Data-driven Intelligent Systems

Lecture 15
Ensemble Learning 2

KNOWLEDGE
TECHNOLOGY

http://www.informatik.uni-hamburg.de/WTM/

AdaBoost Short Summary

For $t = 1,\dots T$

{

Step1: Find best classifier h_t, which minimizes error $\varepsilon_t = \sum_{i=1}^{n} D_t(i) * I_{[h_t(x_i) \neq y_i]}$

where $I_{[h_t(x_i) \neq y_i]} = 1$, if incorrect classification

Stop if $\varepsilon_t \geq 0.5$

Step2, weight of classifier : $\alpha_t = \frac{1}{2} \ln \frac{1 - \varepsilon_t}{\varepsilon_t}$

Step3, weight of data : $D_{t+1}(i) = \frac{D_t(i) \exp(-\alpha_t y_i h_t(x_i))}{Z_t}$

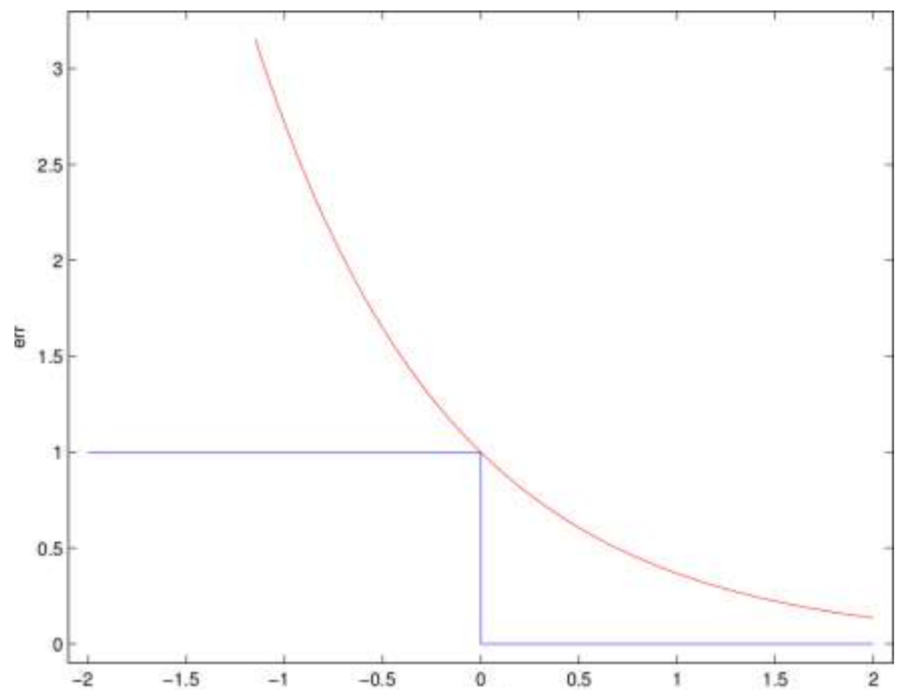
}

Final strong classifier : $H(x) = sign\left(\sum_{t=1}^{T} \alpha_t h_t(x) \right)$

Loss Function View

- AdaBoost finds the α_t that minimize the exponential loss:

$$L_{\exp}(x,y) = e^{-y\,H(x)}$$



- Full objective function:

$$E = \sum_i e^{-1/2\, y_i \sum_t \alpha_t h_t(x_i)}$$

- Upper bound on error:

$$L_{\exp}(x,y) \geq L_{0\text{-}1}(x,y)$$

Adapted from: Aaron Hertzmann, David Fleet:
CSC 411 / CSC D11 Chapter 18 Adaboost

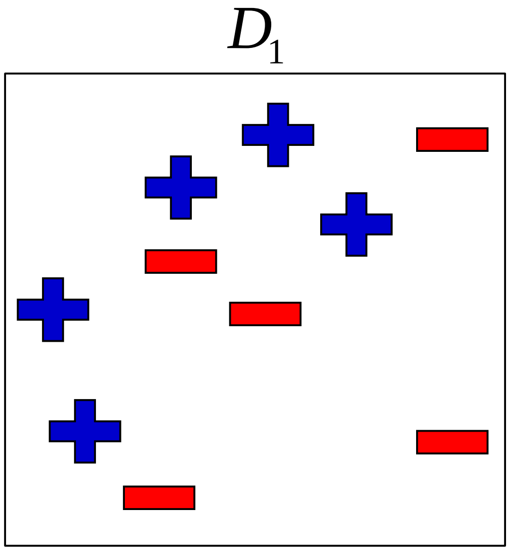
Loss Function View (2)

- Loss function discovered long after the algorithm
- Loss function explains the formula for setting the classifier weights α_t (Step2)
- Weights α_t optimize the objective function *E* in a greedy fashion
- Gradient descent on exponential loss function would not be recommendable
- Adding more classifiers, heuristically, can lead to either:
  - Overfitting
  - Improving test performance, since increasing the margin

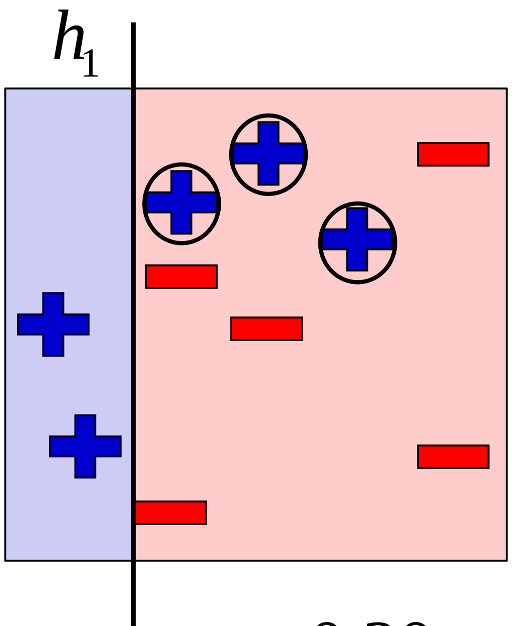
Further reading: Aaron Hertzmann and David Fleet: CSC 411 / CSC D11 Chapter 18 Adaboost

