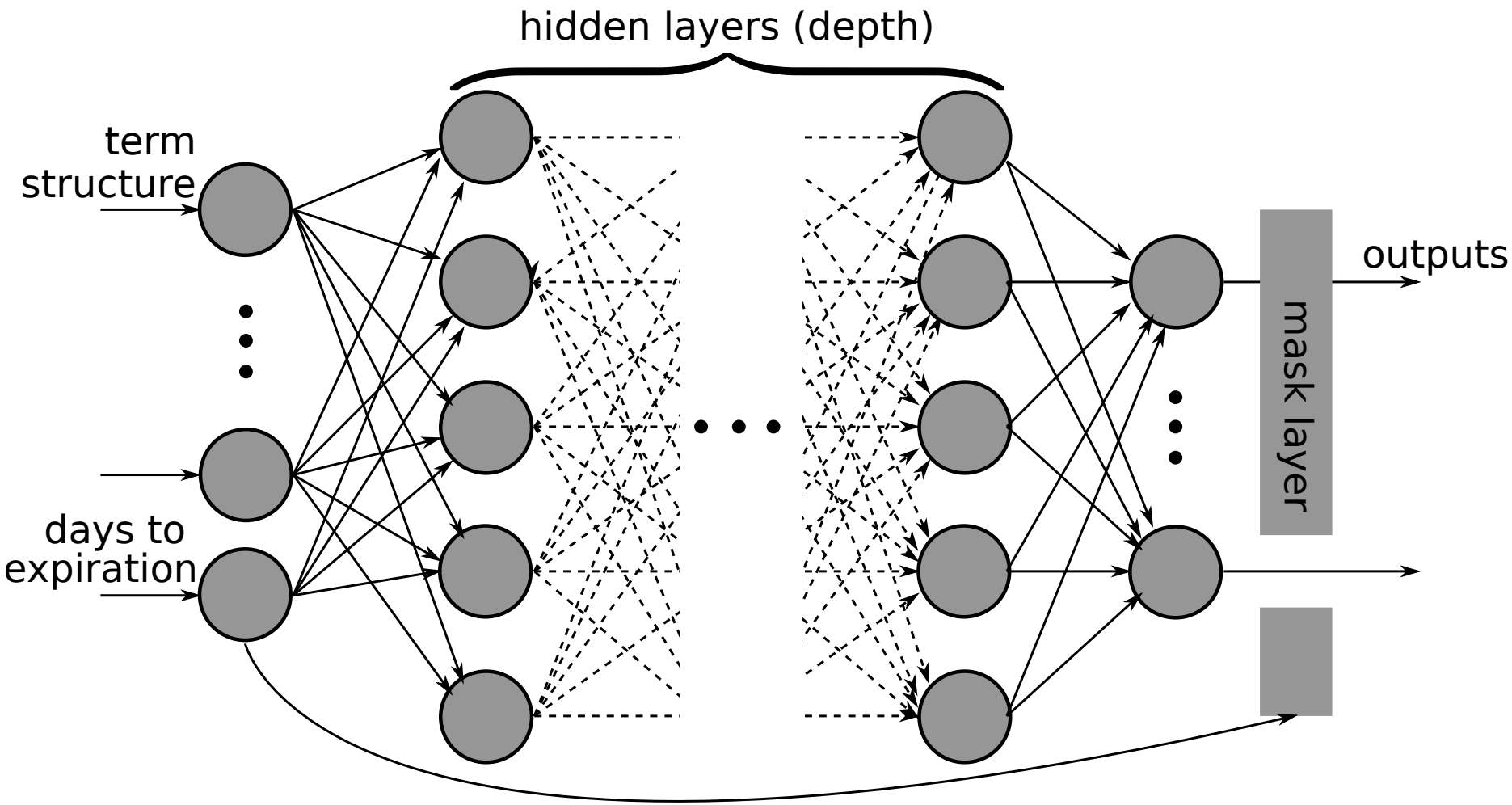
# Network architecture

Using Feedforward Neural Networks.
A very basic model where layers are simply stacked on top of each other.

