Learning and Adaptivity

Ensembles of classifiers - Lecture V

Course Outline

Basic Concepts

- Parametric Method,
- Bayesian Learning and Nonparametrics Methods
- Clustering and Mixture of Gaussians

Supervised Learning, Classification Approaches

- Ensemble Methods and Boosting
- Randomized Trees, Forest

Unsupervised Learning

- Dimensionality Reduction and Manifold Learning (PCA, SNE/t-SNE, MDS, umap)
- Uncertainty Estimation

Reinforcement Learning

- Classical Reinforcement Learning
- Deep Reinforcement Learning

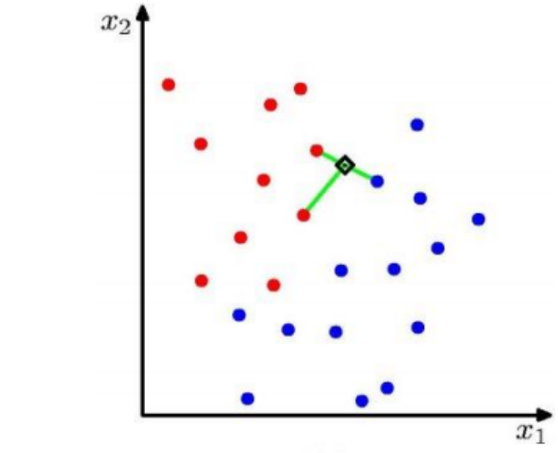
So Far…

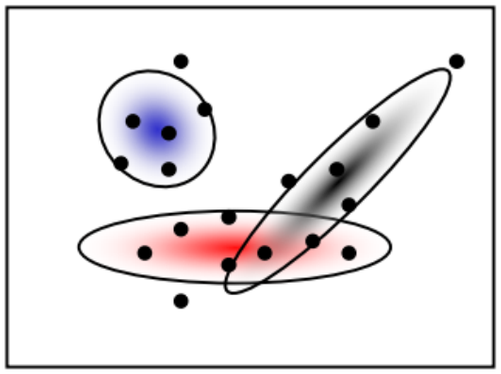
Study:

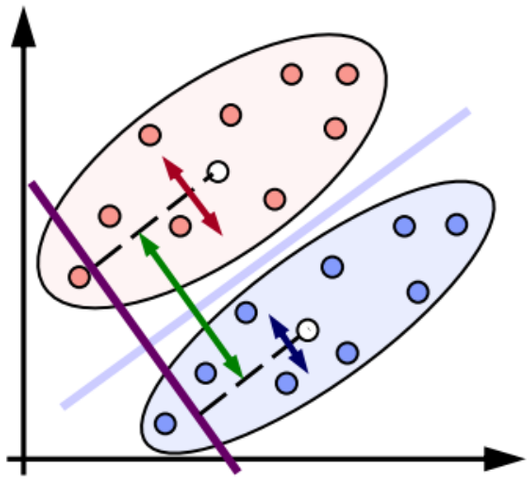
- k-NN
- Bayes classifiers

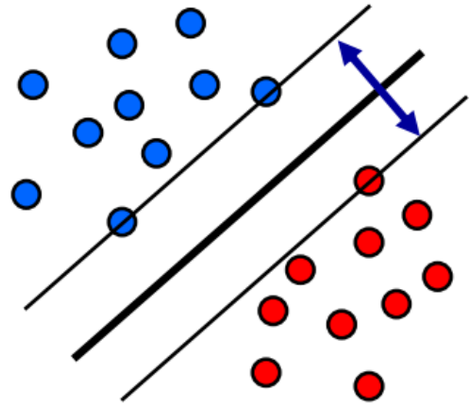
From NN-class

- ~~Linear discriminants~~ NN
- SVMs









Each of them has their strengths and weaknesses…
=>Can we improve performance by combining them?

Slide source: B. Leibe Slides ML, 2015

Ensembles of Classifiers

Intuition

- Assume we have K classifiers
- They are independent (i.e., their errors are uncorrelated).
- Each of them has an error probability $\boldsymbol{p}$<0.5 on training data.
  - Why can we assume that $\boldsymbol{p}$ will not be larger than 0.5?
- Then a simple majority vote of all classifiers should have a lower error than each individual classifier…

Slide source: B. Leibe Slides ML, 2015

Today's Topics

Ensembles of classifiers

- Bagging
- Bayesian Model Averaging

Methods for obtaining a set of classifiers

Combining Classifiers

- Stacking
- Bayesian model averaging
- Boosting

Methods for combining different classifiers

AdaBoost modern SAO for state classifier

- Intuition
- Algorithm
- Analysis
- Extensions

Slide source: B. Leibe Slides ML, 2015

Ensembles of Classifiers

How do we get different classifiers?

Simplest case: train same classifier on different data.

But… where shall we get this additional data from? (training data can be **expensive**!)

Idea: Subsample the training data

Reuse the same training algorithm several times on different

subsets of the training data.

Well-suited for “unstable” learning algorithms

Unstable: small differences in training data can produce very different classifiers

– E.g., Decision trees, neural networks, rule learning algorithms,…

Stable learning algorithms

– E.g., Nearest neighbor, linear regression, SVMs,…

Slide source: B. Leibe Slides ML, 2015

Constructing Ensembles

Cross-Validation

Split the available data into N disjunct subsets.

In each run, train on N-1 subsets for training a classifier.

Estimate the generalisation error on the held-out validation set

E.g. 5-fold cross-validation

Slide source: B. Leibe Slides ML, 2015

Constructing Ensembles

Bagging = ‘Bootstrap aggregation’ (Breiman 1996)

In each run of the training algorithm, randomly select M samples from the full set of N training data points.

If M=n, then on average, 63.2% of the training points will be represented. the results are duplicates.

Injecting randomness

Many (iterative) learning algorithm need a random initialization (e.g.k-means, EM)

Perform multiple runs of the learning algorithm with different random initialisations.