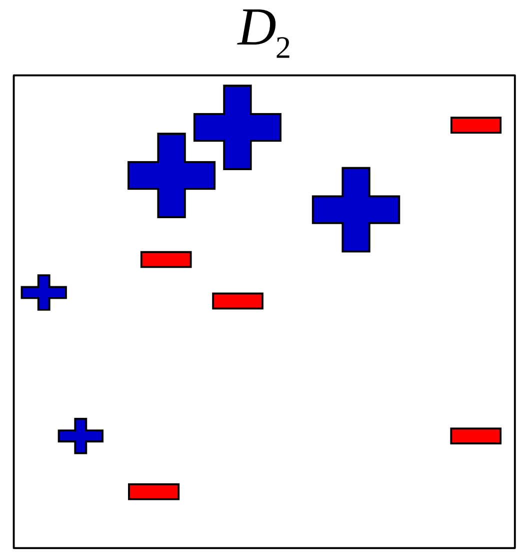
AdaBoost: Toy Example

- Weak classifiers = vertical or horizontal half-planes:



Round One:

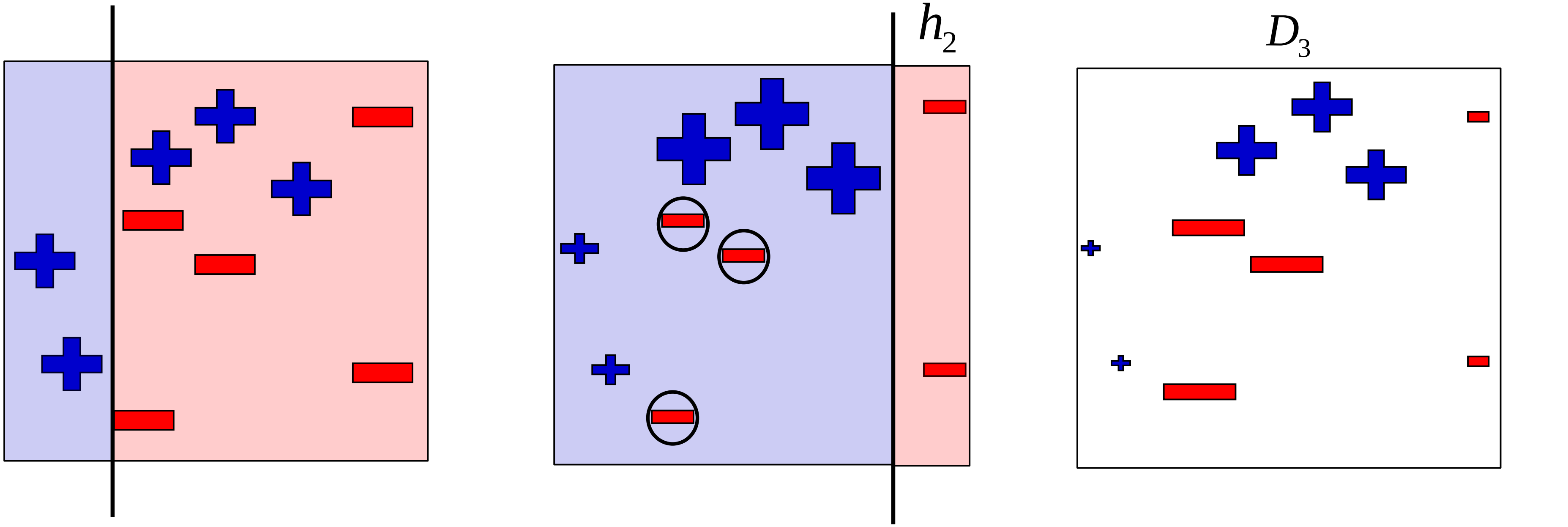




Error $\varepsilon_1 = 0.30$

Weight $\alpha_1 = 0.42$

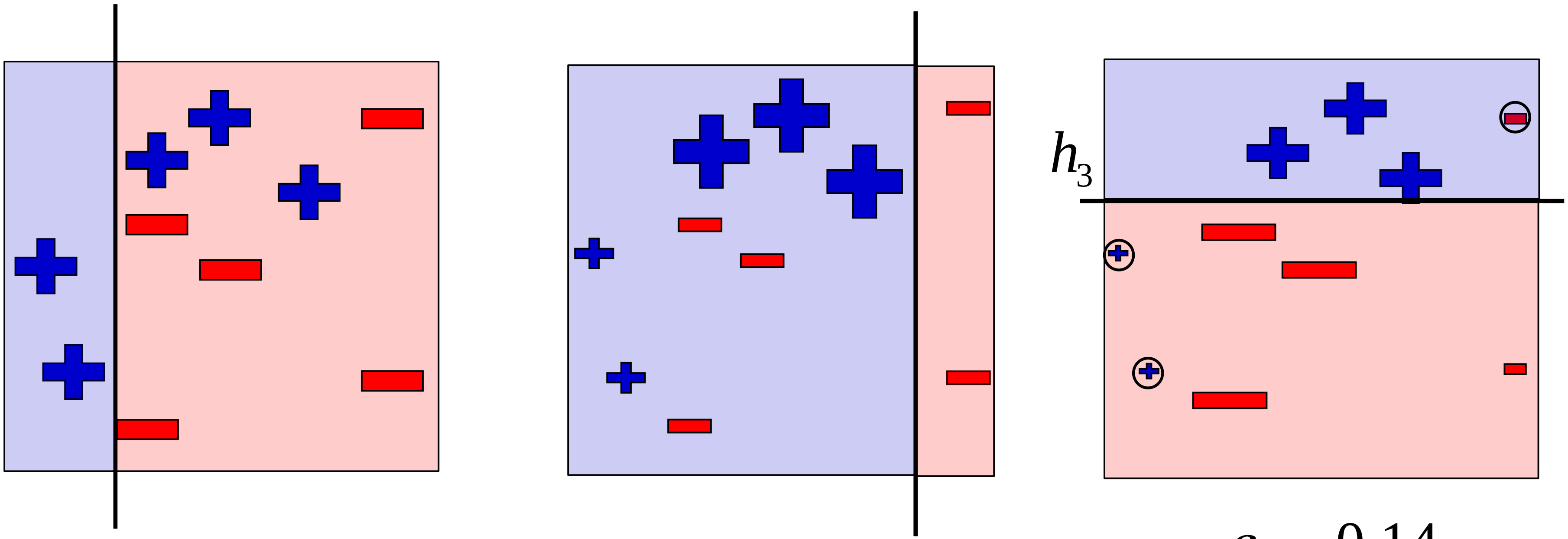
Round Two:



$$\varepsilon_2 = 0.21$$

$$\alpha_2 = 0.65$$

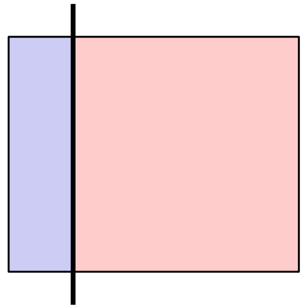
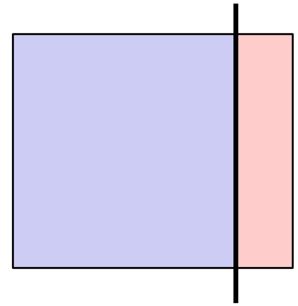
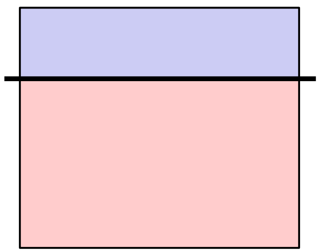
Round Three:

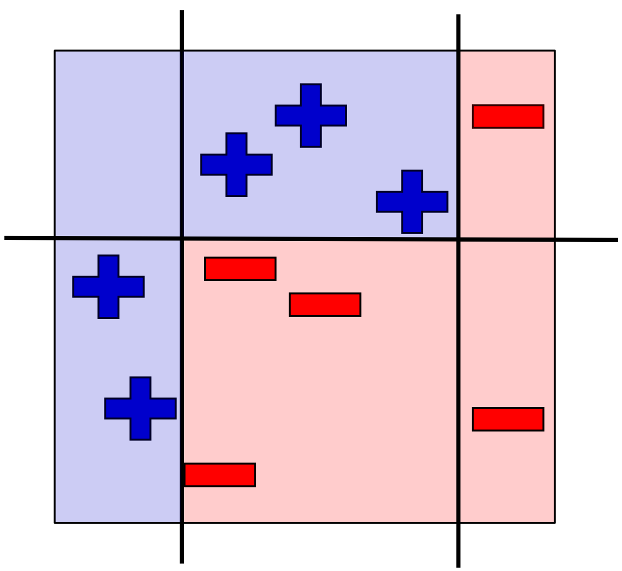


$$\varepsilon_3 = 0.14$$

$$\alpha_3 = 0.92$$

Final Classifier:

$$H_{final} = sign\Big(0.42$$  $$+\,0.65$$  $$+\,0.92$$  $$\Big)$$

$=$ 

Based on these principles of ***AdaBoost Algorithm***, many variants exist depending on:

- how to set the weights, and
- how to combine the hypotheses

Boosting Summary (1)

- Originally developed by computational learning theorists – [Schapire, 1990] (weak learner).
- Revised to become a practical algorithm, ***AdaBoost***, for building ensembles that empirically improves generalization performance [Freund & Shapire, 1996]
- ***AdaBoost*** - key insights:
  - Instead of sampling (as in bagging) re-weigh examples!
  - Final classification based on weighted vote of weak classifiers
  - Needs smaller number of training samples than bagging

Boosting Summary (2)

- Advantages of boosting
  - Flexibility in the choice of weak learners
  - Testing is fast
  - Easy to implement
  - Integrates classification with feature selection
  - Complexity of training linear in the number of training samples
  - Has been extended to multi-class AdaBoost [Zhu et al., 2006]
- Disadvantages
  - Minimizes classification error but not, e.g., false negatives
  - Can overfit in the presence of noise
  - No true hierarchical architecture

Ensemble Learning 2 – Overview

- Boosting for face detection
- Cascades of classifiers
- Ensembles and Neural Networks
  - MLP vs. AdaBoost
  - Dropout regularization

Boosting for Face Detection (1)

- *Basic idea:* slide a window across image
  and evaluate a face model at every location

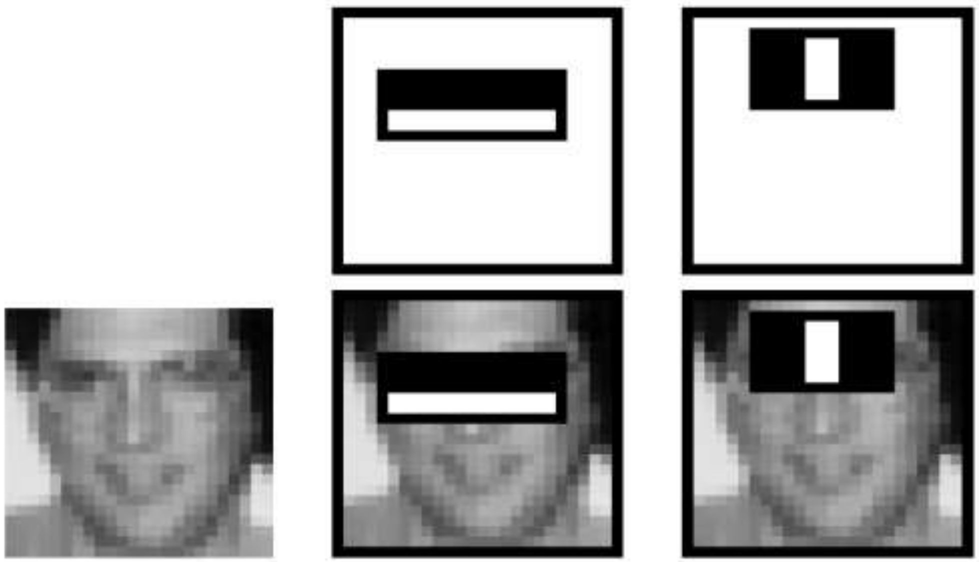
Boosting for Face Detection (2)

- Define ***weak learners*** based on rectangle features
- For each round t of boosting:
  - Evaluate each rectangle filter on each sample
  - Select best filter/threshold combination
  - Calculate filter weight (α_t)
  - Reweight samples ($D_{t+1}(i)$)

- Computational ***complexity*** of learning: O(TNK)
  - T rounds, N samples, K features

Boosting for Face Detection (3)

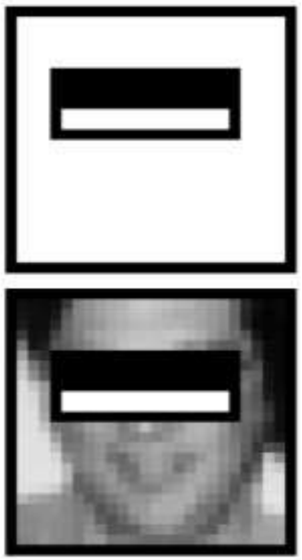
- First two features selected by boosting:

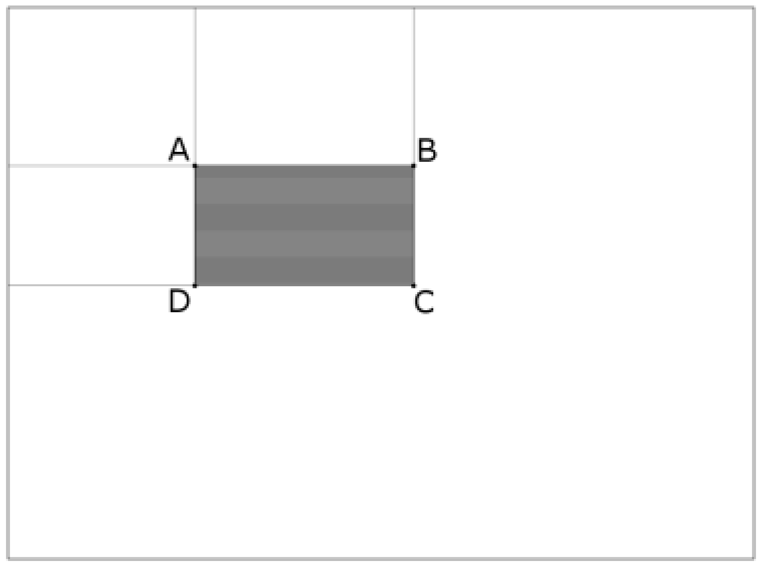


- This feature combination can yield ~100% detection rate, however, while also finding many false positives

Boosting for Face Detection (4)

- Efficient computation of rectangle sums via integral image:





$$I(x,y) = \sum_{\substack{x'<x \\ y'<y}} i(x',y')$$

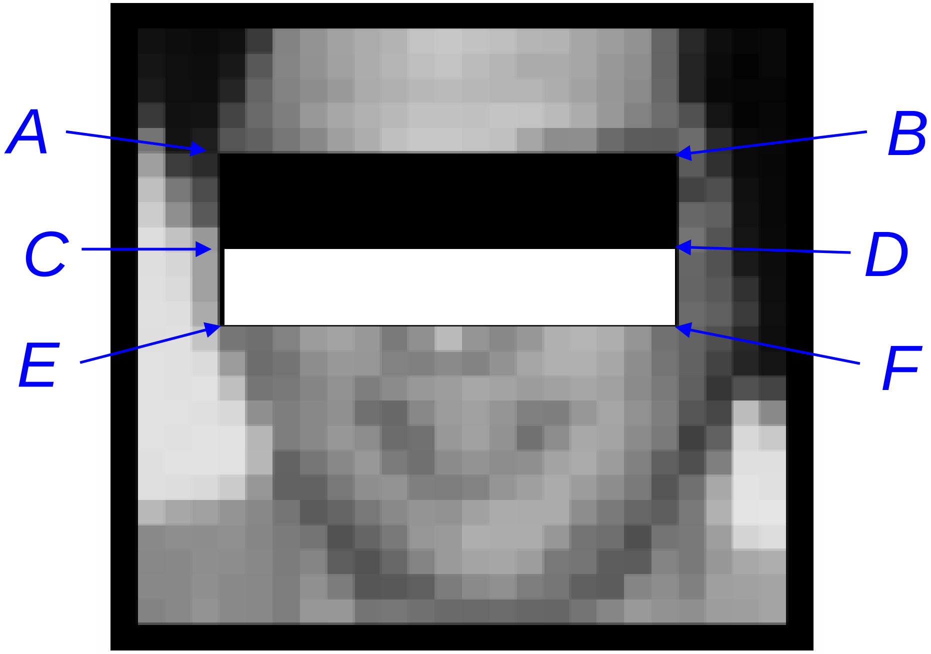
i = pixel brightness value at position (x',y')

Rectangle sum: $R = I(A) + I(C) - I(B) - I(D)$

(= sum of brightness values within rectangular region)

Boosting for Face Detection (5)

- Filter response *f* using rectangle sums *R* on integral image *I*:

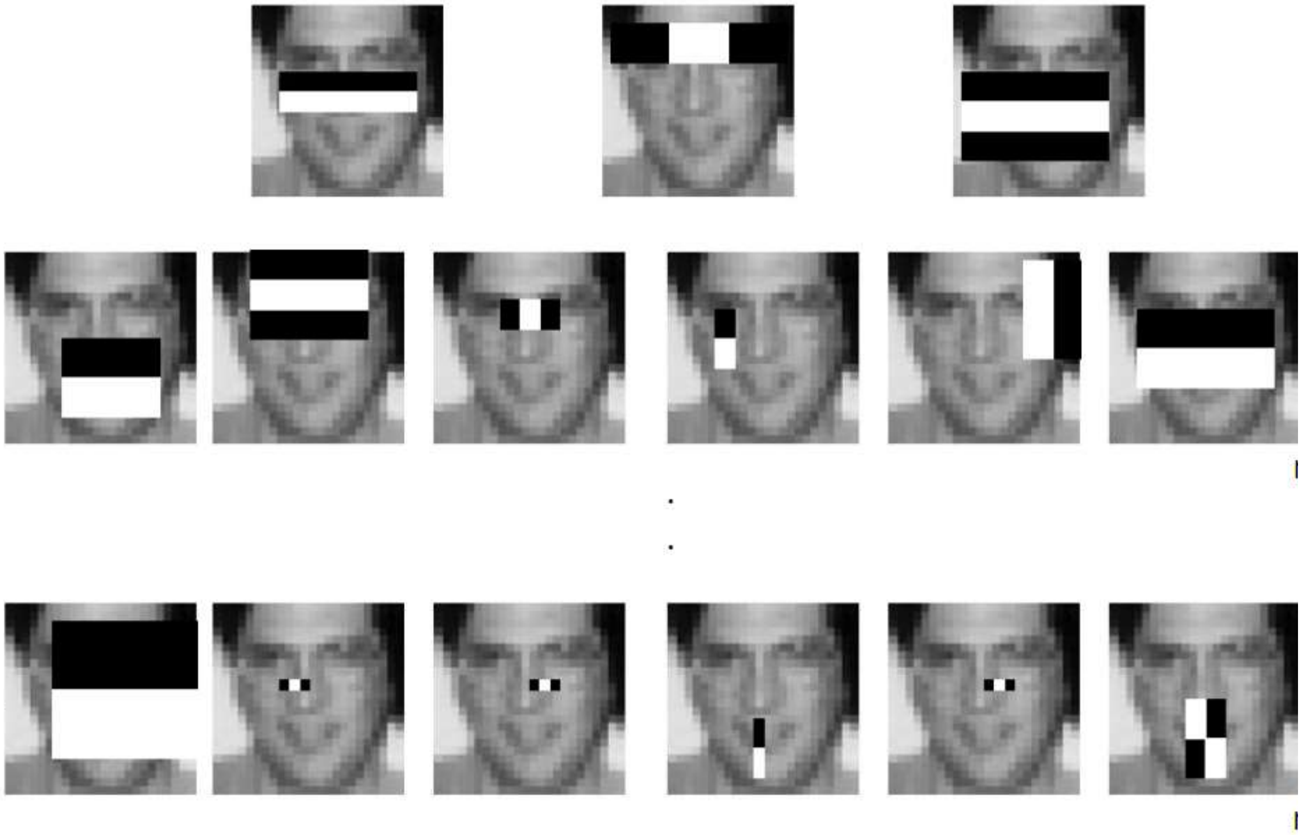


$$
\begin{aligned}
f &= R(CDEF) - R(ABCD) \\
&= I(C) + I(F) - I(E) - I(D) \\
&\quad -(I(A) + I(D) - I(C) - I(B)) \\
&= -I(A) + I(B) + 2I(C) - 2I(D) \\
&\quad -I(E) + I(F)
\end{aligned}
$$

