Tool For Scholars To Study Machine Learning By Image Classification

SCIENTIFIC WORKSHOP “Big Data Analytics”

PETR GUSEV

SPRING 2017

Problem

High threshold of entry into the subject area of machine learning.

Solution

The application with an interactive graphical interface (stand), teaching in real time basics and demonstrates the possibilities of machine learning.

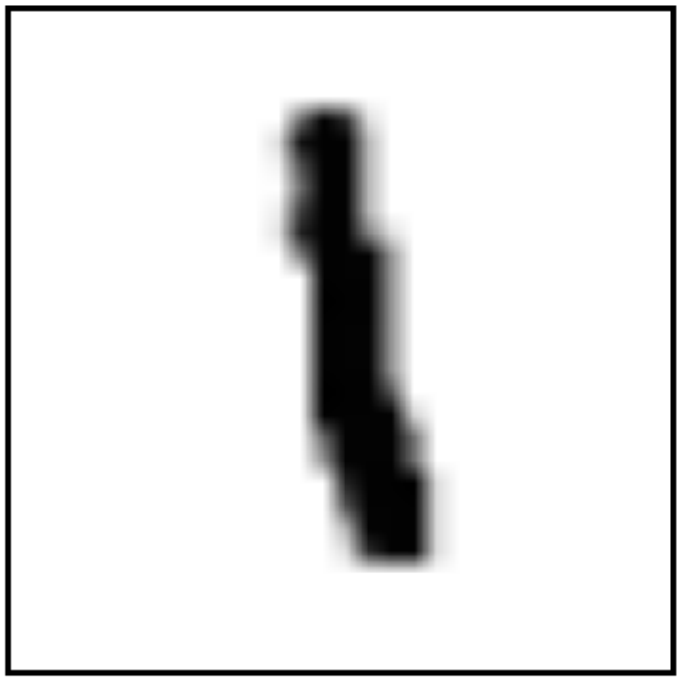
The application should form a basic understanding of what tasks are being accomplished with the help of Machine learning.

Similarly, it should form an idea of the importance of features in the classification problems on the example of image classification.

Goals

1. Choose a **data-set**
2. Train a **model**
3. Look at new **object**
4. **Predict** what digit it represent

MNIST data-set



$\approx$

$$\begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & .6 & .8 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & .7 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & .7 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & .5 & 1 & .4 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & .4 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & .4 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & .7 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & .9 & 1 & .1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & .3 & 1 & .1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}$$

- 28x28 pixel images of hand-written digits
- Every image can be thought of as an array of numbers between 0 and 1 describing how dark each pixel is (in other words, intensity)

MNIST data-set

We have to split data into **3 mutually exclusive** sub-sets:

- **Training** (55.000 to train the algorithm)
- **Testing** (10.000 to test the algorithm)
- **Validation** (5.000 for optimization purposes)

Test data:

- Used **to test** the algorithm, **not to optimize** or **improve** the algorithm.

MNIST data-set

Every MNIST data point has **two parts:**

1. Image of handwritten digit
2. Corresponding **label** (number between 0 and 9) representing the digit drawn in the image

This label will be used **to compare** the predicted digit (by the model) with the **true** digit (given by the data)