Slide source: B. Leibe Slides ML, 2015

Today's Topics

Ensembles of classifiers

- Bagging
- Bayesian Model Averaging

Methods for obtaining a set of classifiers

Combining Classifiers

- **Stacking**
- **Bayesian model averaging**
- **Boosting**

Methods for combining different classifiers

AdaBoost

- Intuition
- Algorithm
- Analysis
- Extensions

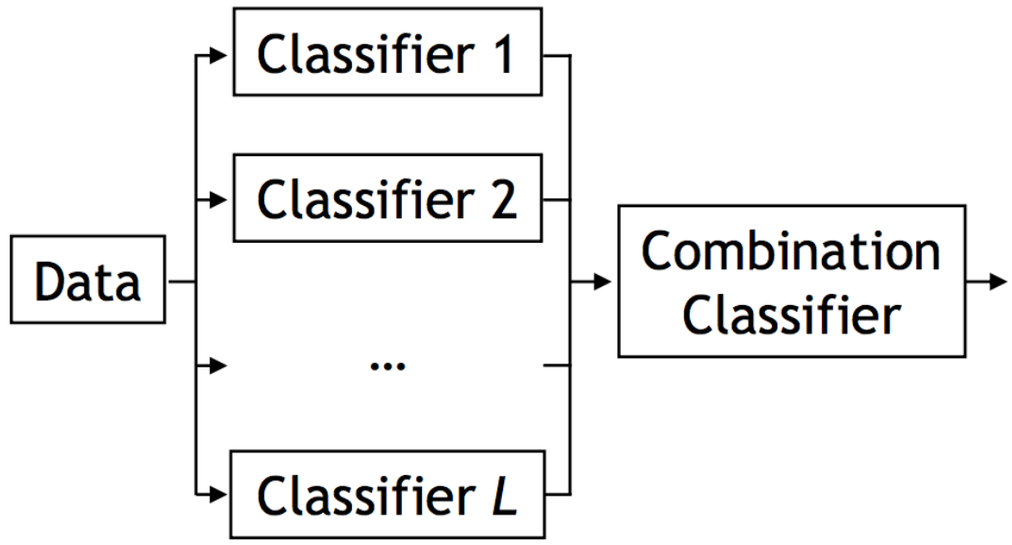
Slide source: B. Leibe Slides ML, 2015

Stacking

Idea

Learn L classifiers (based on the training data)

Find a meta-classifier that takes as input the output of the L first-level classifiers.



Example

Learn L classifiers with leave-one-out cross-validation

Interpret the prediction of the L classifiers as L-dimensional feature vector

Learn ‘level-2’ classifier based on the examples generated this way

Slide source: B. Leibe Slides ML, 2015

Stacking

Why can this be useful?

Simplicity

- We may already have several existing classifiers available

=>No need to train those, they can just be combined with the rest

Correlation between classifiers

- The combination classifier can learn the correlation

=> Better results than simple Naive Bayes combination

Feature combination

- E.g. combine information from different sensors or sources (vision, audio, acceleration, temperature, radar, etc.).

- We can get good training data for each sensor individually, but data from all sensors together is rare.

=> Train each of the L classifiers on its own input data. Only combination classifier needs to be trained on combined input.

Slide source: B. Leibe Slides ML, 2015

Model Combination

E.g. Mixture of Gaussians

Several components are combined probabilistically.

Interpretation: different data points can be generated by different components.

We model the uncertainty which mixture component is responsible for generating the corresponding data point:

$$p(x) = \sum_{k=1}^{K} \pi_k \mathcal{N}(x|\mu_k, \Sigma_k)$$

For i.i.d. data, we write the marginal probability of a data set $X = \{x_1, ..., x_N\}$ in the form:

$$p(X) = \prod_{n=1}^{N} p(x_n) = \prod_{n=1}^{N} \sum_{k=1}^{K} \pi_k \mathcal{N}(x_n|\mu_k, \Sigma_k)$$

Bayesian Model Averaging

Model Averaging

Suppose we have H different models $h=1,...,H$ with prior probabilities $p(h)$

Construct the marginal distribution over the data set

$$p(X) = \sum_{h=1}^{H} p(X|h)p(h)$$

Interpretation

Just one model is responsible for generating the entire data set

The probability distribution over h just reflects our uncertainty with model that is

As the size of the data set increases, this uncertainty reduces, and $p(X|h)$ becomes focused on just one of the models

Slide source: B. Leibe Slides ML, 2015

Note the Different Interpretations!

Model Combination (e.g., Mixtures of Gaussians)

Different data points generated by different model components.

Uncertainty is about which component created which data point.

⇨ One latent variable z_n for each data point:

$$p(X) = \prod_{n=1}^{N} p(x_n) = \prod_{n=1}^{M} \sum_{z_n} p(x_n, z_n)$$

Bayesian Model Averaging

The whole data set is generated by a single model.

Uncertainty is about which model was responsible.

⇨ One latent variable z for the entire data set:

$$p(X) = \sum_{z} p(X, z)$$

Slide source: B. Leibe Slides ML, 2015

Model Averaging: Expected Error

Combine M predictors $y_m(x)$ for target output $h(x)$.

E.g. each trained on a different bootstrap data set **by bagging**.

The committee prediction is given by

$$y_{COM}(\mathbf{x}) = \frac{1}{M}\sum_{m=1}^{M} y_m(\mathbf{x})$$

The output can be written as the true value plus some error

$$y(\mathbf{x}) = h(\mathbf{x}) + \epsilon(\mathbf{x})$$

Thus, the expected sum-of-squares error takes the form

$$\mathbb{E}_{\mathbf{x}} = \left[\{y_m(\mathbf{x}) - h(\mathbf{x})\}^2\right] = \mathbb{E}_{\mathbf{x}}\left[\epsilon_m(\mathbf{x})^2\right]$$

Slide source: B. Leibe Slides ML, 2015

Model Averaging: Expected Error

Average error of individual models: $\mathbb{E}_{AV} = \frac{1}{M}\sum_{m=1}^{M}\mathbb{E}_{\mathbf{x}}\left[\epsilon_m(\mathbf{x})^2\right]$

Average error of committee:

$$\mathbb{E}_{COM} = \mathbb{E}_{\mathbf{x}}\left[\left\{\frac{1}{M}\sum_{m=1}^{M}y_m(\mathbf{x}) - h(\mathbf{x})\right\}\right] = \mathbb{E}_{\mathbf{x}}\left[\left\{\frac{1}{M}\sum_{m=1}^{M}\epsilon_m(\mathbf{x})\right\}\right]$$

Assumptions

Errors have zero mean: $\mathbb{E}_{\mathbf{x}}\left[\epsilon_m(\mathbf{x})\right] = 0$

Errors are uncorrelated: $\mathbb{E}_{\mathbf{x}}\left[\epsilon_m(\mathbf{x})\epsilon_j(\mathbf{x})\right] = 0$

Then: $\mathbb{E}_{COM} = \frac{1}{M}\mathbb{E}_{AV}$

Slide source: B. Leibe Slides ML, 2015

Model Averaging: Expected Error

Average error of committee $\mathbb{E}_{COM} = \frac{1}{M}\mathbb{E}_{AV}$

This suggests that the average error of a model can be reduced by a factor of M simply by averaging M versions of the model!

