Inhaltbasierte Bild- und Videoanalyse Zusammenfassung

Thomas Mohr

Contents

1 Introduction

1.1 Data vs. Information Retrieval

Aspect	Data Retrieval	Information Retrieval
Information	explicit	implicit
Model	deterministic	probabilistic
Results	exact	fuzzy
Query	once	iterative refinement
Fault tolerance	none	yes
Result collection	set	list

1.2 Different Color Spaces

- RGB: Red, Green, Blue
  - Monitor and display technology
  - Easily usable from a programmer's perspective
  - RGB is non-linear
  - If RGB value changes: Hue **and** saturation **and** brightness change
  - Appearance of colors depends on output device

Abbr.	Color	R	G	B
S	Black	0.00	0.00	0.00
R	Red	1.00	0.00	0.00
Y	Yellow	1.00	1.00	0.00
G	Green	0.00	1.00	0.00
C	Cyan	0.00	1.00	1.00
B	Blue	0.00	0.00	1.00
M	Magenta	1.00	0.00	1.00
W	White	1.00	1.00	1.00
K	50% Gray	0.50	0.50	0.50
$\mathbf{R}_{75}$	75% Red	0.75	0.00	0.00
$\mathbf{R}_{50}$	50% Red	0.50	0.00	0.00
$\mathbf{R}_{25}$	25% Red	0.25	0.00	0.00
P	Pink	1.00	0.50	0.50

- YC_bC_r
  - Y: Luminance
  - C_b: Blue difference level
  - C_r: Red difference level
- HSB/HSV/HLS

- Hue
- Saturation
- Brightness (or Value, or Lightness)

1.2.1 Color Space Conversion (RGB2HSV)

$$V \leftarrow max(R, G, B)$$
$$S \leftarrow \begin{cases} \frac{V - min(R,G,B)}{V} & \text{if } V \neq 0 \\ 0 & \text{otherwise} \end{cases}$$
$$H \leftarrow \begin{cases} \frac{60(G-B)}{V-min(R,G,B)} & \text{if } V = R \\ \frac{120+60(B-R)}{V-min(R,G,B)} & \text{if } V = G \\ \frac{240+60(R-G)}{V-min(R,G,B)} & \text{if } V = B \end{cases}$$

1.2.2 Color Spaces for TV (Television)

- YUV/YC_bC_r: Color space(s) for standardized recording, storage, transmission and display of color TV video (frames)
- Color space is seperated in (analog TV)
  - 1 luminance component Y
  - 2 color components U, V
- $YUV \rightarrow$ PAL/NTSC analog TV
- $YC_bC_r \rightarrow$ digital TV

1.2.3 YUV (Component) Color Space

- Color space conversion: RGB $\rightarrow$ YUV

$$Y = 0.299 * R + 0.587 * G + 0.114 * B$$
$$U = 0.492 * (B - Y)$$
$$V = 0.877 * (R - Y)$$

- In matrix form (linear mapping)
  - RGB $\rightarrow$ YUV

$$\begin{pmatrix} Y \\ U \\ V \end{pmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ -0.147 & -0.289 & 0.436 \\ 0.615 & -0.515 & -0.100 \end{bmatrix} * \begin{pmatrix} R \\ G \\ B \end{pmatrix}$$

- YUV → RGB

$$\begin{pmatrix} R \\ G \\ B \end{pmatrix} = \begin{bmatrix} 1.000 & 0.000 & 1.140 \\ 1.000 & -0.395 & -0.581 \\ 1.000 & 2.032 & 0.000 \end{bmatrix} * \begin{pmatrix} Y \\ U \\ V \end{pmatrix}$$

1.2.4 YC_bC_r Component Color Space

- Variant (extension) of YUV
- International standard for digital TV
- Common for JPEG (JFIF, EXIF)
- RGB → YC_bC_r

$$\begin{pmatrix} Y \\ C_b \\ C_r \end{pmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ -0.169 & -0.331 & 0.500 \\ 0.500 & -0.419 & -0.081 \end{bmatrix} * \begin{pmatrix} R \\ G \\ B \end{pmatrix}$$