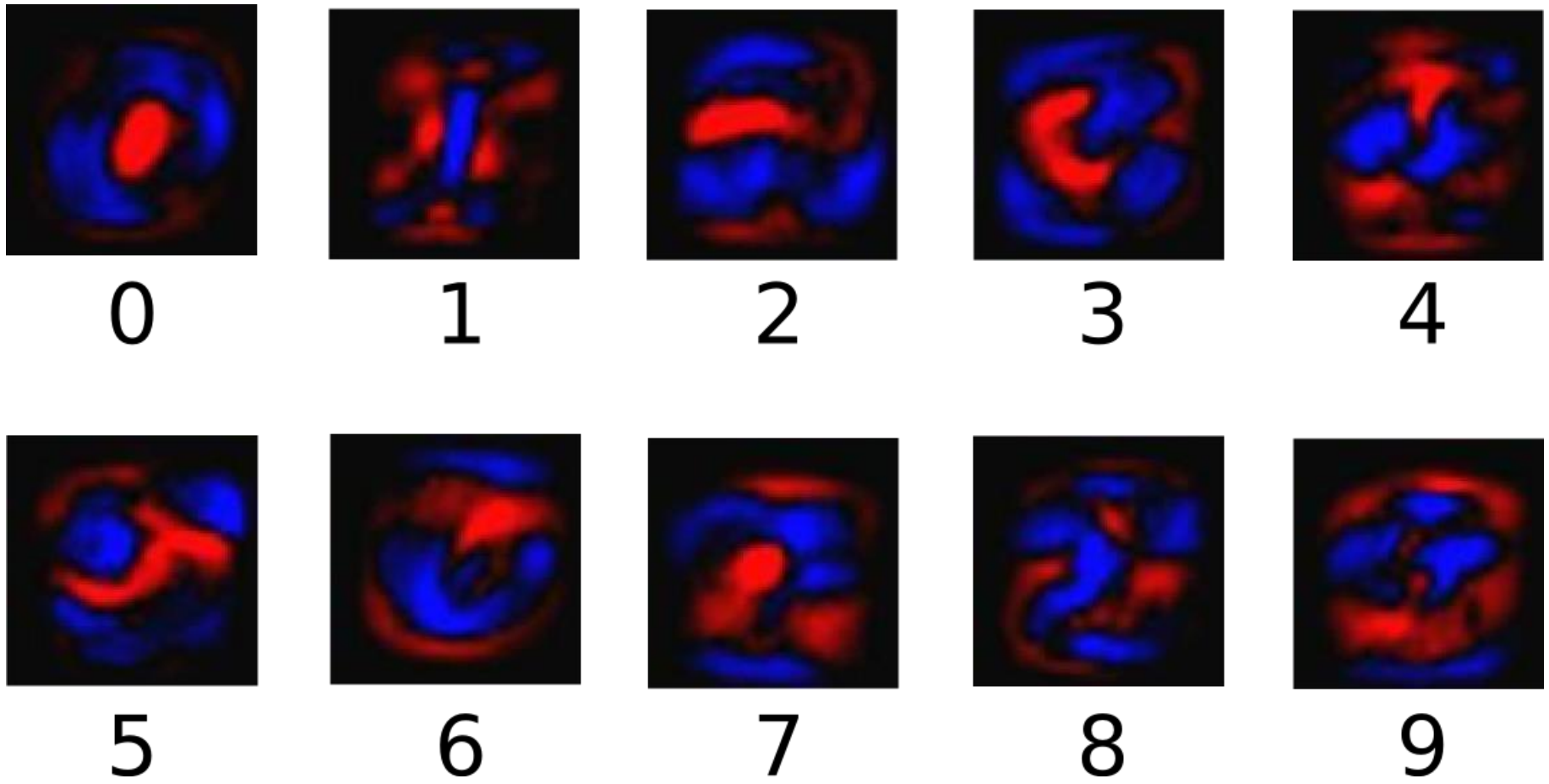
Weights

Goal of the model:

- In order to distinguish the images, we will need a **filter**.
- Better filter => better detection

The following diagram shows an example of the weights (the filter) one model learned for each of these classes.

Weights



Weights

- The weight is a matrix, for each digit, which stores a value for each pixel.

- The weight of pixel is:

  - Negative, if that pixel has a high evidence against the image being in that class
  - Positive, if it is evidence in favor

Evaluation

For example:

The weights used to determine if an image shows a 0-digit:

- Positive reaction to an image of circle
- Negative reaction to images with content in the center of the circle

Weights

To measure the **evidence** of a digit being in one class, one computes a **weighted sum** of the pixel intensities:

- Each image is an **array** of values between 0 and 1
- Each filter has a **weight** for each pixel
- Multiply each pixel's intensity with its corresponding weight and compute the **sum**

Weights

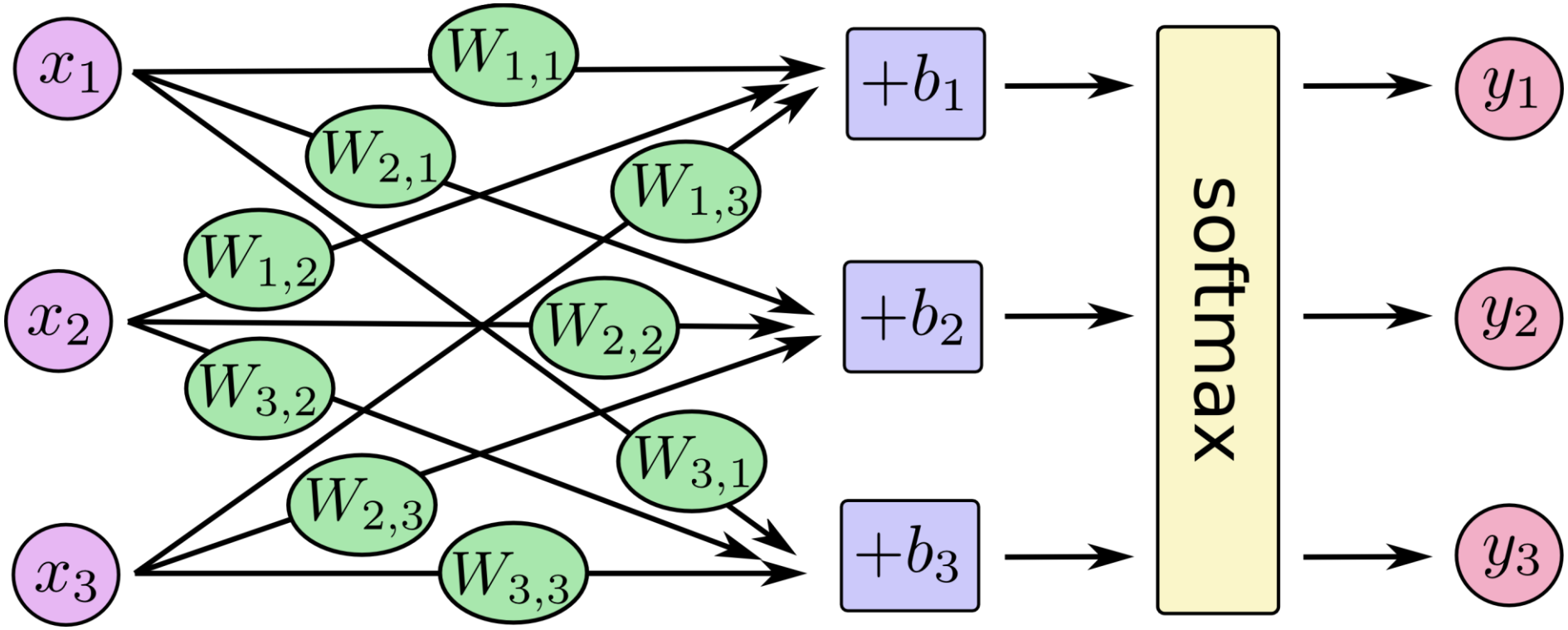
These **evidences** are difficult to interpret, because the numbers may be very small or large, positive or negative (normalization needed).

