Very Deep Learning
Classification, CNN, AlexNet

Marcus Liwicki, Muhammad Zeshan Afzal

University of Kaiserslautern

Insiders Technologies GmbH

University of Fribourg

liwicki@cs.unifr.ch m.afzal@insiders-technologies.de

Disclaimer

- You should know NN, SGD, and CNN by now
- Yet we will do a recap at the beginning to close the gap between well advanced and dull ☺ students
- Video and site suggestions will be online by Thursday, latest – we expect you to know matter
- No need to watch all lectures – these are only suggestions, you should know about the concepts
- Using other's material is a trend in DL (e.g. Stanford lecture). Differences to TU-KL lectures?

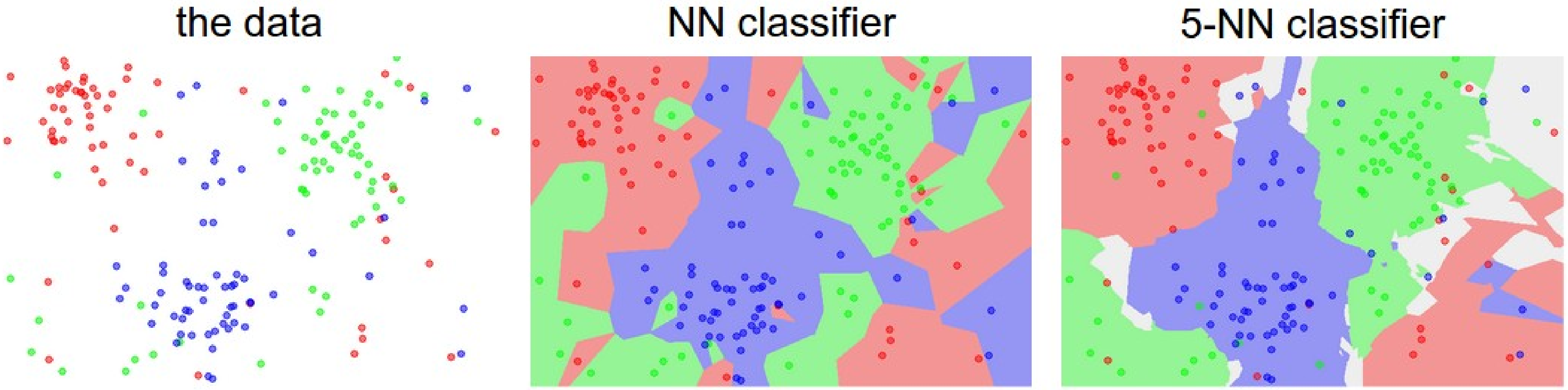
Outline

- Classification (KNN, Linear Classifier)
- Loss Function & Gradient Descent
- CNN
- AlexNet

Note that some of the pictures are taken from http://cs231n.github.io/ (CS231n, Andrej Karpathy)

Classification k-NN

- Distance to every training image (k nearest)



- How good would a 1-NN using Euclidean distance work if test set equals training set?
- How would it be if it would use Manhattan distance?

Linear Classifier

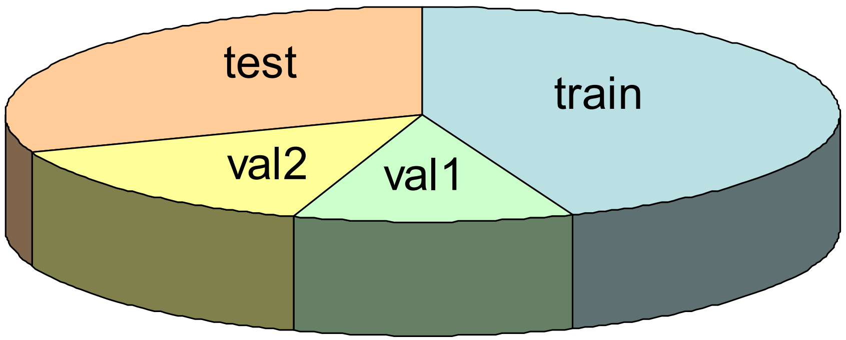
$$f(x_i, W, b) = Wx_i + b$$

- For pixel-images it finds the representative (colour,position) for the class (average image)

- Which dataset would be difficult for a linear classifier?
- http://vision.stanford.edu/teaching/cs231n/linear-classify-demo/

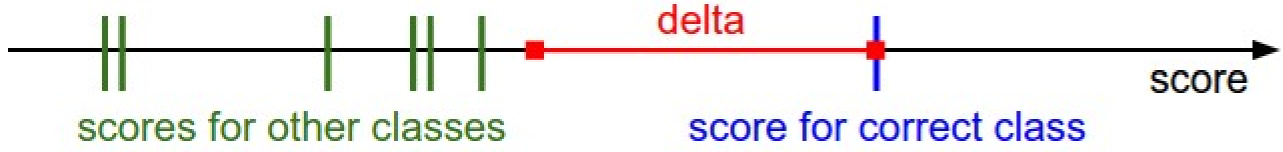
Hyperparameters

- It is very (!!) important to use independent test data
  - Typically 50% for training
  - 20% for validation
  - 30% for testing
- However, might change
  - Depending on number of data available
  - Example:



- Remember: never touch the test set for optimizing anything

Loss function



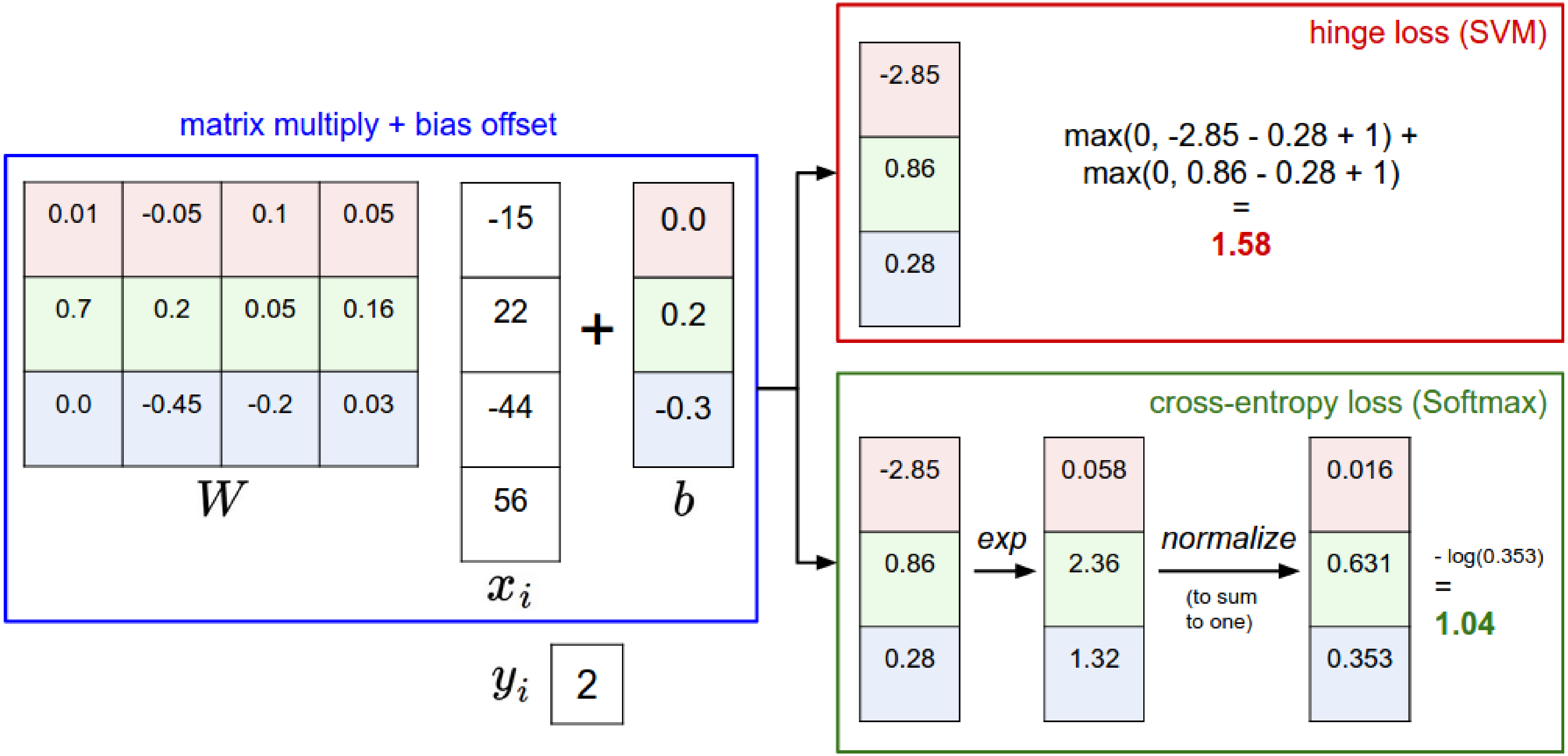
- For one sample: $L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + \Delta)$

  (SVM-loss, hinge)

  $L_i = -\log\left(\frac{e^{s_{y_i}}}{\sum_j e^{s_j}}\right)$

  (Softmax)
- Regularization: $L = overall\ loss + \lambda R(W)$
  - $R(W) = \sum_k \sum_l W_{k,l}^2$
- When would the hinge loss be minimal?
- When will the regularization be minimal?
- Nice effects of Regularization
  - Weights do not grow too much
  - Prefers taking all features into account $[1,0,0] vs [\frac{1}{3}, \frac{1}{3}, \frac{1}{3}]$