Spectacular indeed…

This sounds almost too good to be true…

And it is… Can you see where the problem is?

Unfortunately, this result depends on the assumption that the errors are all uncorrelated.

In practice, they will typically be highly correlated.

Still, it can be shown that $\mathbb{E}_{COM} \leq \mathbb{E}_{AV}$

Slide source: B. Leibe Slides ML, 2015

Discussion: Ensembles of Classifiers

Set of simple methods for improving classification

Often effective in practice this be useful?

Apparent contradiction

We have stressed before that a classifier should be trained on samples from the distribution on which it will be tested.

Resampling seems to violate this recommendation.

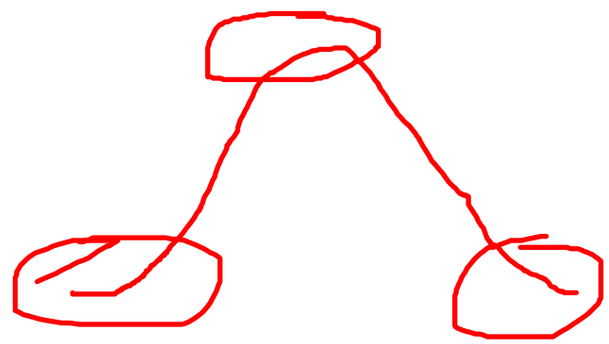
Why can a classifier trained on a weighted data distribution do better than one trained on the i.i.d. sample?

Explanation

We do not attempt to model the full category distribution here.

Instead, try to find the decision boundary more directly.

Also, increasing number of component classifiers broadens the class of implementable decision functions.



Having different experts to model the data

Slide source: B. Leibe Slides ML, 2015

Today's Topics

Ensembles of classifiers

- Bagging
- Bayesian Model Averaging

Combining Classifiers

- Stacking
- Bayesian model averaging
- Boosting

AdaBoost

- **Intuition**
- **Algorithm**
- **Analysis**
- **Extensions**

Slide source: B. Leibe Slides ML, 2015

AdaBoost – "Adaptive Boosting"

Main idea

Instead of resampling, reweight misclassified training examples.

Increase the chance of being selected in a sampled training set.

Or increase the misclassification cost when training on the full set.

Components

$h_m(x)$: "weak" or base classifier

– Condition: <50% training error over any distribution

$H(x)$: "strong" or final classifier

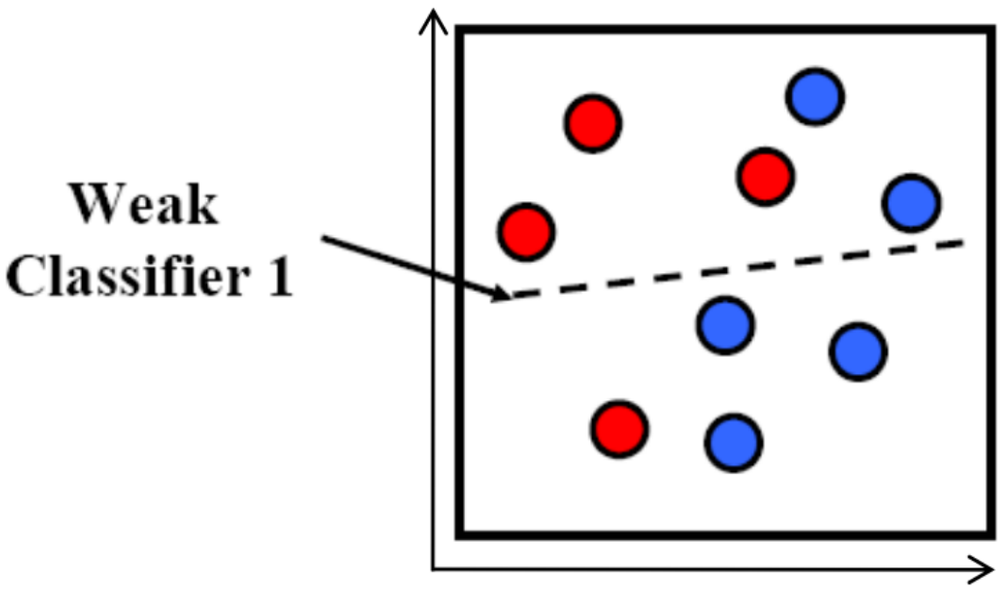
AdaBoost:

Construct a strong classifier as a thresholded linear combination of the weighted weak classifiers:

$$H(\mathbf{x}) = sign\left(\sum_{m=1}^{M} \alpha_m h_m(\mathbf{x})\right)$$

Slide source: B. Leibe Slides ML, 2015

AdaBoost: Intuition



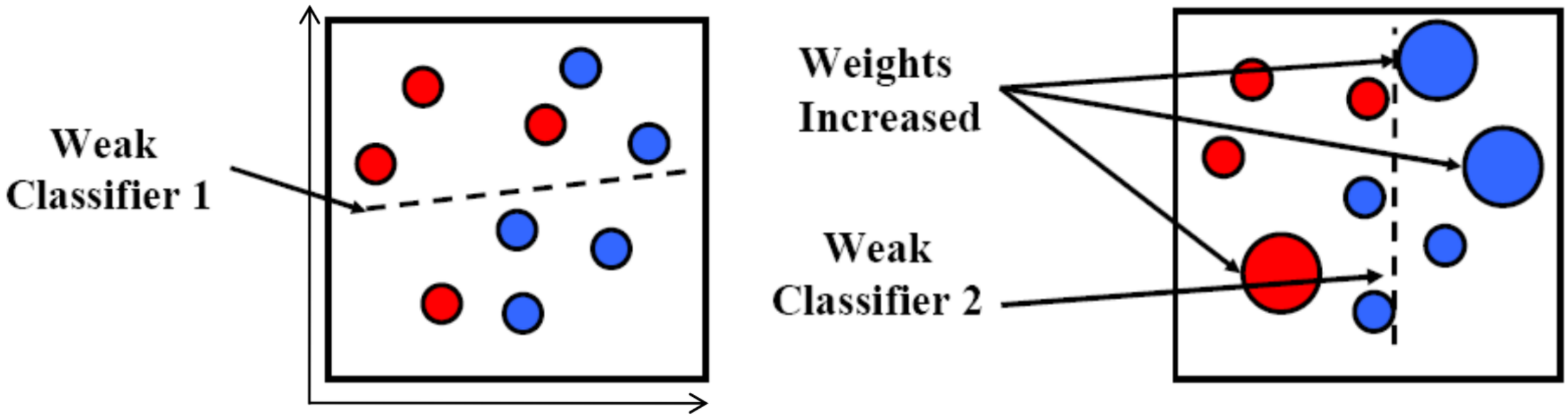
Consider a 2D feature space with positive and negative examples.