Hyperparameters:

| | |
|---|---|
| Depth | Number of hidden layers |
| Width | Number of nodes per layer |
| Activations | For introducing nonlinearity at each at each node's ouput |
| Loss function | For comparing predictions and target values at the network's end (here: *mean squared error*) |
| Optimizer | For iteratively updating the weights (mostly *Adam*) |
| Regularizations | So the network doesn't just learn data by heart |
| ... | and many more ... |

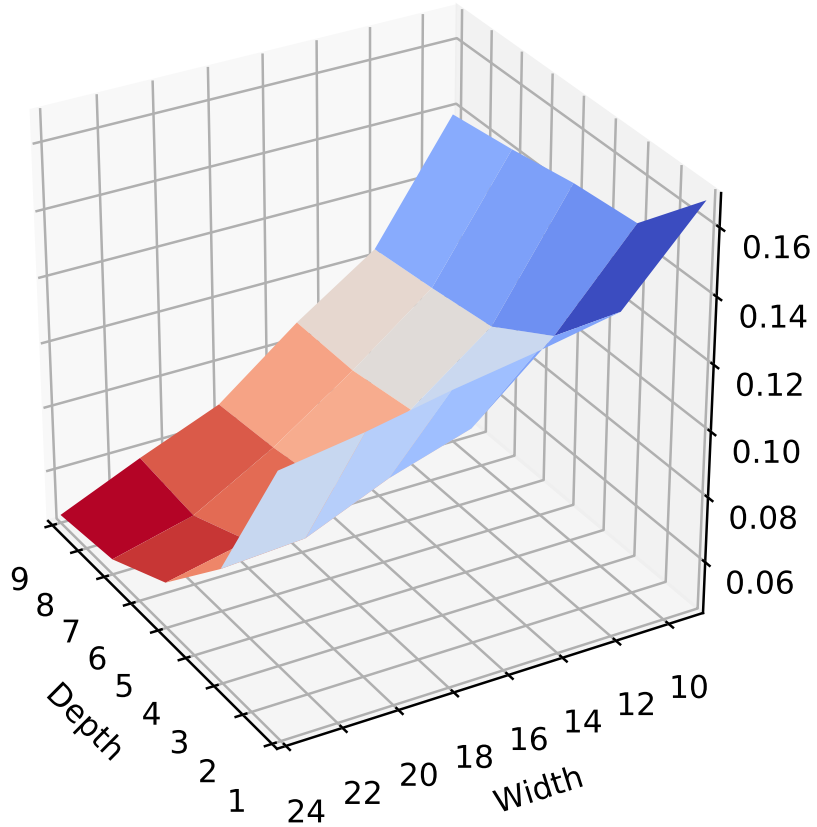
# Network architecture



# Outline

# Results

During training the loss mostly looks like this:

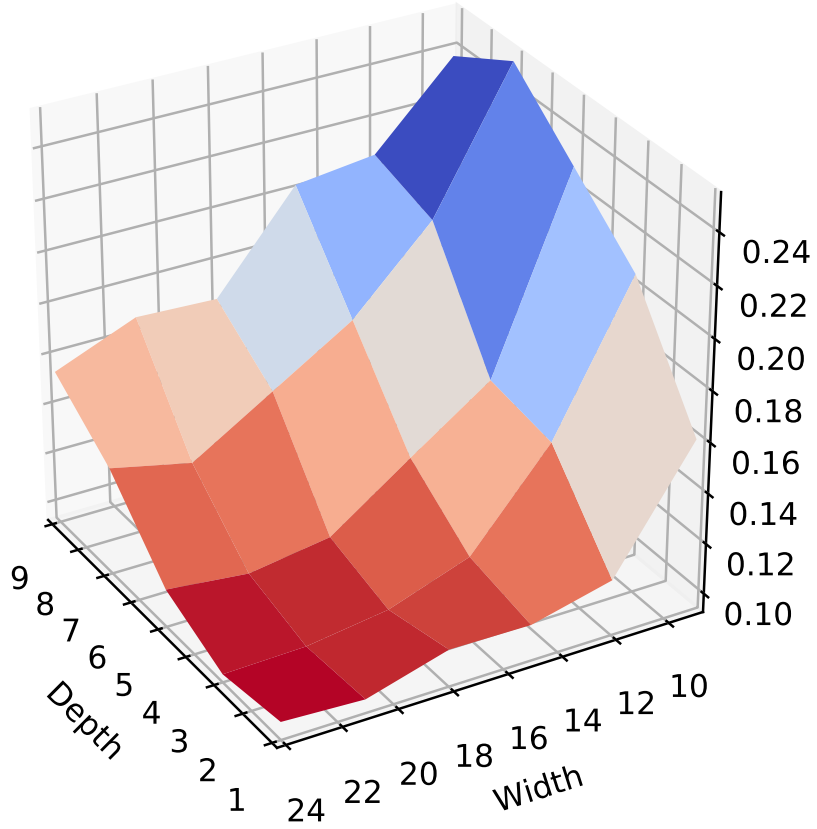



But that's just overfitting – learning the training data by heart.