- Classifier response: *h = sign(f)*
- Face detected, if *h = 1*

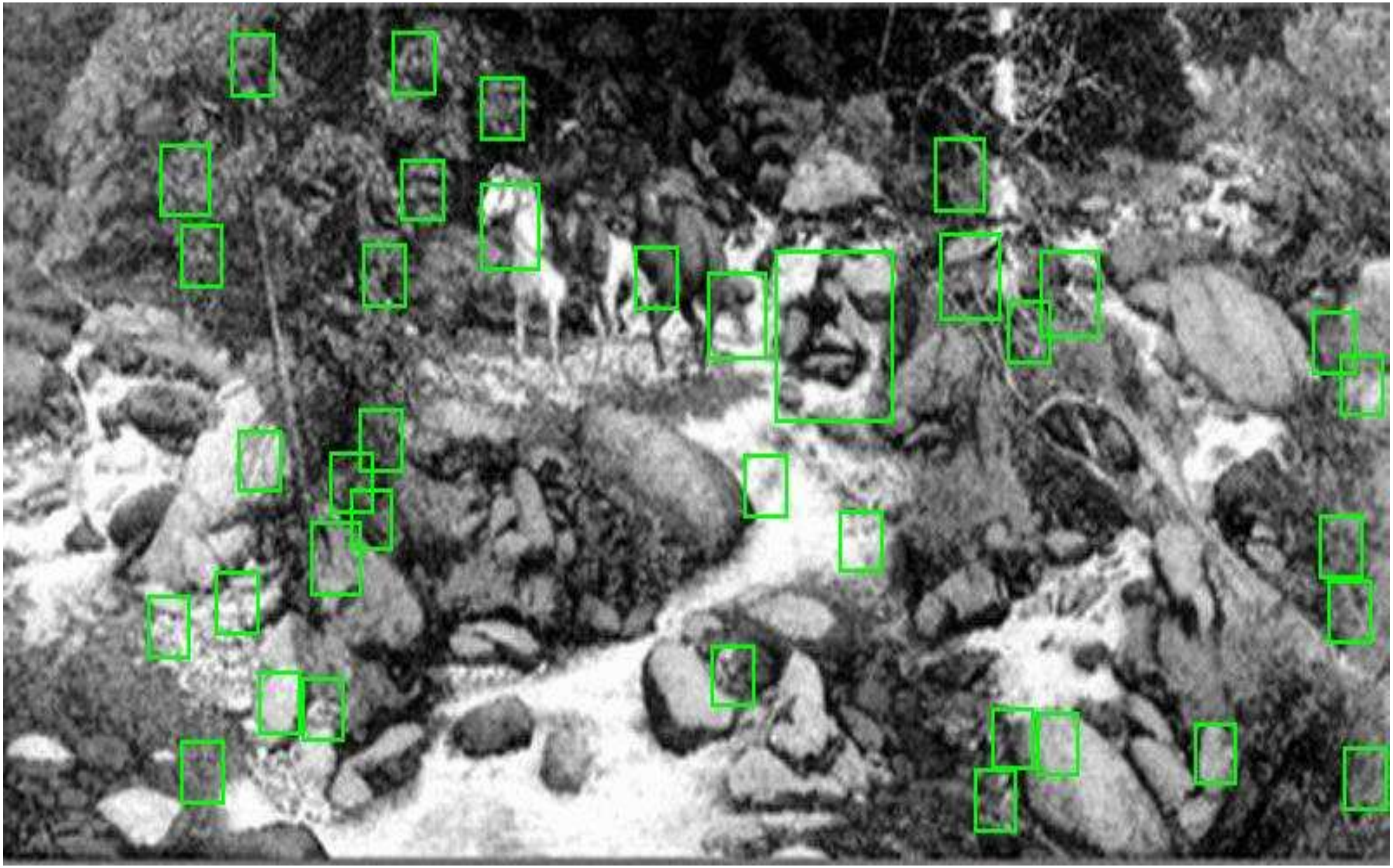
Boosting for Face Detection (6)

- More features selected by boosting:

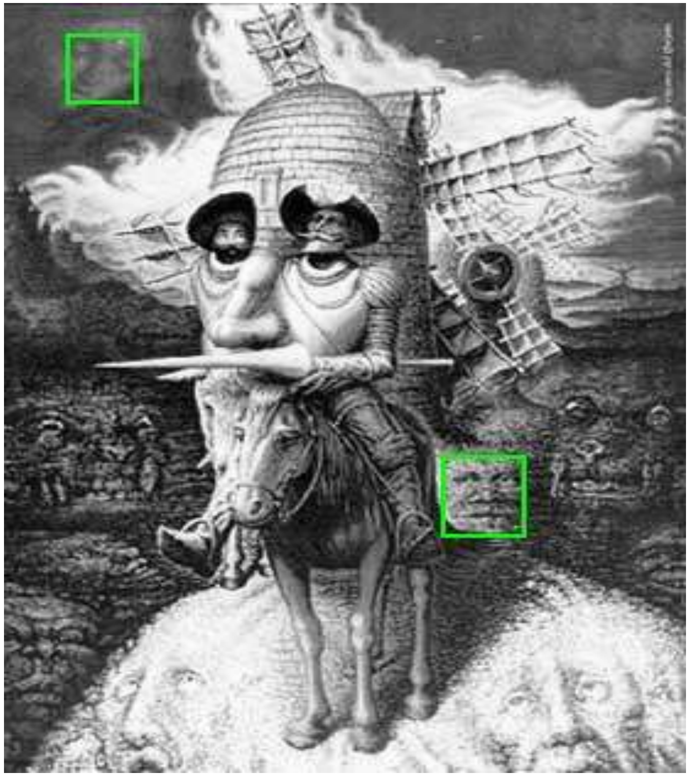


Boosting for Face Detection (7)