There are only 10 possible things that a given image can be.

For each image, we want to give the probabilities for being each digit.

Converting evidences into a probability distribution over 10 cases representing probabilities of our input being in each class.

Weights


x_1
x_2
x_3
$W_{1,1}$
$W_{2,1}$
$W_{1,3}$
$W_{1,2}$
$W_{2,2}$
$W_{3,2}$
$W_{3,1}$
$W_{2,3}$
$W_{3,3}$
$+b_1$
$+b_2$
$+b_3$
softmax
y_1
y_2
y_3

Weights

For example:

Our model might look at picture of a 9 and output:

- An **80%** chance of being a **9**
- A **5%** chance of being an **8** (because of top loop)
- A **bit** of probability to all the others because it isn't 100% sure

Weights

Converting the evidence into probabilities using the **softmax** function:

$$y = softmax(evidence)$$
$$softmax(x) = normalize(\exp(x))$$
$$softmax(x)_i = \exp(x_i)/\sum_j \exp(x_j)$$

Now we can take the highest probability for being in a specific class, which will represent the prediction of the handwritten digit.

Weights

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \text{softmax}\left(\begin{bmatrix} W_{1,1} & W_{1,2} & W_{1,3} \\ W_{2,1} & W_{2,2} & W_{2,3} \\ W_{3,1} & W_{3,2} & W_{3,3} \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} \right)$$

The element of the i'th row is an estimate of how likely the input image is to be of the i'th class.

Weights

To remember: The weights and biases represent the filter.

In order to make the model better at classifying the input images, the ML part will try to:

- Change the weights and biases
- Optimize them

But first, we need to know how well the model currently performs.

Training

In our case, training means to get a better filter, so that we can distinguish the handwritten digits better.

We need a definition:

- To describe what it means for the model to be good
- Or to describe what it means for the model to be bad

We call this the **cost** and represents how far off our model is from our desired outcome.

We try to **minimize** that **error/cost**.

Cost

To determine the cost of our model, we are using a function called cross-entropy:

$$H_{y'}(y) = -\sum_i y'_i \log(y_i)$$

Where:

- y is predicted probability distribution
- y' is the true distribution for one handwritten digit

The cross-entropy is measuring how inefficient our predictions are for describing the truth.

Training

The **cross-entropy**:

- Is a **performance measure** used in classification
- Is a function that is always positive
- Equals zero, if the predicted output of the model exactly matches the desired output

The goal of our training is to minimize the cross-entropy so it gets as close to zero as possible.

Training

The preceding procedure gives us a **cost function**, which we want to minimize in order to get a better model.

To minimize the cross-entropy one can use the **gradient descent algorithm:**

- **Shifts** each variable a little bit in the direction that **reduces the cost**
- **Adjusts** the current weights and biases
- Uses a learning rate of 0.5

Consequently, in minimizes the mean of the entropies for the images which we used to train!

Training

In machine learning, the model:

- Is represented by a graph, which stores all the computations
- Can use the **backpropagation algorithm** to efficiently determine how your variables (weights and biases) affect the cost you ask to minimize

Training

Backpropagation is a common method of training artificial neural networks used in combination with an optimization method (gradient descent).

Meaning:

1. Calculate the gradient of a cost function (cross-entropy)
2. Go back in the model
3. Adjust parameters (weights and biases) in order to optimize the cost function at the current state

Training

There are 55.000 images in the training-set. It takes a long time and is expensive to calculate the gradient of the model using all these images.