Smaller loss is better, highlighted in red

# Results

The loss when looking at validation data (unseen by network):




Deep networks have terrible performance on the problem.

---

Smaller loss is better, highlighted in red

# Results

After finding good hyperparameters, how do we do on the test set?

| | Epoch | **Test Loss** |
|---|---|---|
| Naive | | 0.1318 |
| Basic | 307 | 0.1563 |
| With dropout | 174 | 0.2984 |
| Self-normalizing | 242 | 0.1259 |
| Minutely data | 70 | 0.1294 |
| Additional inputs | 268 | 0.1321 |
| Single spreads | 200 | 0.1760 |

Smaller loss is better, highlighted in red

# Results

After finding good hyperparameters, how do we do on the test set?

| | Epoch | **Test Loss** |
|---|---|---|
| Naive | !!! | 0.1318 |
| Basic | 307 | 0.1563 |
| With dropout | 174 | 0.2984 |
| Self-normalizing | 242 | 0.1259 |
| Minutely data | 70 | 0.1294 |
| Additional inputs | 268 | 0.1321 |
| Single spreads | 200 | 0.1760 |

Our best model has equal performance as the naive approach. 

Smaller loss is better, highlighted in red

# Why did it fail?

Let's make the problem easier: Just predict if spread prices will:

(a) rise

(b) fall

(c) roughly stay the same

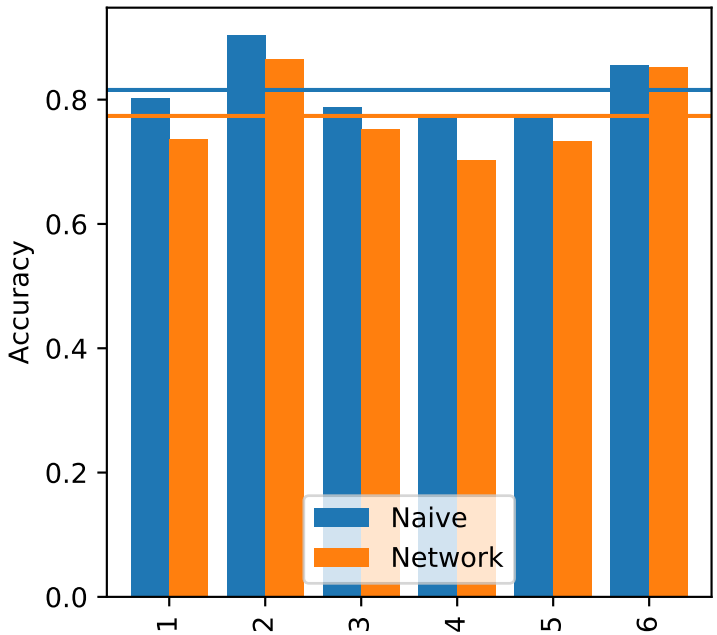
# Why did it fail?

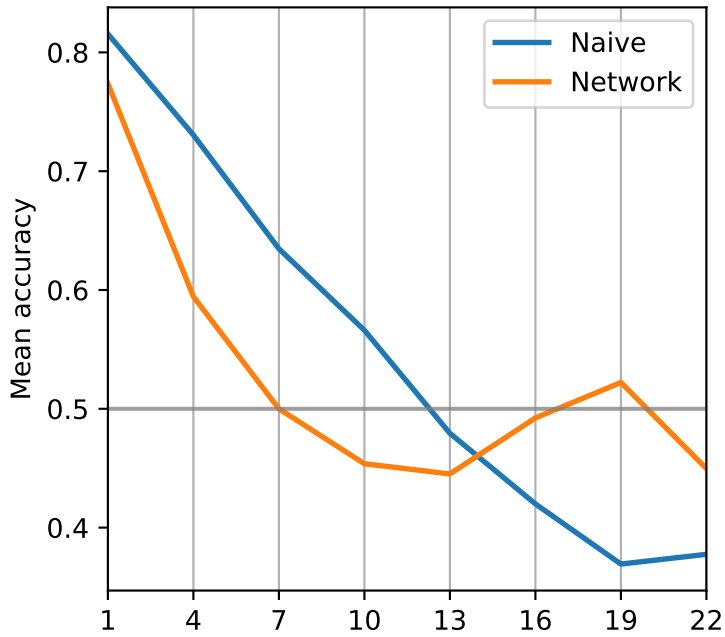
Let's make the problem easier: Just predict if spread prices will:

(a) rise

(b) fall

(c) roughly stay the same




For *Accuracy* like measured above, higher is better.