- YC_bC_r → RGB

$$\begin{pmatrix} R \\ G \\ B \end{pmatrix} = \begin{bmatrix} 1.000 & 0.000 & 1.403 \\ 1.000 & -0.334 & -0.714 \\ 1.000 & 1.173 & 0.000 \end{bmatrix} * \begin{pmatrix} Y \\ C_b \\ C_r \end{pmatrix}$$

1.3 Evaluation of Multimedia Retrieval Systems

- Speed of answering queries
- Judge relevance of retrieved document
- Popular measures
  - Recall
  - Precision
  - Average precision

1.3.1 Performance Measures

- Recall
  ”Probability” that a relevant document is found (recognition rate)
- Precision
  ”Probability” that a found document is relevant
- Average precision
  Averaged precision across all ranks with a relevant document for a result list

1.3.1.1 Recall & Precision

- Given are
  - R
    Number of relevant objects or events in test data set
  - C
    Number of correctly detected relevant objetcs/events
  - D
    Number of detected objects/events, including the irrelevant ones (”false alarms”)
- Therefore

$$\begin{aligned}\text{recall} &= \frac{C}{R} \\ &= \frac{\text{true positives}}{\text{true positives} + \text{false positives}} \\ \text{precision} &= \frac{C}{D} \\ &= \frac{\text{true positives}}{\text{true positives} + \text{false negatives}}\end{aligned}$$

- F1 measure is a combination of both, harmonic mean

$$f1 = \frac{2 * \text{recall} * \text{precision}}{\text{recall} + \text{precision}}$$

1.3.1.2 Average Precision

- A is the result set of returned documents
- $L_k = \{I_1, \ldots, I_k\}$ is the subset of the k most similiar responses
- R ist the set of returned relevant documents in the top-k response
- R_{all} is the set of all relevant documents in the entire data collection
- $\psi(I_i)$ is a function that is
  - 1 if $I_i \in R$
  - 0 if $I_i \notin R$

$$\begin{aligned}avg\_precision &= \frac{1}{|R_{all}|} \sum_{k=1}^{|A|} \frac{\left|R \cap L^k\right|}{k} \psi(l_k) \\ avg\_precision* &= \frac{1}{|R_{result\_set}|} \sum_{k=1}^{|A|} \frac{\left|R \cap L^k\right|}{k} \psi(l_k)\end{aligned}$$