Boosting for Face Detection (7)

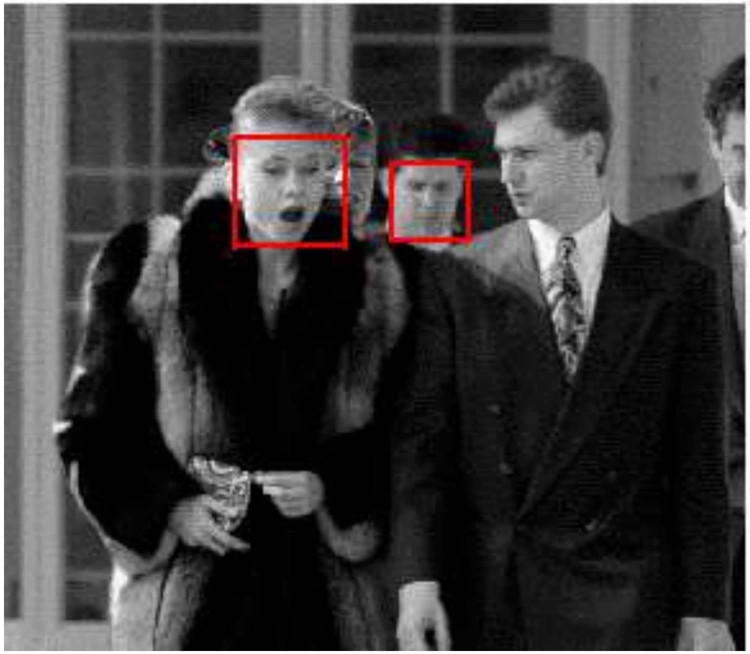


Boosting for Face Detection (8)



Boosting for Face Detection (9)

- Scale- and shift invariance are built-in



- Limitations with occlusion and rotations

Anti Automatic Face Detection

- Create asymmetry & new features, conceil landmarks, …

… Boosting for Face Detection …

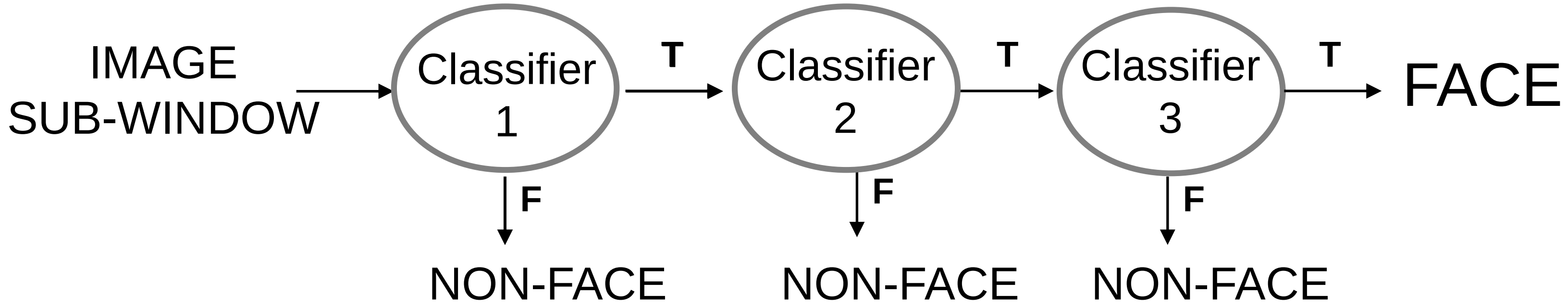
- Evaluate face model at every location
- *Inefficient:* detailed analysis of large image regions

Ensemble Learning 2 – Overview

- Boosting for face detection
- Cascades of classifiers
- Ensembles and Neural Networks
  - MLP vs. AdaBoost
  - Dropout regularization

Attentional Cascades

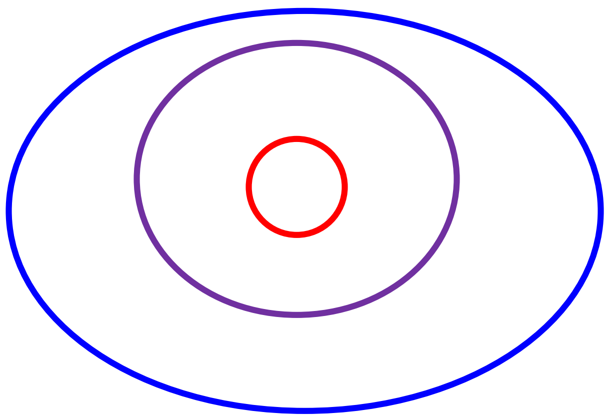
- Start with simple classifiers which reject many of the negative sub-windows while detecting (almost) all positive sub-windows
- Positive response from the first classifier triggers the evaluation of a second (more complex) classifier, and so on…
- A negative outcome at any point leads to the immediate rejection of the sub-window

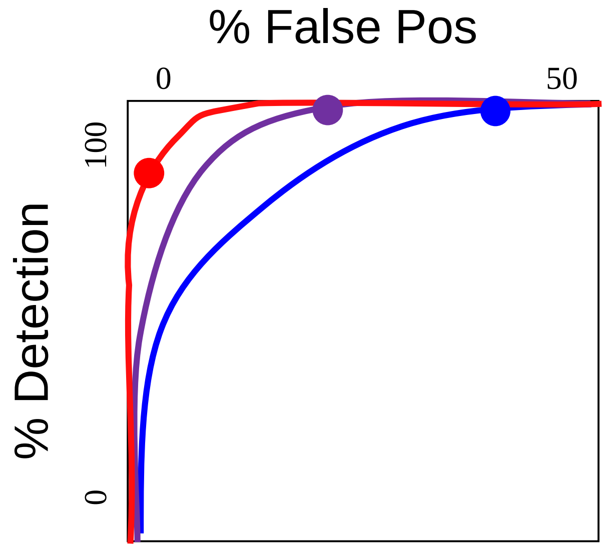


Attentional Cascades (2)