It is better to do an iteration:

- Use a small batch of images (for example 100) in each iteration of the optimizer. Using new subset every time is cheap and has much of the same benefit as taking all
- Execute the optimizer using those 100 trained samples. Gradually improve the weights and biases of the model
- Repeat this procedure a lot of times. Adapt and generalize the weights of all different kind of handwritten images

Using small batches of random data is called **stochastic training.**

Training

Summary of the training phase:

1. Calculate the **entropy** of the predicted and true digit
2. Calculate the **mean** over **all the entropies** (measure of performance)
3. Minimize the mean (0 means that all out predictions are correct)
4. Use the gradient method to **minimize the cost**

Evaluation

We want to check where we predicted the correct label.

For each image (of the **test data!**):

1. Check if the **predicted** digit is the **same** as the true digit
2. This gives us a **vector** of Booleans
3. Transform the Booleans into numbers
4. Calculate the **average** of these numbers
5. Calculate the **accuracy** of the model

<u>For example:</u>

[True, False, True, True] => **75%**

Evaluation

After no optimization iteration, the accuracy on the test-set is **9.8%**

This is because the model has only been initialized and not optimized at all:

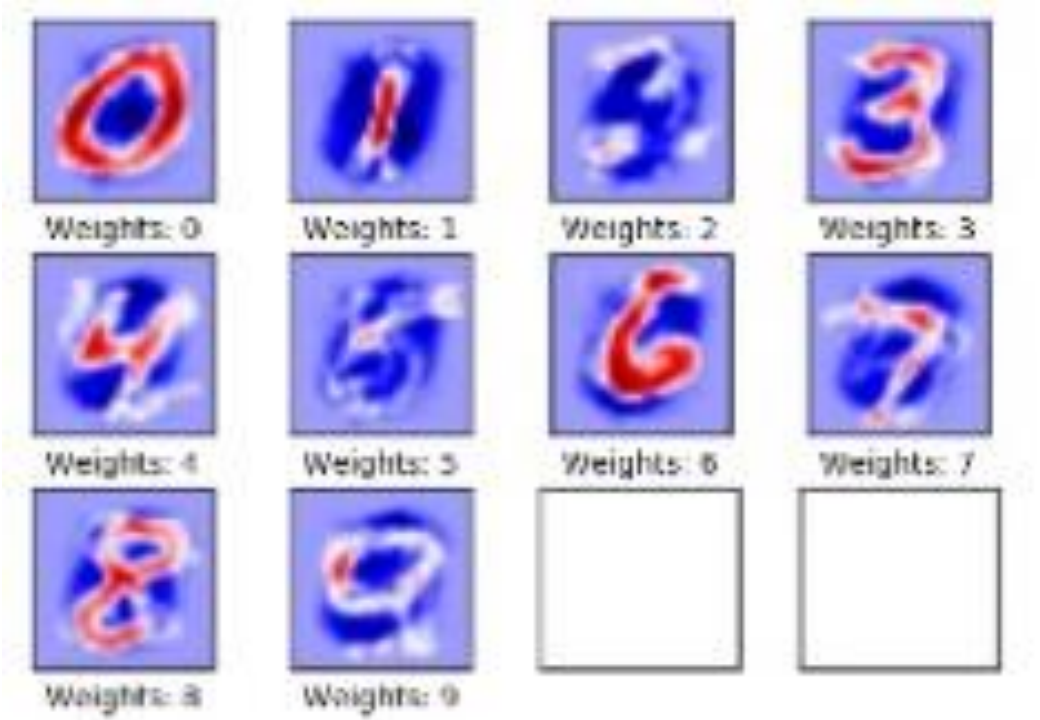
- At initial step, the weights and biases are **0**
- All the evidences will become **zero** and we get **10** zero values
- **Softmax** will output [0.1, ... ,0.1]
- The **maximum** of this list is **0.1**
- All the images are predicted as **0**

Evaluation

The model has increased its accuracy on the test-set to **40%**.