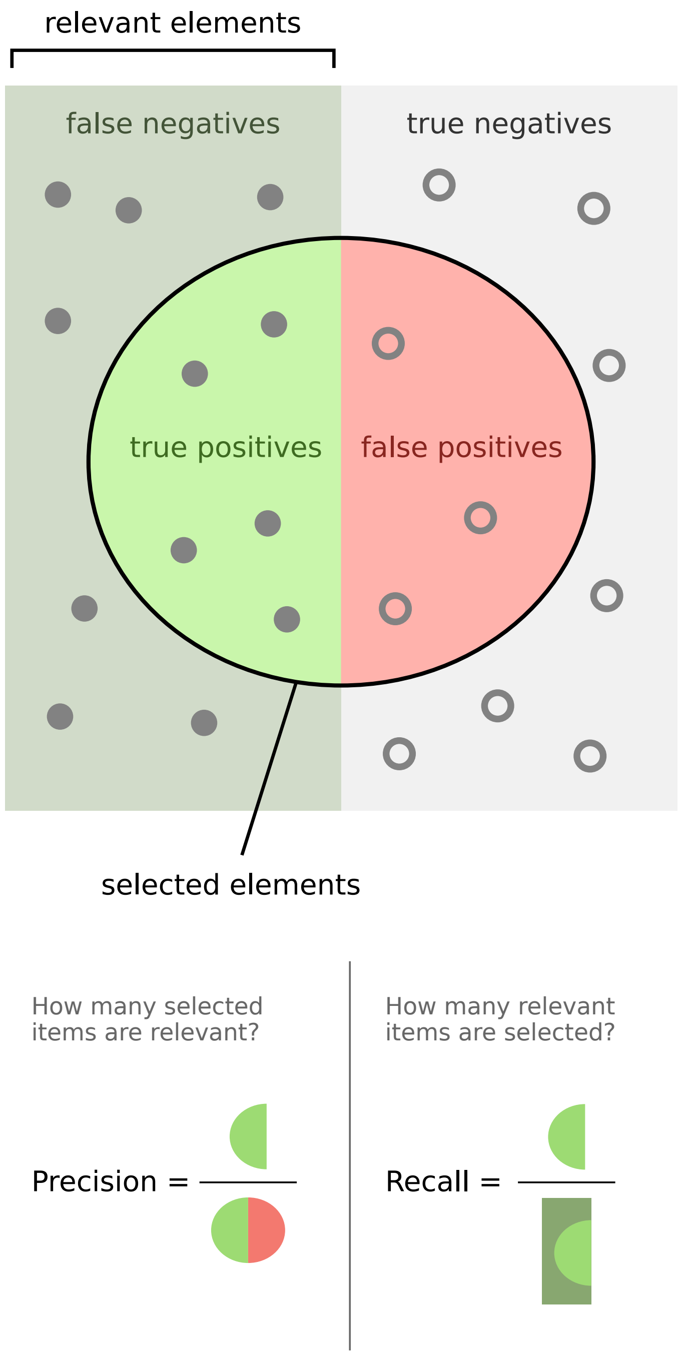


Figure 1: Precision/Recall

1. **relevant document** (precision@1:100%)
2. irrelevant document
3. irrelevant document
4. **relevant document** (precision@4:50%)
5. irrelevant document
6. **relevant document** (precision@6:50%)
7. irrelevant document
8. irrelevant document

$$\text{average precision} : \frac{200}{3} = 67\%$$
$$\text{precision overall} : 37,5\%$$

1.3.2 Recall-Precision Curve

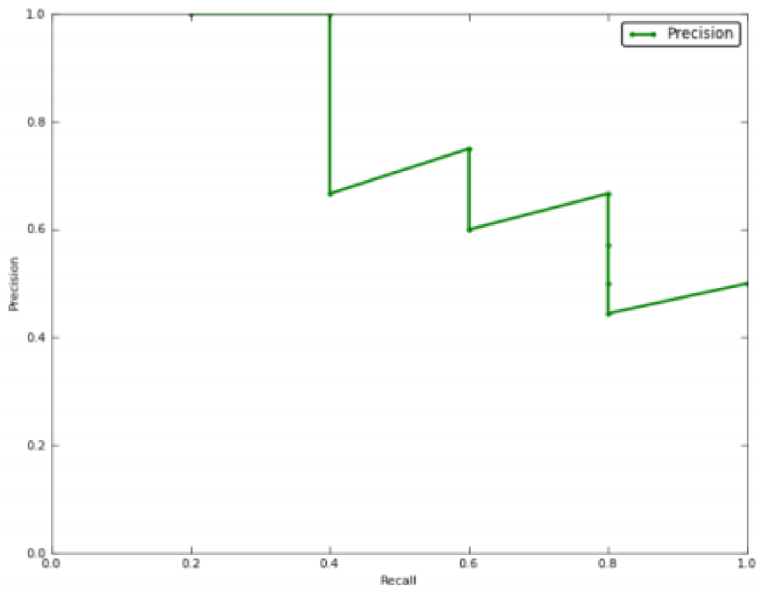


Figure 2: Average precision approximates the area under the curve

2 Image Classification

2.1 Challenges

- Semantic Gap
  - An image is just a big grid of numbers between [0,255]
- Viewpoint Variation
  - All pixels change when the camera moves
- Illumination (Lighting)
- Deformation (Shape)
- Occlusion
- Background Clutter (Similiar background pattern)
- Intraclass Variation

2.2 Machine Learning Algorithms

- Supervised Learning (”right answers” given)
  - Classification
    discrete valued output
  - Regression
    continuous valued output
- Unsupervised Learning
  - Find hidden structure from unlabeled data
  - No ground truth given
  - Clustering
  - Dimensionality reduction

2.2.1 Nearest Neighbor

- Nearest Neighbor
  - Memorize all data and labels
  - Predict the label of most similiar training image
- K-Nearest Neighbor
  - Instead of copying label from nearest neighbor, take **majority vote** from K closest points
  - Never used on images

        - Very slow at test time
        - Distance metrics on pixels are not informative. The following share the same L2 distance:
            - Original
            - Boxed
            - Shifted
            - Tinted