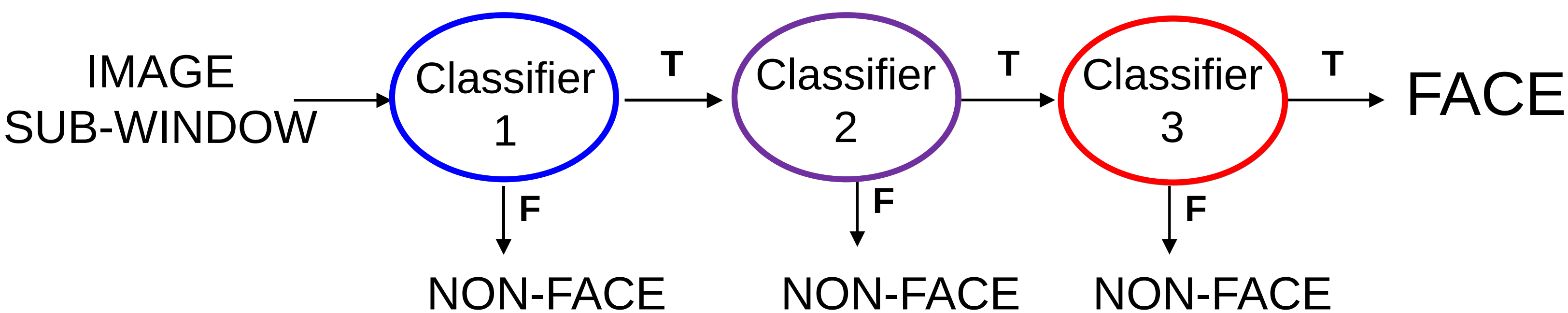
- Chain classifiers that are progressively more complex and have lower false positive rates

predicted positives





Receiver Operating Characteristic



Ensembles and AdaBoost Summary

- **Ensembles** combine classifiers to improve the accuracy
  - Together, act as one strong classifier
  - Simple ensemble example: equal voting over all members

- **AdaBoost**: algorithm to select the classifier with the lowest error on a training set
  - Taking into account the weights from the single images
  - Get different weak classifiers that complement each other
  - The result is a weighted voting
    over all weak classifiers

Ensemble Learning 2 – Overview

- Boosting for face detection
- Cascades of classifiers
- Ensembles and Neural Networks
  - MLP vs. AdaBoost
  - Dropout regularization

AdaBoost vs. MLP with 1 Hidden Layer

perceptron-like output

$$\text{Final strong classifier: } \; H(x) = sign\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$$

weights to output unit

weak classifiers / hidden units

AdaBoost		**MLP**
H, h_t binary	—	differentiable transfer func.
weak classifiers constructed	—	hidden neurons trained
weak classifiers selected sequentially (cf. *decis. tree*)	—	training simultaneously
Hierarchy of ensembles possible; need supervised training at each level	—	hierarchically extendable → deep NN end-to-end trainable per backpropagation

Ensemble Learning 2 – Overview

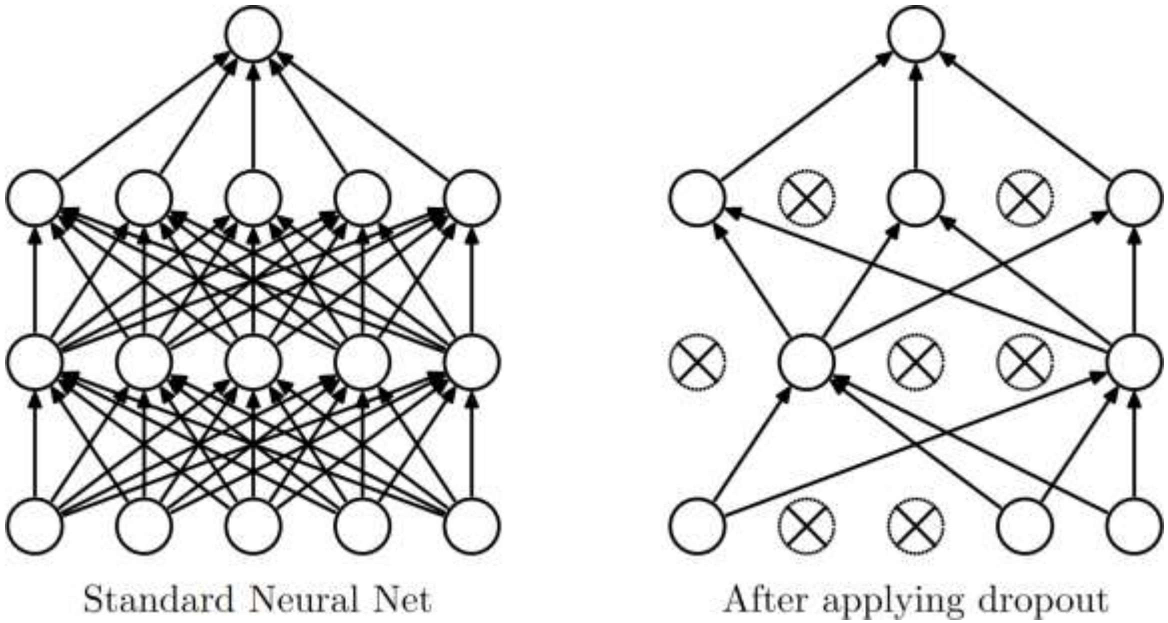
- Boosting for face detection
- Cascades of classifiers
- Ensembles and Neural Networks
  - MLP vs. AdaBoost
  - Dropout regularization

Dropout – Against Overfitting

- Neural networks overfit to noise in the training data, if too many trainable parameters are supported by too few data
- SRM advises to constrain the network structure:
  - Crossvalidation – find network of optimal complexity / size
  - Early stopping – prevent parameters to overfit
  - Regularization – e.g. limit the size of the parameters
- Another technique: dropout

Dropout – Idea

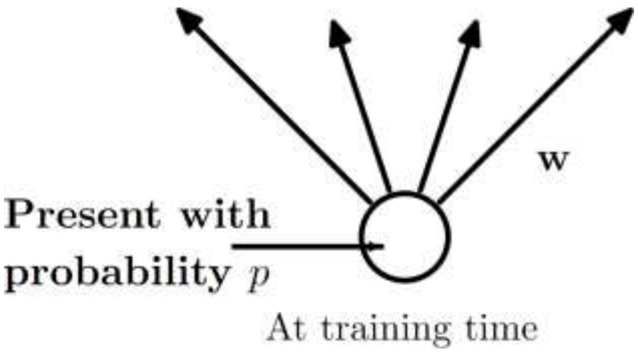
- Randomly inactivate individual neurons
  - prevent reliance on individual nodes
  - for each data point, there is a different network structure

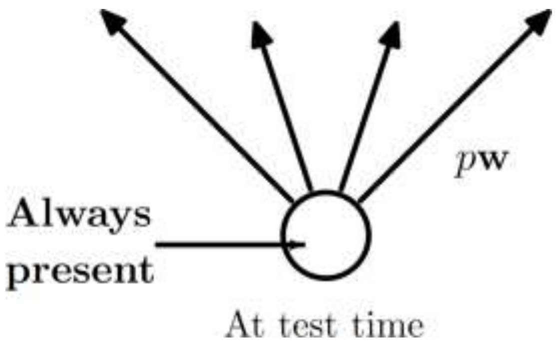


Srivastava et al. (2014) Dropout: A Simple Way to Prevent Neural Networks from Overfitting