Each weak classifier splits the training examples with at least 50% accuracy.

Examples misclassified by a previous weak learner are given more emphasis at future rounds.

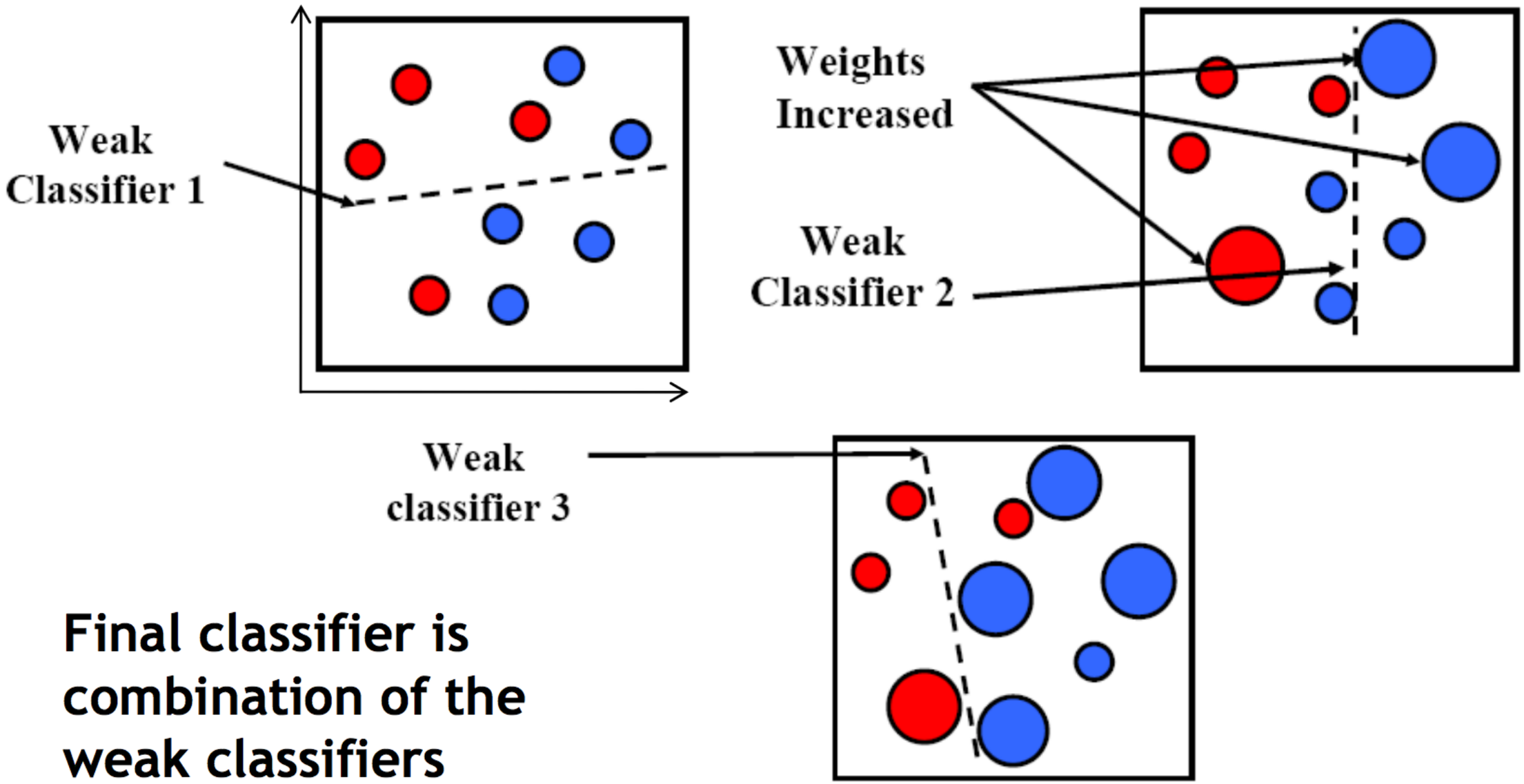
Slide source: B. Leibe Slides ML, 2015

AdaBoost: Intuition



Slide source: B. Leibe Slides ML, 2015

AdaBoost: Intuition



Slide source: B. Leibe Slides ML, 2015

AdaBoost – Formalization

2-class classification problem

- Given: training set $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$
  with target values $\mathbf{T} = \{t_1, \ldots, t_N\}$, $t_n \in \{-1,1\}$
- For each training point, the associated weights $\mathbf{W}=\{w_1, \ldots, w_N\}$

Basic steps

- In each iteration, AdaBoost trains a new weak classifier $h_m(x)$ based on the current weighting coefficients
  - We then adapt the weighting coefficients for each point
  - Increase w_n if x_n was misclassified by $h_m(x)$
  - Decrease w_n if x_n was classified correctly by $h_m(x)$
- Make predictions using the final combined model

$$H(\mathbf{x}) = sign\left(\sum_{m=1}^{M} \alpha_m h_m(\mathbf{x})\right)$$

Slide source: B. Leibe Slides ML, 2015

AdaBoost - Algorithm

1. Initialization: Set $w_n^{(1)} = \frac{1}{N}$ for $n = 1, ...N$
2. For $m = 1, ...M$ iterations

a) Train a new weak classifier $h_m(x)$ using the current weighting coefficients $W^{(m)}$ by minimizing the weighted error function

$$J_m = \sum_{n=1}^{N} w_n^{(m)} I(h_m(\mathbf{x}) \neq t_n)$$

I(A) { 1 if A is true; 0 else

b) Estimate the weighted error of this classifier on X:

$$\epsilon_m = \frac{\sum_{n=1}^{N} w_n^{(m)} I(h_m(\mathbf{x}) \neq t_n)}{\sum_{n=1}^{N} w_n^{(m)}}$$

c) Calculate a weighting coefficient for $h_m(x)$: $a_m = ?$

d) Update the weighting coefficients: $w_n^{(m+1)} = ?$

How should we do this exactly?