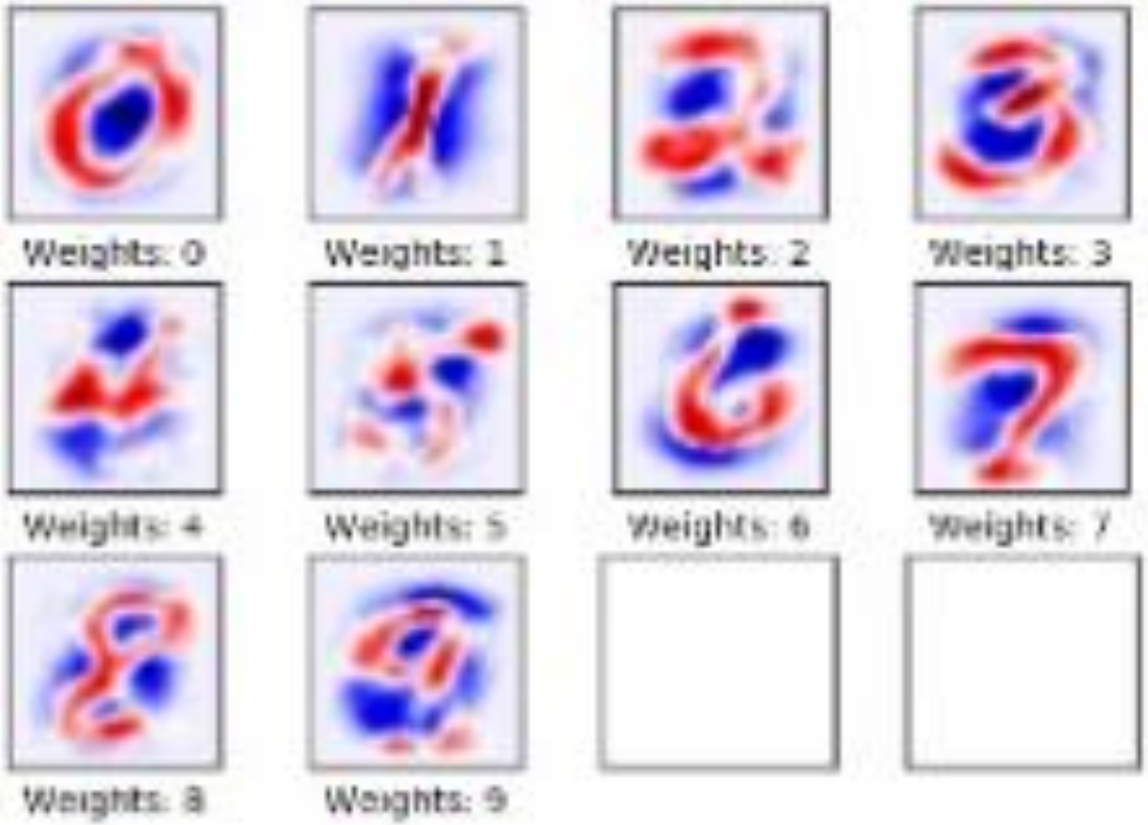
The weights mostly look like digits they're supposed to recognize. This is because only **1** iteration has been preformed.

The weights are <u>only</u> trained on **100** images.



<u>The weights after **1** iteration</u>

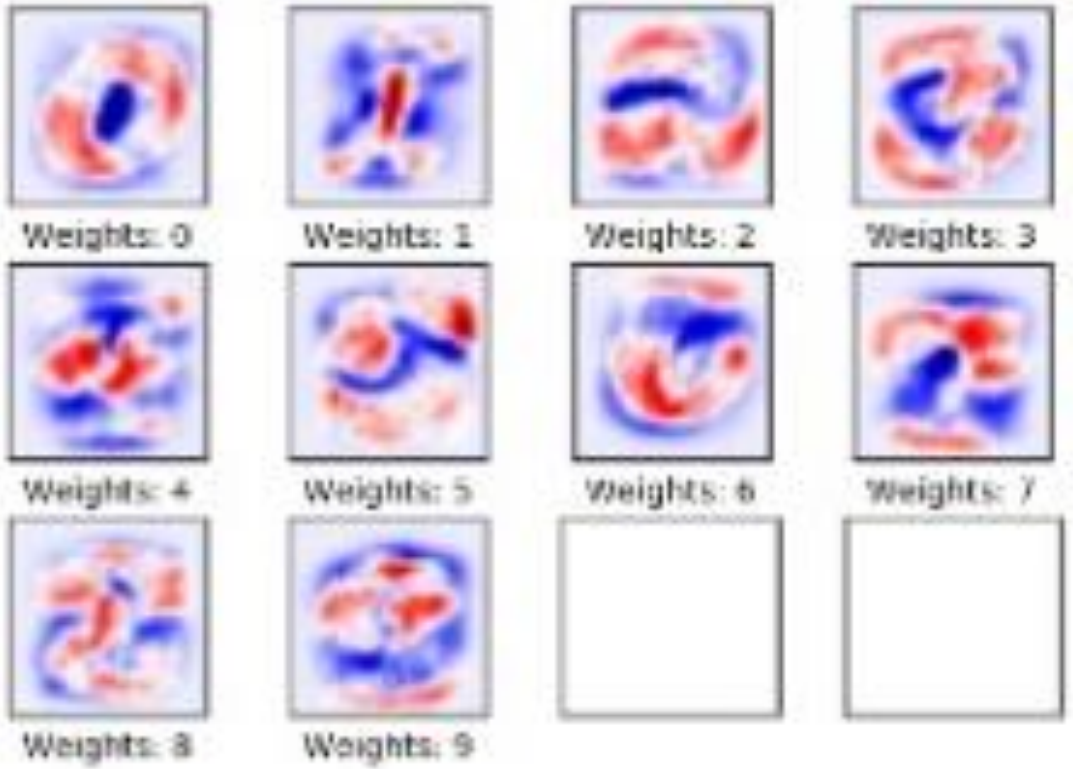
Evaluation



The weights for our model after 10 iterations:

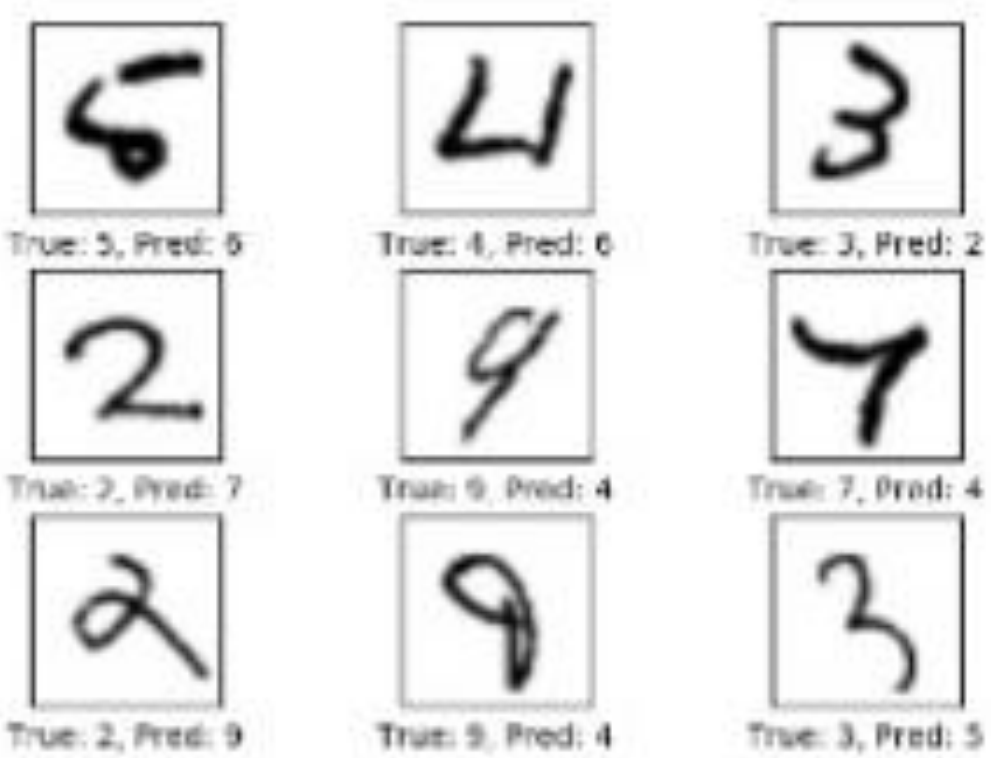
Evaluation

After 1000 iterations,
accuracy – **91%**



What didn’t go well

Simple model cannot reach much better performance and more complex models are therefore needed.