2.2.2 Distance metrics

- L1 (Manhattan) distance

$$d_1(I_1, I_2) = \sum_p |I_1^p - I_2^p|$$

$$\left| \overset{\text{test image}}{\begin{array}{|c|c|c|c|}\hline 56 & 32 & 10 & 18 \\ \hline 90 & 23 & 128 & 133 \\ \hline 24 & 26 & 178 & 200 \\ \hline 2 & 0 & 255 & 220 \\ \hline \end{array}} - \overset{\text{training image}}{\begin{array}{|c|c|c|c|}\hline 10 & 20 & 24 & 17 \\ \hline 8 & 10 & 89 & 100 \\ \hline 12 & 16 & 178 & 170 \\ \hline 4 & 32 & 233 & 112 \\ \hline \end{array}} \right| = \overset{\text{pixel-wise absolute value differences}}{\begin{array}{|c|c|c|c|}\hline 46 & 12 & 14 & 1 \\ \hline 82 & 13 & 39 & 33 \\ \hline 12 & 10 & 0 & 30 \\ \hline 2 & 32 & 22 & 108 \\ \hline \end{array}} \xrightarrow{\text{add}} 456$$

- L2 (Euclidean) distance

$$d_2(I_1, I_2) = \sqrt{\sum_p (I_1^p - I_2^p)^2}$$

2.2.3 Hyperparameters

- What is the best value of **k** to use?
    - Split data into train, validation and test
    - Cross-Validation
        - Split data into **folds**, try each fold as validation and average the results
        - Useful for small datasets, but not used too frequently in deep learning
- What is the best **distance** to use?
    - L1 distance
    - L2 distance
- These are **hyperparameters**: Choices about the algorithm that we set rather than learn

2.3 Linear Classification

- Score function

$$f(x, W) = Wx + b$$

  - Input: Image of a cat with 4 pixels
  - Output: Score for each class

input image(x)

56	231
24	2

$$\begin{array}{c} W \\ \begin{array}{|c|c|c|c|}\hline 0{,}2 & -0{,}5 & 0{,}1 & 2{,}0 \\ \hline 1{,}5 & 1{,}3 & 2{,}1 & 0{,}0 \\ \hline 0 & 0{,}25 & 0{,}2 & -0{,}3 \\ \hline \end{array} \end{array} \times \begin{array}{c} x \\ \begin{array}{|c|}\hline 56 \\ \hline 231 \\ \hline 21 \\ \hline 2 \\ \hline \end{array} \end{array} + \begin{array}{c} b \\ \begin{array}{|c|}\hline 1{,}1 \\ \hline 3{,}2 \\ \hline -1{,}2 \\ \hline \end{array} \end{array} = \begin{array}{|c|}\hline -96{,}8 \\ \hline 437{,}9 \\ \hline 61{,}95 \\ \hline \end{array} \begin{array}{c} \textbf{cat} \\ \text{dog} \\ \text{ship} \end{array}$$

- Loss function

$$\frac{1}{N} \sum_i L_i(f(x_i, W), y_i)$$

  - Tells how good the current classifier ist
  - Given a dataset of examples $\{(x_i, y_i)\}_{i=1}^N$
    - x_i is an image
    - y_i is an (integer) label

- Multiclass SVM Loss
  - "Hinge loss"

$$\begin{aligned} L_i &= \sum_{j \neq y_i} \begin{cases} 0 & \text{if } s_{y_i} \leq s_j + 1 \\ s_j - s_{y_i} + 1 & \text{otherwise} \end{cases} \\ &= \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1) \end{aligned}$$

2.4 Regularization

$$L(W) = \underbrace{\frac{1}{N} \sum_{i=1}^{N} L_i(f(x_i, W), y_i)}_{\substack{\textbf{Data loss: } \text{Model predictions} \\ \text{should match training data}}} + \underbrace{\lambda R(W)}_{\substack{\textbf{Regularization}\text{: Model} \\ \text{should be "simple", so it} \\ \text{works on test data}}}$$