Slide source: B. Leibe Slides ML, 2015

AdaBoost – Historical Development

Originally motivated by Statistical Learning Theory

- AdaBoost was introduced in 1996 by Freund & Schapire.
- It was empirically observed that AdaBoost often tends not to overfit. (Breiman 96, Cortes & Drucker 97, etc.)
- As a result, the margin theory (Schapire et al. 98) developed, which is based on loose generalization bounds.
  - Note: margin for boosting is not the same as margin for SVM.
  - A bit like retrofitting the theory…
- However, those bounds are too loose to be of practical value.

Different explanation (Friedman, Hastie, Tibshirani, 2000)

- Interpretation as sequential minimization of an exponential error function ("Forward Stagewise Additive Modeling").
- Explains why boosting works well.
- Improvements possible by altering the error function.

Slide source: B. Leibe Slides ML, 2015

AdaBoost – Minimizing Exponential Error

Exponential error function

$$E = \sum_{n=1}^{N} \exp\{-t_n f_m(\mathbf{x}_n)\}$$

where $f_m(x)$ is a classifier defined as a linear combination of base classifiers $h_l(x)$:

$$f_m(\mathbf{x}) = \frac{1}{2} \sum_{l=1}^{m} \alpha_l h_l(\mathbf{x})$$

Goal

Minimize E with respect to both the weighting coefficients α_l and the parameters of the base classifiers $h_l(x)$.

Slide source: B. Leibe Slides ML, 2015

AdaBoost – Minimizing Exponential Error

Sequential Minimization

Suppose that the base classifiers $h_1(x),\ldots,h_{m-1}(x)$ and their coefficients $\alpha_1,\ldots,\alpha_{m-1}$ are fixed.

=> Only minimize with respect to α_m and $h_m(x)$.

$$E = \sum_{n=1}^{N} \exp\{-t_n f_m(\mathbf{x}_n)\} \quad \textbf{with} \quad f_m(\mathbf{x}) = \frac{1}{2}\sum_{l=1}^{m} \alpha_l h_l(\mathbf{x})$$

$$= \sum_{n=1}^{N} \exp\left\{\underbrace{-t_n f_{m-1}(\mathbf{x}_n)}_{\textbf{= const.}} - \frac{1}{2} t_n \alpha_m h_m(\mathbf{x}_n)\right\}$$

$$= \sum_{n=1}^{N} w_n^{(m)} \exp\left\{-\frac{1}{2} t_n \alpha_m h_m(\mathbf{x}_n)\right\}$$

Slide source: B. Leibe Slides ML, 2015

AdaBoost – Minimizing Exponential Error

$$E = \sum_{n=1}^{N} w_n^{(m)} \exp\left\{-\frac{1}{2} t_n \alpha_m h_m(\mathbf{x}_n)\right\}$$

Observation:

- **Correctly classified points:** $t_n h_m(\mathbf{x}_n) = +1$ $\Rightarrow$ **collect in** $\mathcal{T}_m$
- **Misclassified points:** $t_n h_m(\mathbf{x}_n) = -1$ $\Rightarrow$ **collect in** $\mathcal{F}_m$

Rewrite the error function as

$$E = e^{-\alpha_m/2} \sum_{n \in \mathcal{T}_m} w_n^{(m)} + e^{\alpha_m/2} \sum_{n \in \mathcal{F}_m} w_n^{(m)}$$

$$= \left(e^{\alpha_m/2} \qquad\qquad\right) \sum_{n=1}^{N} w_n^{(m)} I(h_m(\mathbf{x}_n) \neq t_n)$$

Slide source: B. Leibe Slides ML, 2015

AdaBoost – Minimizing Exponential Error

$$E = \sum_{n=1}^{N} w_n^{(m)} \exp\left\{-\frac{1}{2} t_n \alpha_m h_m(\mathbf{x}_n)\right\}$$

Observation:

- **Correctly classified points:** $t_n h_m(\mathbf{x}_n) = +1$ $\Rightarrow$ **collect in** $\mathcal{T}_m$
- **Misclassified points:** $t_n h_m(\mathbf{x}_n) = -1$ $\Rightarrow$ **collect in** $\mathcal{F}_m$

Rewrite the error function as

$$E = e^{-\alpha_m/2} \sum_{n \in \mathcal{T}_m} w_n^{(m)} + e^{\alpha_m/2} \sum_{n \in \mathcal{F}_m} w_n^{(m)}$$

$$= \left(e^{\alpha_m/2} - e^{-\alpha_m/2}\right) \sum_{n=1}^{N} w_n^{(m)} I(h_m(\mathbf{x}_n) \neq t_n) + e^{-\alpha_m/2} \sum_{n=1}^{N} w_n^{(m)}$$

Slide source: B. Leibe Slides ML, 2015

AdaBoost – Minimizing Exponential Error

Minimize respect to $h_m(\mathbf{x})$: $\dfrac{\partial E}{\partial h_m(\mathbf{x}_n)} \stackrel{!}{=} 0$

$$E = \left(e^{\alpha_m/2} - e^{-\alpha_m/2}\right) \sum_{n=1}^{N} w_n^{(m)} I(h_m(\mathbf{x}_n) \neq t_n) + e^{-\alpha_m/2} \sum_{n=1}^{N} w_n^{(m)}$$

= const. **= const.**

⇒ This is equivalent to minimizing

$$J_m = \sum_{n=1}^{N} w_n^{(m)} I(h_m(\mathbf{x}) \neq t_n)$$

(our weighted error function from step 2a) of the algorithm)