Dropout – During Learning and Usage

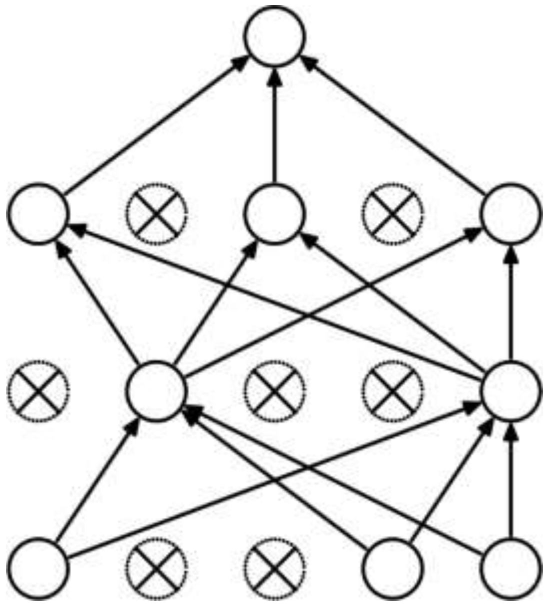
- Learning:
  - a unit has probability *p* to be present
- Usage by Monte-Carlo averaging:
  - use network many times with different dropout instances; use averaging or voting for the result
- Usage by weight scaling:
  - units always present at test time, scale output weights by *p*
    (this compensates for the increased number of units in a layer)





Srivastava et al. (2014) Dropout: A Simple Way to Prevent Neural Networks from Overfitting

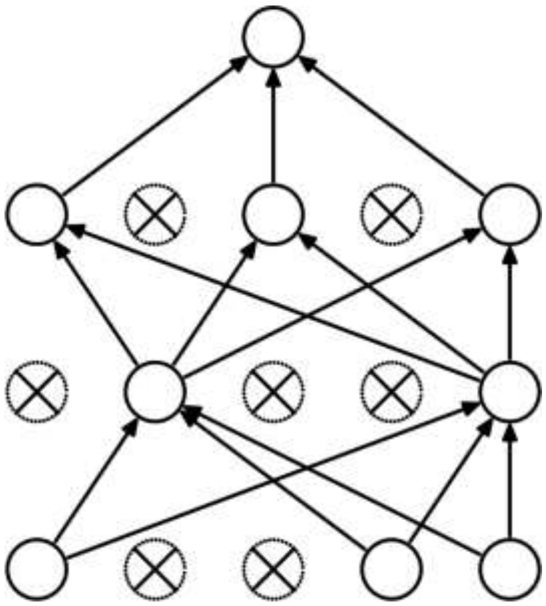
Dropout – Enforces Redundant Coding



- Processing cannot rely on individual units
  - units within a layer must share their function
- Dropout on input units ~ feature bagging

Dropout – Interpretation as Ensemble

- Dropout is training a large ensemble of models (that share parameters).
- Each instance is one model, which gets trained on a small subset of the data

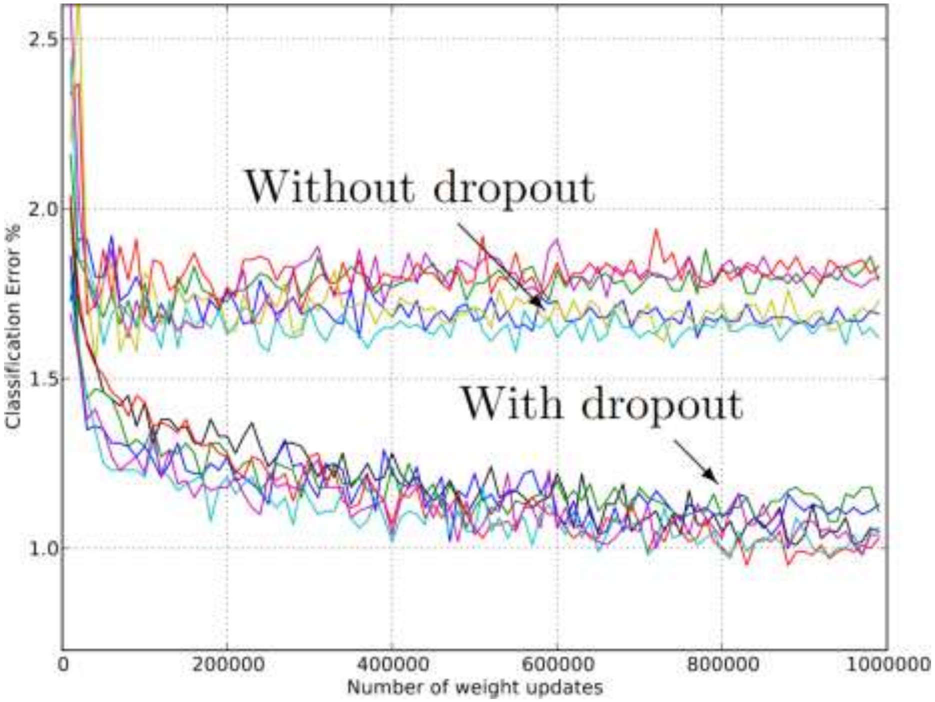


- Overall task then solved by an ensemble of NNs

Srivastava et al. (2014) Dropout: A Simple Way to Prevent Neural Networks from Overfitting

Dropout – Lower Test / Validation Error

- Dropout will most likely increase the training error
- The more important test (validation) error has been shown to descrease, for several different architectures



Example task: image classification (MNIST handwritten digits); networks had 2-4 hidden layers and 1024-2048 units each

Srivastava et al. (2014) Dropout: A Simple Way to Prevent Neural Networks from Overfitting

Comparison: Classification with …

Bagging & co.		Boosting
Achieve more trustable ensemble results, compared to individual noisy classifier	—	Build more complex classifiers from weak classifiers
Each individual has the solution	—	Individual classifier is too weak
Diversity e.g. through different subspaces or subsets of training data	—	Diverse weak classifiers selected by reweighting of training data
Joint decision e.g. by majority vote	—	Combination by weighted vote

Ensembles Summary

- Ensembles better than an individual
- Diversity is key
- Bagging – resampling of data
- Boosting – reweighting of data – AdaBoost
- Dropout – regularization via an ensemble of neural networks