- **L2 regularization** $R(W) = \sum_k \sum_l W_{k,l}^2$
- L1 regularization $R(W) = \sum_k \sum_l |W_{k,l}|$
- Elastic net (L1+L2) $R(W) = \sum_k \sum_l \beta W_{k,l}^2 + |W_{k,l}|$
- Max norm regularization
- Dropout
- Batch normalization

3 Neural Networks Intro

3.1 Softmax Classifier

- **Given**: score function

$$s = f(x_i, W)$$

- **Wanted**: probability distribution over K different possible outcomes
- **Softmax function** "squashes" a K-dimensional vector of arbitrary real values to a K-dimensional vector of real values in the range [0,1] that add up to 1

$$P(Y = k \mid X = x_i) = \boxed{\frac{e^s_k}{\sum_j e^s_j}}$$

3.1.1 Loss function for Softmax Classifier

Want to maximize the log likelihood
$\rightarrow$ minimize the negative log likelihood of the correct class:

$$\begin{aligned} L_i &= -\log P(Y = y_i \mid X = x_i) \\ &= -\log\left(\frac{e^{s_{y_i}}}{\sum_j e^{s_j}}\right) \end{aligned}$$

Figure 3: Log loss formula

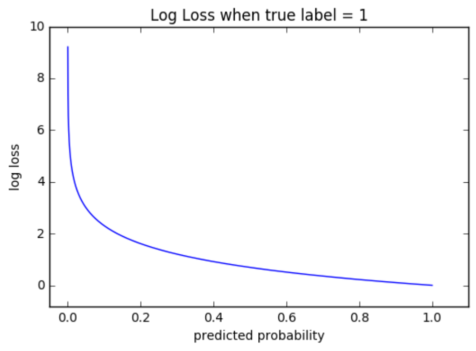


Figure 4: Log loss plot

3.1.2 Softmax Classifier Example

$$\begin{array}{l} \text{cat} \\ \text{car} \\ \text{frog} \end{array} \boxed{\begin{array}{c} \mathbf{3.2} \\ 5.1 \\ -1.7 \end{array}} \overset{\exp}{\rightarrow} \boxed{\begin{array}{c} \mathbf{24.5} \\ 164.0 \\ 0.18 \end{array}} \overset{\text{normalize}}{\rightarrow} \boxed{\begin{array}{c} \mathbf{0.13} \\ 0.87 \\ 0.00 \end{array}} \rightarrow L_i = -\log(0.13) = 0.89$$

unnormalized log probabilities $\rightarrow$ unnormalized probabilities $\rightarrow$ probabilities

3.2 Gradients

3.2.1 Following the gradient

- In 1-dimension, the derivative of a function:

$$\frac{df(x)}{dx} = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

- In multiple dimensions, the **fradient** is the vector of (partial derivatives) along each dimension
- The slope in any direction is the **dot product** of the direction with the gradient
- The direction of the steepest descent is the **negative gradient**

3.2.2 Computing the gradient

- **Numerical** gradient
  - Easy way
  - Slow
  - Approximate
- **Analytic** gradient
  - Fast
  - Exact
  - More error-prone

3.2.3 Gradient descent

- Procedure of repeatedly evaluating the gradient and then performing a parameter update
- Variants
  - Vanilla/Batch gradient descent

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

- Mini-batch gradient descent

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

- Stochastic gradient descent
  - Mini-batch with only a single example (at a time)

- Follow the gradient until we're happy with the results
- Core of all Neural Network libraries

3.2.3.1 Effect of step size

- The gradient only tells us the direction
- How far should we step along this direction?
- **Step size (learning rate)**:
  Most important hyperparameter setting in training neural networks
  - Small step size → make consistent, but very small progress
  - Large step size → attempt to descend faster, may fail