⇒ *We're on the right track. Let's continue…*

Slide source: B. Leibe Slides ML, 2015

AdaBoost – Minimizing Exponential Error

Minimize respect to α_m: $\dfrac{\partial E}{\partial \alpha_m} \stackrel{!}{=} 0$

$$E = \left(e^{\alpha_m/2} - e^{-\alpha_m/2}\right) \sum_{n=1}^{N} w_n^{(m)} I(h_m(\mathbf{x}_n) \neq t_n) + e^{-\alpha_m/2} \sum_{n=1}^{N} w_n^{(m)}$$

$$\left(\frac{1}{2}e^{\alpha_m/2} + \frac{1}{2}e^{-\alpha_m/2}\right) \sum_{n=1}^{N} w_n^{(m)} I(h_m(\mathbf{x}_n) \neq t_n) \stackrel{!}{=} \frac{1}{2}e^{-\alpha_m/2} \sum_{n=1}^{N} w_n^{(m)}$$

weighted error $\epsilon_m := \dfrac{\sum_{n=1}^{N} w_n^{(m)} I(h_m(\mathbf{x}_n) \neq t_n)}{\sum_{n=1}^{N} w_n^{(m)}} = \dfrac{e^{-\alpha_m/2}}{e^{\alpha_m/2} + e^{-\alpha_m/2}}$

$$\epsilon_m = \frac{1}{e^{\alpha_m} + 1}$$

⇒ ***Update for the α coefficients:*** $\alpha_m = \ln\left\{\dfrac{1-\epsilon_m}{\epsilon_m}\right\}$

Slide source: B. Leibe Slides ML, 2015

AdaBoost – Minimizing Exponential Error

Remaining step: update the weights

Recall that

$$E = \sum_{n=1}^{N} \underbrace{w_n^{(m)} \exp\left\{-\frac{1}{2} t_n \alpha_m h_m(\mathbf{x}_n)\right\}}_{\text{This becomes } w_n^{(m+1)} \text{ in the next iteration.}}$$

Therefore

$$\begin{aligned} w_n^{(m+1)} &= w_n^{(m)} \exp\left\{-\frac{1}{2} t_n \alpha_m h_m(\mathbf{x}_n)\right\} \\ &= \ldots \\ &= w_n^{(m)} \exp\left\{\alpha_m I(h_m(\mathbf{x}_n) \neq t_n)\right\} \end{aligned}$$

=> Update for the weight coefficients

Slide source: B. Leibe Slides ML, 2015

AdaBoost – Final Algorithm

1. **Initialization: Set** $w_n^{(1)} = \frac{1}{N}$ **for** $n = 1,\ldots,N$

2. **For** $m = 1,\ldots,M$ **iterations**

   a) **Train a new weak classifier** $h_m(\mathbf{x})$ **using the current weighting coefficients** $\mathbf{W}^{(m)}$ **by minimizing the weighted error function**

$$J_m = \sum_{n=1}^{N} w_n^{(m)} I(h_m(\mathbf{x}) \neq t_n)$$

   b) **Estimate the weighted error of this classifier on** $\mathbf{X}$**:**

$$\epsilon_m = \frac{\sum_{n=1}^{N} w_n^{(m)} I(h_m(\mathbf{x}) \neq t_n)}{\sum_{n=1}^{N} w_n^{(m)}}$$

   c) **Calculate a weighting coefficient for** $h_m(\mathbf{x})$**:** $\alpha_m = \ln\left\{\frac{1-\epsilon_m}{\epsilon_m}\right\}$

   d) **Update the weighting coefficients:**

$$w_n^{(m+1)} = w_n^{(m)} \exp\{\alpha_m I(h_m(\mathbf{x}_n) \neq t_n)\}$$

Slide source: B. Leibe Slides ML, 2015

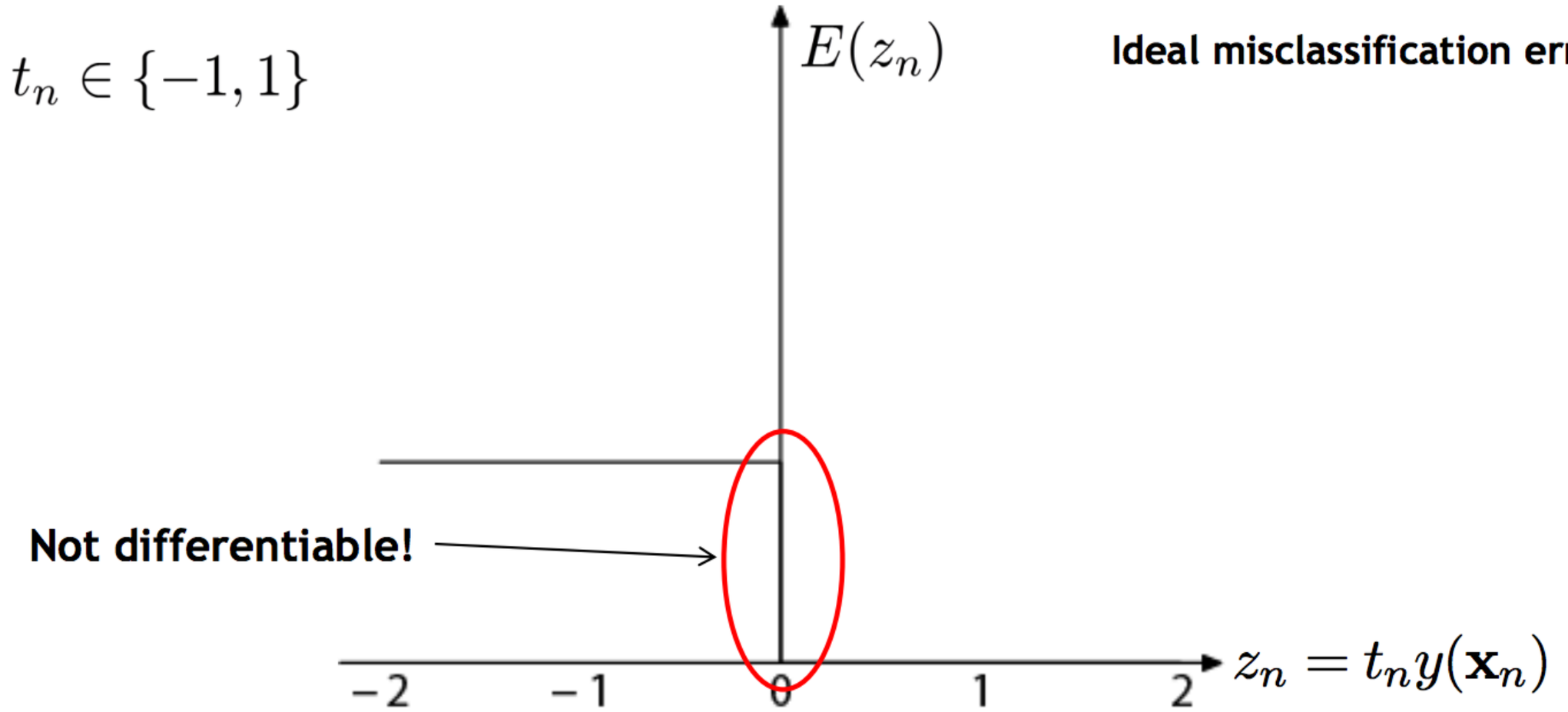
AdaBoost – Analysis

Results of this derivation

- We know that AdaBoost minimises an exponential error function in a sequential fashion.
- This allows us to analyse AdaBoost's behaviour in more detail.
- In particular, we can see how robust it is to outlier data points.