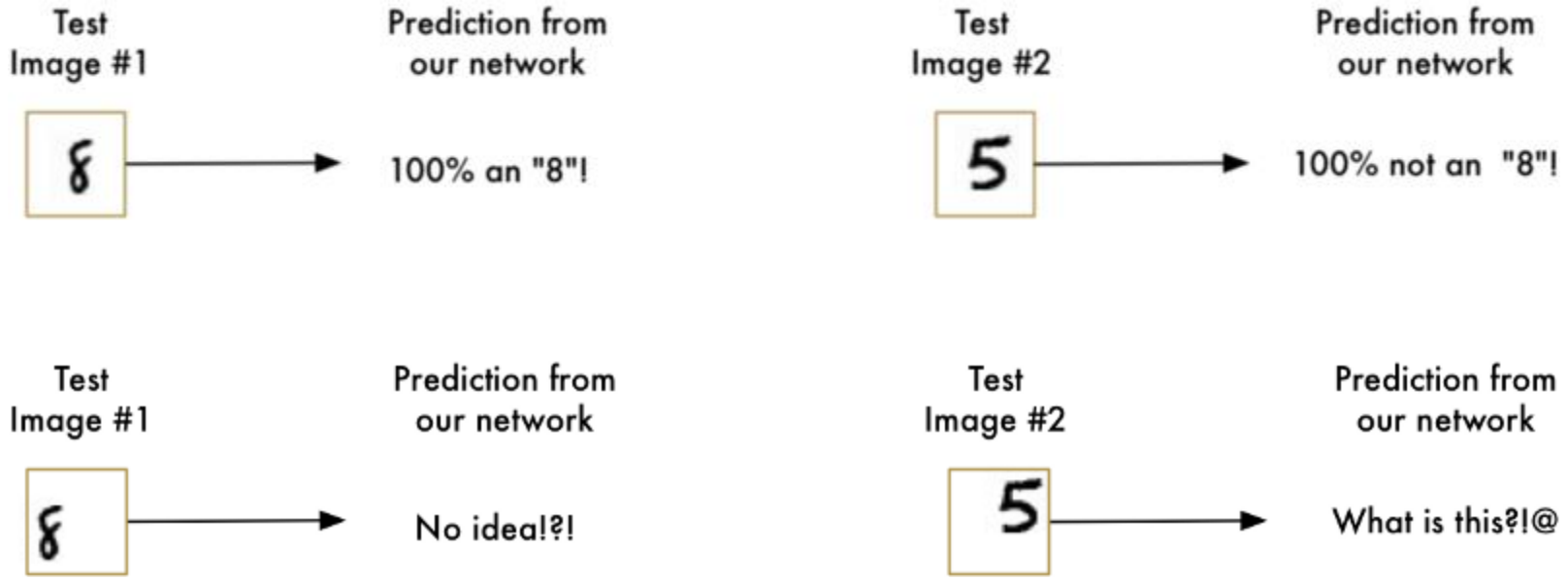
Other options

This simple model had about **92%** classification accuracy for recognizing hand-written digits in the MNIST data-set.

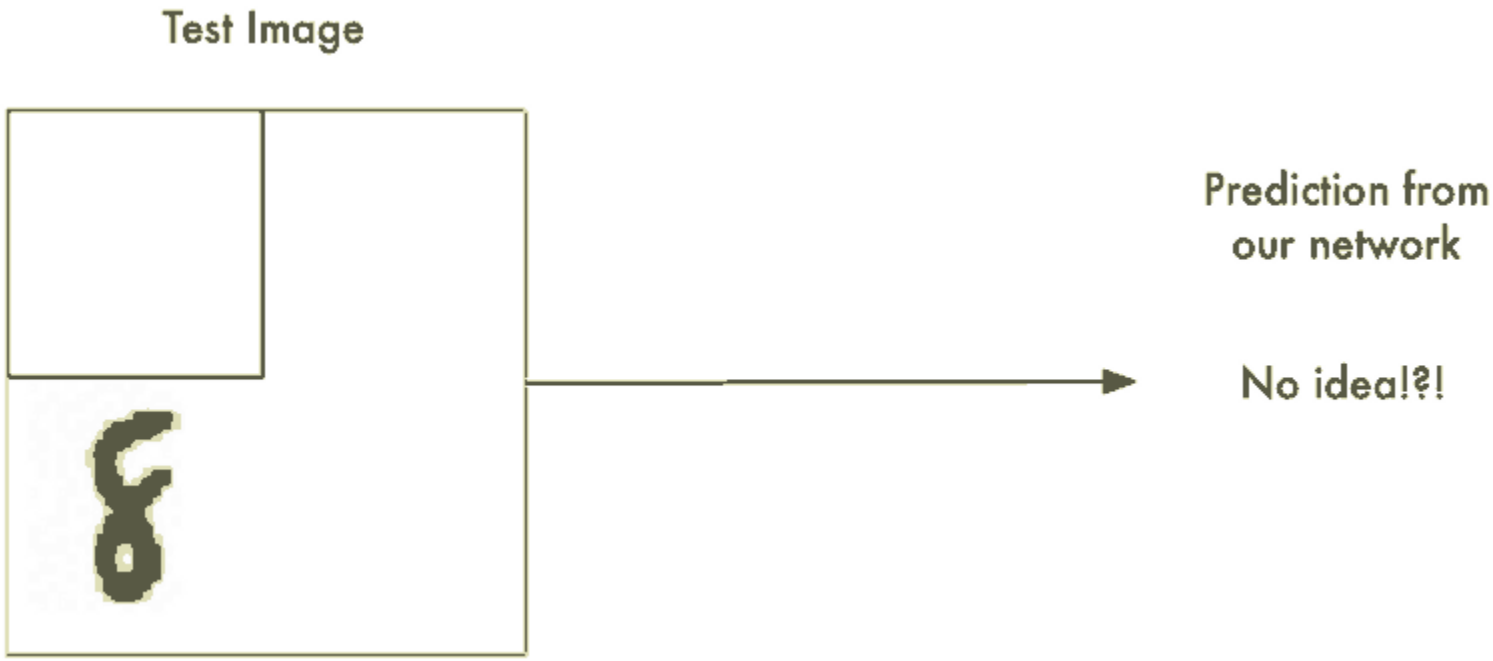
As other researchers say, **CNN** has a classification accuracy of about **99%** or more.

Convolutional Networks work by moving small filters across the input image. This means the filters are re-used for recognizing patterns throughout the entire input image. This makes the CN much more powerful.