# Why did it fail?

Conclusions:

- Neural networks work best in vision, speech and language
  - These are easy for humans
  - Very high-dimensional (many inputs)
- Financial problems are mostly hard for humans
- Additionally few inputs were used here
- Therefore the network only learned an (bad) approximation of the naive solution

> *"Most tasks that consist of mapping an input vector to an output vector, and that are easy for a person to do rapidly, can be accomplished via deep learning..."*
>
> *[Goodfellow et al.: Deep Learning]*

# Why did it fail?

Conclusions:

- Neural networks work best in vision, speech and language
  - These are easy for humans
  - Very high-dimensional (many inputs)
- Financial problems are mostly hard for humans
- Additionally few inputs were used here
- Therefore the network only learned an (bad) approximation of the naive solution

> *"Most tasks that consist of mapping an input vector to an output vector, and that are easy for a person to do rapidly, can be accomplished via deep learning..."*
>
> *[Goodfellow et al.: Deep Learning]*

# Why did it fail?

Conclusions:

- Neural networks work best in vision, speech and language
  - These are easy for humans
  - Very high-dimensional (many inputs)
- Financial problems are mostly hard for humans
- Additionally few inputs were used here
- Therefore the network only learned an (bad) approximation of the naive solution

> *"Most tasks that consist of mapping an input vector to an output vector, and that are easy for a person to do rapidly, can be accomplished via deep learning..."*
>
> *[Goodfellow et al.: Deep Learning]*

# Why did it fail?

Conclusions:

- Neural networks work best in vision, speech and language
  - These are easy for humans
  - Very high-dimensional (many inputs)
- Financial problems are mostly hard for humans
- Additionally few inputs were used here
- Therefore the network only learned an (bad) approximation of the naive solution

> *"Most tasks that consist of mapping an input vector to an output vector, and that are easy for a person to do rapidly, can be accomplished via deep learning..."*
>
> *[Goodfellow et al.: Deep Learning]*

# Why did it fail?

Conclusions:

- Neural networks work best in vision, speech and language
  - These are easy for humans
  - Very high-dimensional (many inputs)
- Financial problems are mostly hard for humans
- Additionally few inputs were used here
- Therefore the network only learned an (bad) approximation of the naive solution

> *"Most tasks that consist of mapping an input vector to an output vector, and that are easy for a person to do rapidly, can be accomplished via deep learning..."*
>
> *[Goodfellow et al.: Deep Learning]*

# Why did it fail?

Conclusions:

- Neural networks work best in vision, speech and language
  - These are easy for humans
  - Very high-dimensional (many inputs)
- Financial problems are mostly hard for humans
- Additionally few inputs were used here
- Therefore the network only learned an (bad) approximation of the naive solution

> *"Most tasks that consist of mapping an input vector to an output vector, and that are easy for a person to do rapidly, can be accomplished via deep learning..."*
>
> *[Goodfellow et al.: Deep Learning]*

# Why did it fail?

Conclusions:

- Neural networks work best in vision, speech and language
  - These are easy for humans
  - Very high-dimensional (many inputs)
- Financial problems are mostly hard for humans
- Additionally few inputs were used here
- Therefore the network only learned an (bad) approximation of the naive solution

> *"Most tasks that consist of mapping an input vector to an output vector, and that are easy for a person to do rapidly, can be accomplished via deep learning..."*
>
> *[Goodfellow et al.: Deep Learning]*

# Why did it fail?

Conclusions:

- Neural networks work best in vision, speech and language
  - These are easy for humans
  - Very high-dimensional (many inputs)
- Financial problems are mostly hard for humans
- Additionally few inputs were used here
- Therefore the network only learned an (bad) approximation of the naive solution

*"Most tasks that consist of mapping an input vector to an output vector, and that are easy for a person to do rapidly, can be accomplished via deep learning..."*

*[Goodfellow et al.: Deep Learning]*

# Where to go from here?

- Use other machine learning techniques when appropriate
- Neural networks for high-dimensional input data
  - Self-normalizing networks seem promising
- And most important: Learn more math

For code, data and further analysis see:
`https://github.com/leyhline/vix-term-structure`

# Where to go from here?

- Use other machine learning techniques when appropriate
- Neural networks for high-dimensional input data
  - Self-normalizing networks seem promising
- And most important: Learn more math

For code, data and further analysis see:
`https://github.com/leyhline/vix-term-structure`

# Thanks for listening.