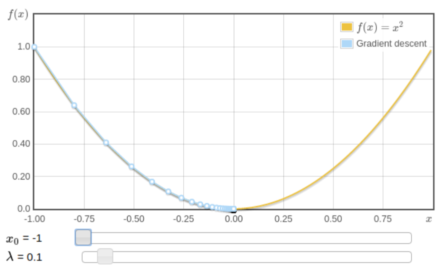


Figure 5: Small step size

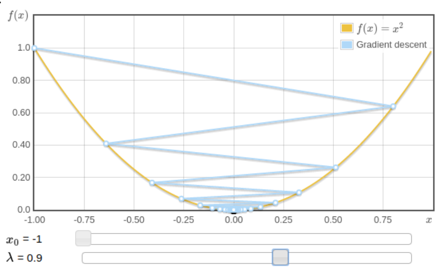


Figure 6: Large step size

3.3 Backpropagation

- Resursive application of the chain rule along a computational graph to compute the gradients of all inputs/parameters/intermediates
- Calculate the gradient of the loss function with respect to the weights/inputs
- Phases
  - Propagation
    - Forward propagation of a training pattern's input through the neural network in order to generate the network's output value(s).
    - Backward propagation of the propagation's output activations through the neural network using the training pattern target in order to generate the deltas (the difference between the targeted and actual output values) of all output and hidden neurons.
  - Weight update
    - For each weight the following steps must be followed:
      - The weight's output delta and input activation are multiplied to find the gradient of the weight.
      - A ratio (percentage) of the weight's gradient is subtracted from the weight.
    - This ratio (percentage) influences the speed and quality of learning; it is called the learning rate. The greater the ratio, the faster the neuron trains, but the lower the ratio, the more accurate the training is. The sign of the gradient of a weight indicates whether the error varies directly with, or inversevely to, the weight. Therefore, the weight must be updated in the opposite direction, "descending" the gradient.

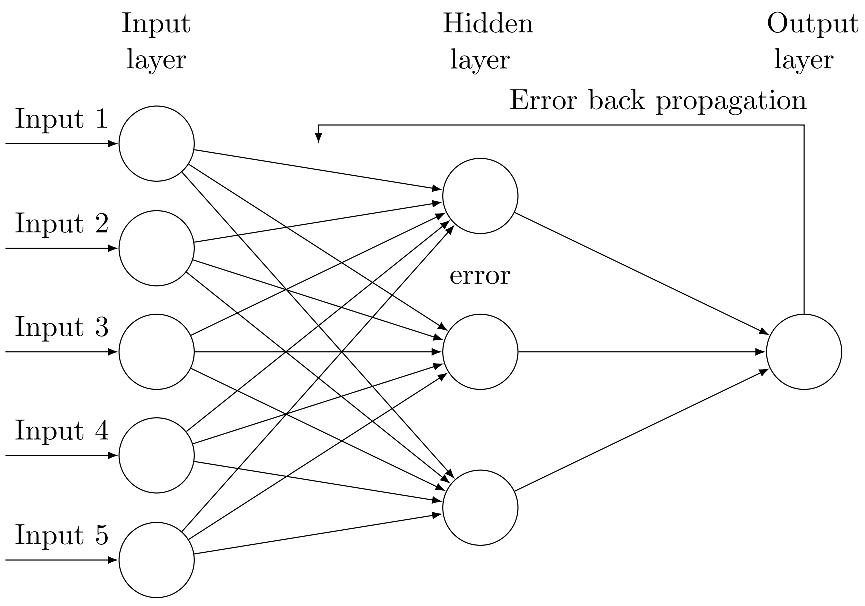
3.3.1 Gates

- Add
  Gradient distributor (both operands gradients are set to backpropagated gradient)
- Max
  Gradient router (maximum input of operands receives gradient; other operand is set to 0.00)
- Mul
  Gradient switcher (gradient multiplied by input from other operand)