Problems of raw data



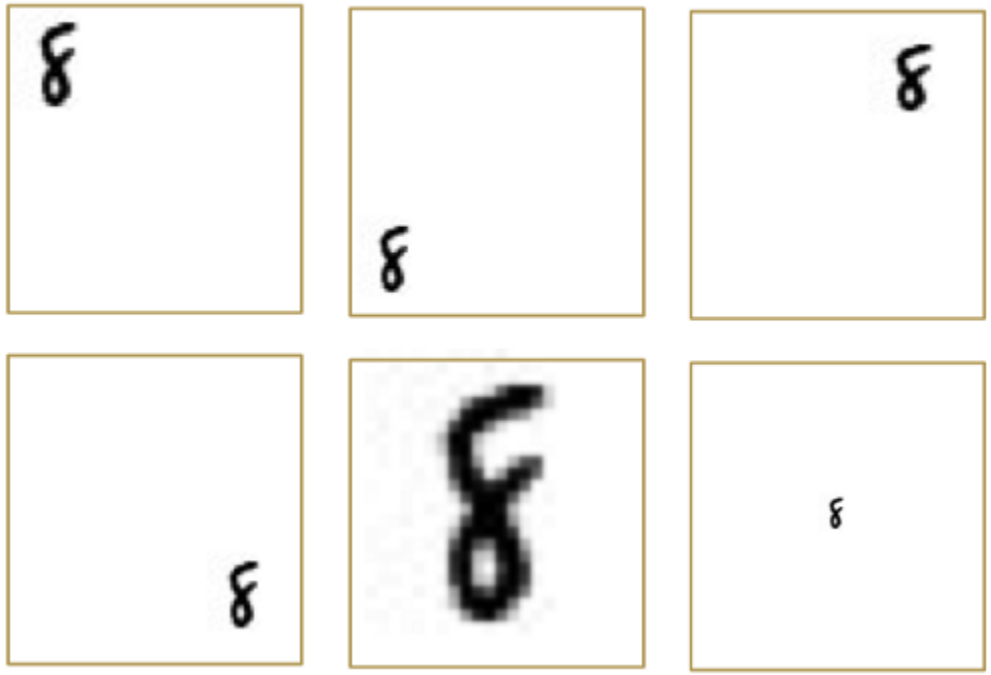
Brute Force Idea #1



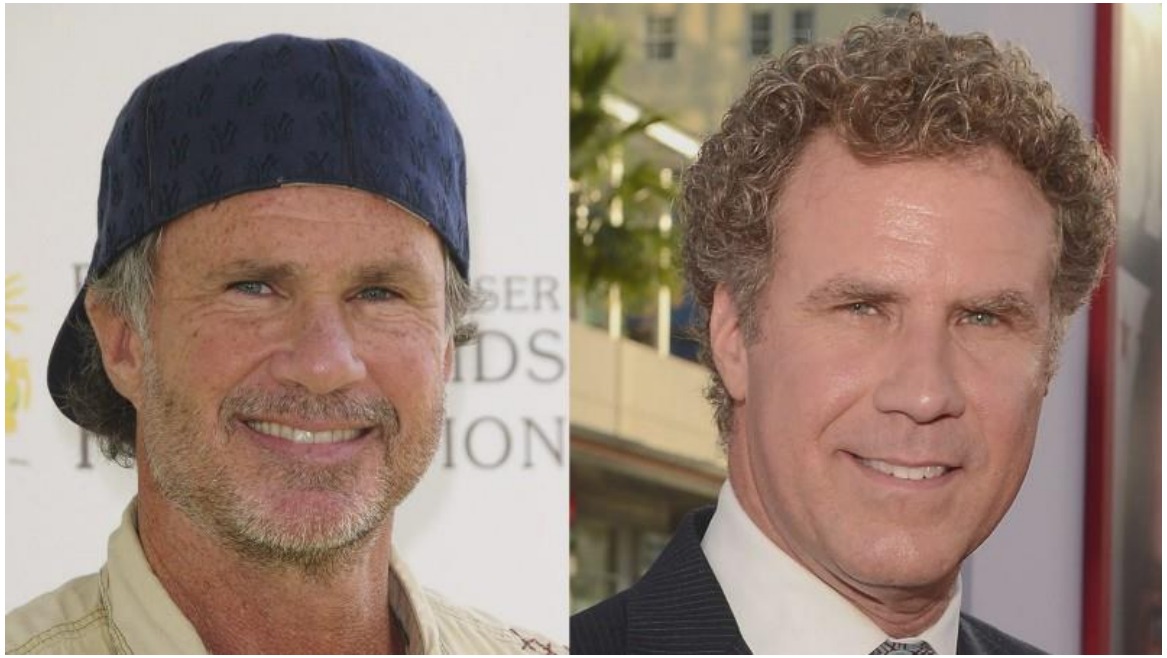
Brute Force Idea #2

Original Training Image

More training images generated by a script



Face recognition techniques



One of these people is Will Farrell.
The other is Chad Smith. I swear they are different people!

HOG

Face detection went mainstream in the early 2000's when Paul Viola and Michael Jones invented a way to detect faces that was fast enough to run on cheap cameras.

However, much more reliable solutions exist now. We're going to use a method invented in 2005 called **Histogram of Oriented Gradients** — or just **HOG** for short.

HOG