Hinge Loss vs. Softmax



- Note: One can think of any other loss function

Gradient Descent

- Computing the gradient (e.g., SVM loss)

$$L_i = \sum_{j \neq y_i} \left[\max(0, w_j^T x_i - w_{y_i}^T x_i + \Delta) \right]$$

$$\nabla_{w_{y_i}} L_i = - \left(\sum_{j \neq y_i} \mathbb{1}(w_j^T x_i - w_{y_i}^T x_i + \Delta > 0) \right) x_i$$

```
# Vanilla Minibatch Gradient Descent

while True:
  data_batch = sample_training_data(data, 256) # sample 256 examples
  weights_grad = evaluate_gradient(loss_fun, data_batch, weights)
  weights += - step_size * weights_grad # perform parameter update
```

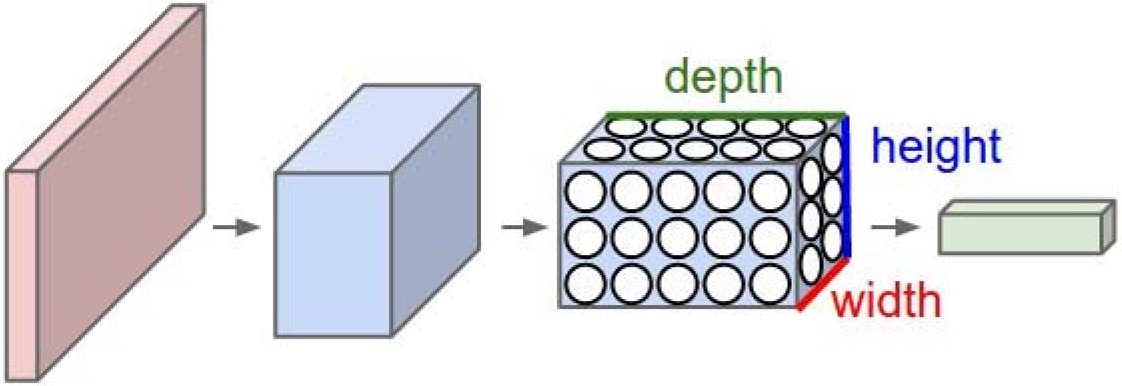
Optimized Gradient Descent

- Simple gradient descent is slow
- Momentum can be used $m_s = \beta m_{s-1} + \nabla$
- Adaptive gradient (per weight), Adam

- Whatever gradient you use – perform a gradient check (analytic verified by numeric gradient)
- Note that often ReLU is assumed, not tanh or sigmoid $f(z) = \max(0, z)$ – Why?

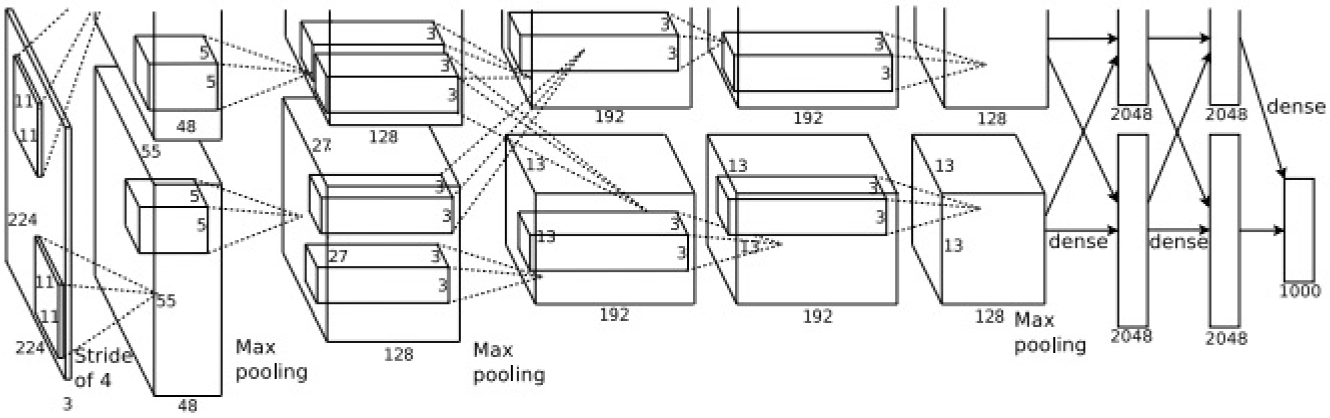
ReLU existed already for a long time (ref. to paper from 1994)

Convolutional Neural Network



AlexNet

- 227x227x3 → 55x55x96



- Why 227 and not 224?

Why Convolution – not Correlation?

- Correlation

$$F \circ I(x,y) = \sum_{j=-N}^{N} \sum_{i=-N}^{N} F(i,j) I(x+i, y+j)$$

- Convolution (same, but mirrored in x and y)

$$F \star I(x,y) = \sum_{j=-N}^{N} \sum_{i=-N}^{N} F(i,j) I(x-i, y-j)$$

- See also: http://www.cs.umd.edu/~djacobs/CMSC426/Convolution.pdf

Useful Tricks to Improve Learning

- Augmenting training data
  - Shift, rotation, (elastic) scale, and combination
  - Color-shift, Simple image filters
  - Random noise
  - Always depends on the application
- Use dropout (What? Caveats? Rate? Where?)
- Multiple classifier combination