Slide source: B. Leibe Slides ML, 2015

Recap:Error Functions



Ideal misclassification error function (black)

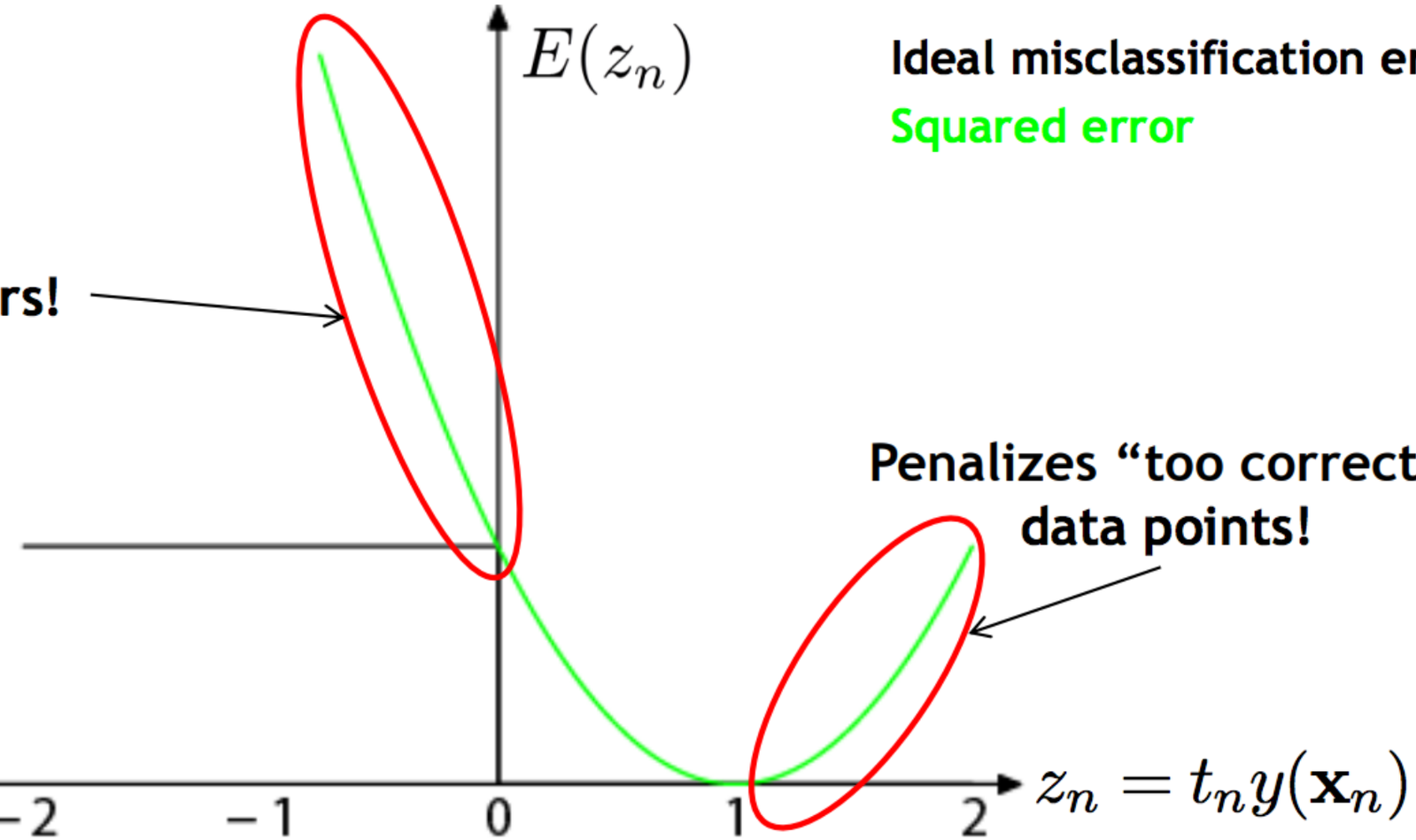
- This is what we want to approximate,
- Unfortunately, it is not differentiable.
- The gradient is zero for misclassified points.

  => We cannot minimize it by gradient descent.