3.3.2 Vectorized operations

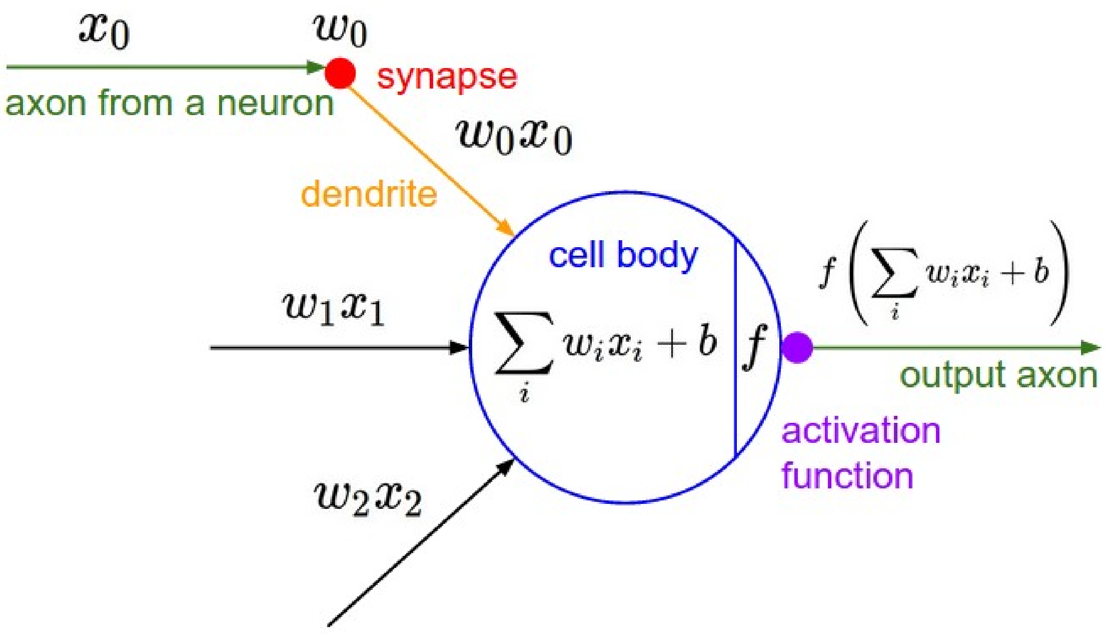
- Jacobi matrix
  TODO

Figure 7: Backpropagation

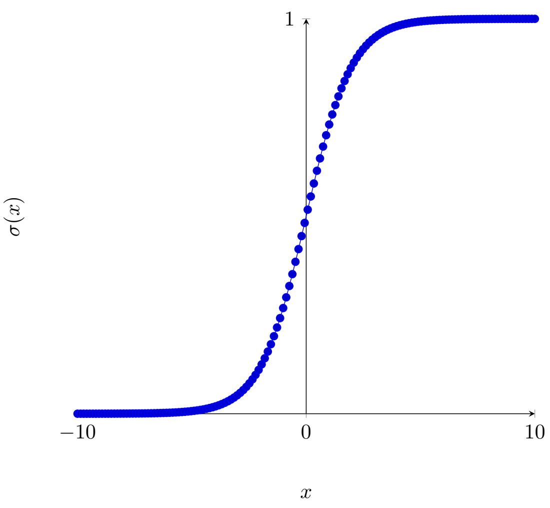


3.4 Neurons

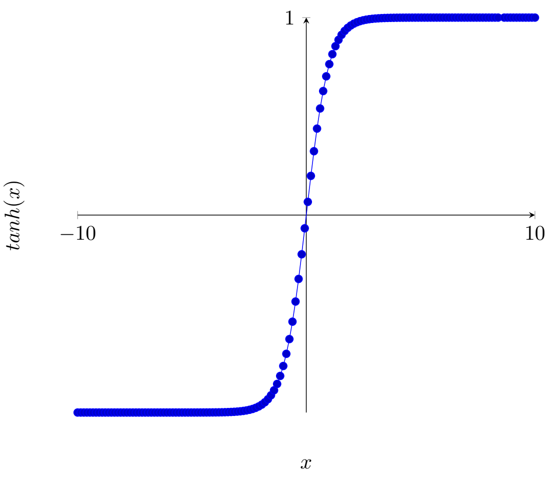
Figure 8: Neuron



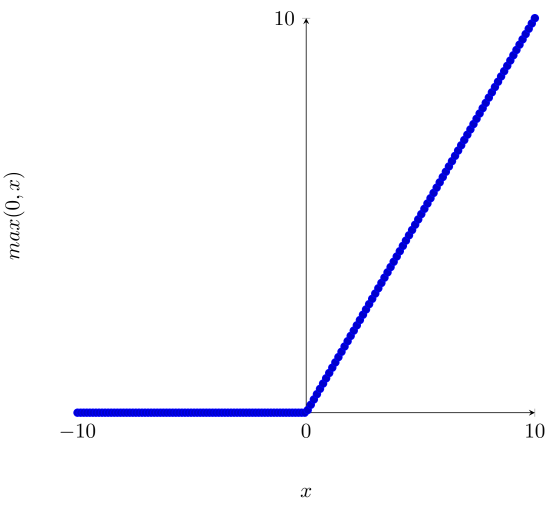
3.5 Activation functions



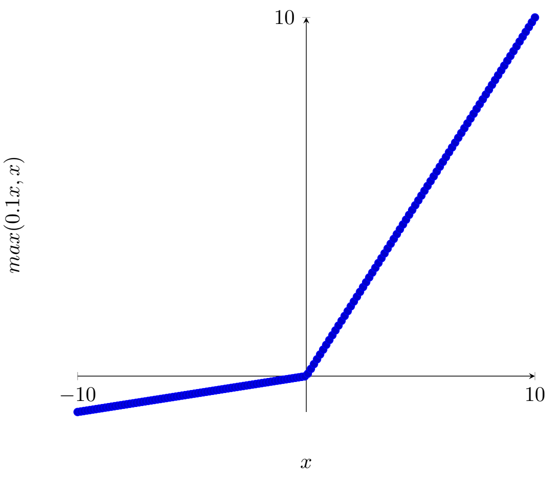
a) Sigmoid: $\sigma(x) = \frac{1}{1+\exp^{-x}}$



b) $tanh(x)$



c) ReLU: $max(0, x)$



d) Leaky ReLU: $max(0.1x, x)$

4 Image Features

4.1 Histograms

- Histograms provide simple image statistics:
  - Number of pixels, min and max grey level
  - Average and standard deviation
  - Indicator for image quality
- Spatial information is lost
- Different images can have the same (or a similiar) histogram

4.1.1 Histograms of Color Images

- Luminance histogram
  - Distribution of grey level intensities
  - (can be) computed based on color information
- Histograms of color components
  - Three histograms: Distribution of intensities in each color channel
  - Alternative: One combined color histogram

4.2 (Color) Moments of Order 1-3

- Mean of (gray level) image (1. order moment)

$$\begin{aligned} m(I) &= \frac{1}{\sum_{k=0}^{255} h_k} \sum_{k=0}^{255} k * h_k \\ &= \frac{1}{W * H} \sum_{y=0}^{H-1} \sum_{x=0}^{W-1} i(x, y) \end{aligned}$$

- Variance (2. order moment, mean squared error)

$$\sigma^2(I) = \frac{1}{W * H} \sum_{k=0}^{255} (k - m)^2 * h_k$$

- Skewness (3. order moment, degree of asymmetry)

$$s^3(I) = \frac{1}{W * H} \sum_{k=0}^{255} \frac{(k - m)^3 * h_k}{\sigma}$$

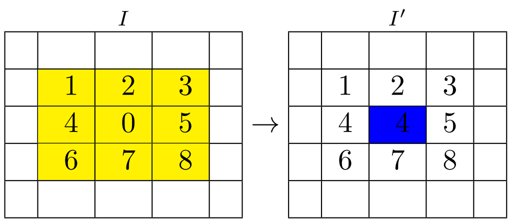
- Where
  - h_k is entry in bin k
  - W is the width of the image
  - H is the height of the image

4.3 Color Correlogram

- Given a color c, e.g., (r, g, b) $\rightarrow$ what is the probability that a pixel of color c' appears in distance d?
- A Color Correlogram is a histogram about frequencies that c' appears in the neighborhood of pixels with color c

4.4 Smooth/Blur

- Can not be realized via point operators
- Brightness and contrast do not change
- Geometry of image remains unchanged
- Therefore each pixel is computed by averaging it's sorrounding neighbors



$$I'(u, v) = \frac{p_0 + p_1 + p_2 + p_3 + p_4 + p_5 + p_6 + p_7 + p_8}{9}$$

4.5 Sobel Edge Operator

TODO

4.6 Bag of Visual Words

1. Build codebook
   - Extract random patches
   - Cluster patches to form "codebook" of "visual words"
2. Encode images

4.7 Scale-Invariant Feature Tranform (SIFT)

TODO

4.8 Histogram of Oriented Gradients (HoG)

TODO

4.9 Image Segmentation

TODO

5 Convolutional Neural Networks