Slide source: B. Leibe Slides ML, 2015

Recap:Error Functions

$t_n \in \{-1, 1\}$



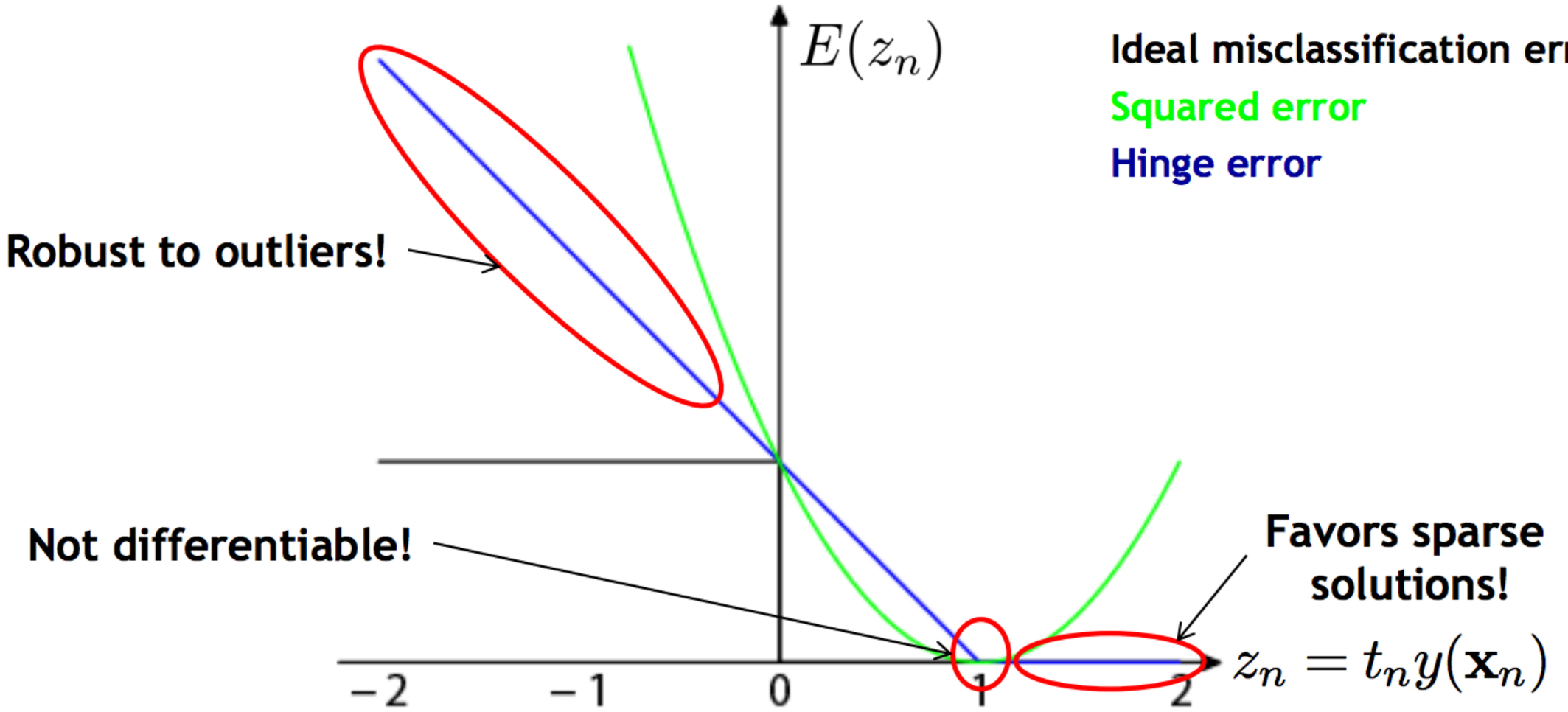
Squared error used in Least-Squares Classification

- Very popular, leads to closed-form solutions.
- However, sensitive to outliers due to squared penalty.
- Penalizes "too correct" data points

  => Generally does not lead to good classifiers.

Slide source: B. Leibe Slides ML, 2015

Recap:Error Functions

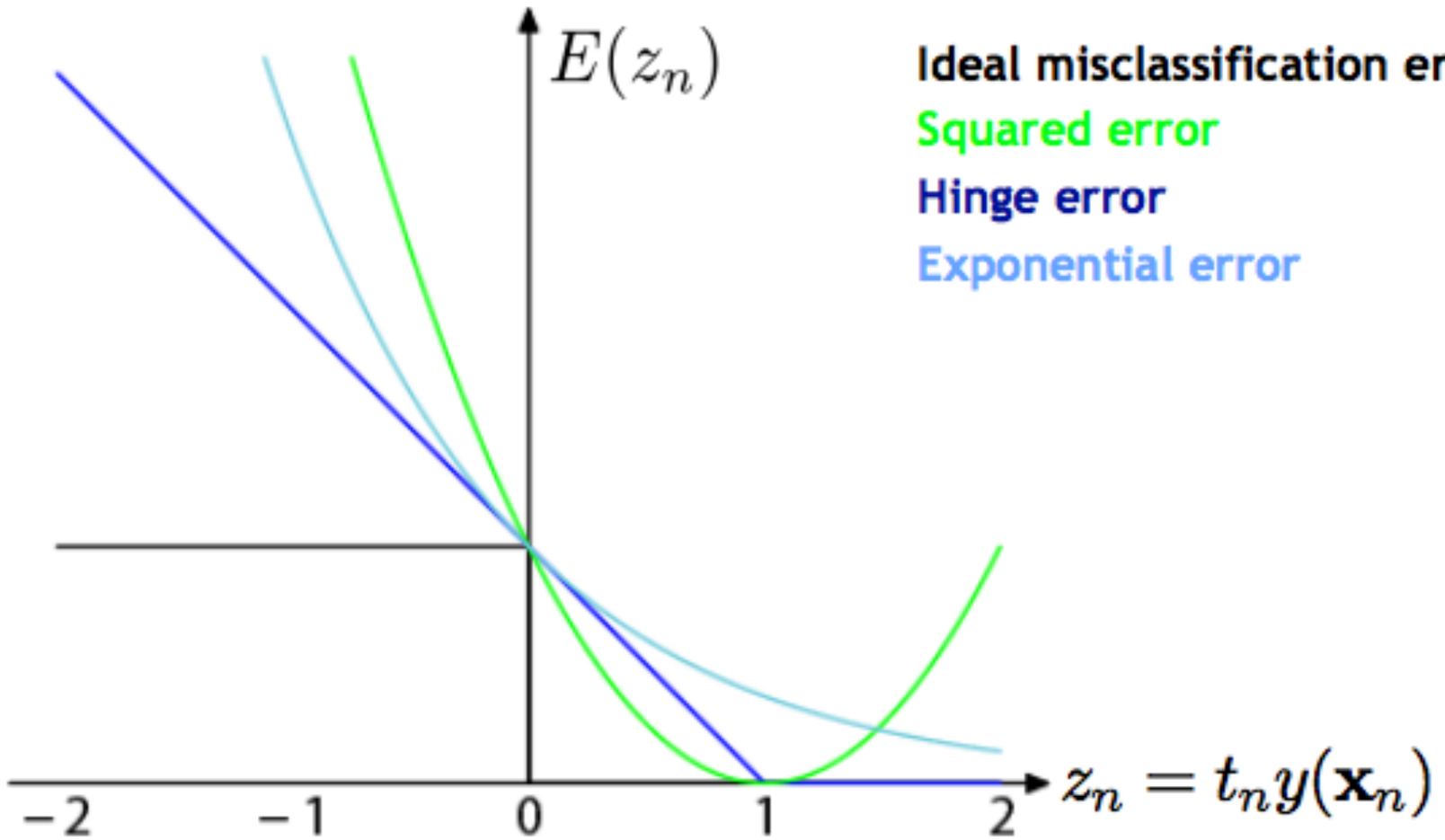


"Hinge error" used in SVMs

- Zero error for points outside the margin ($z_n > 1$) => sparsity.
- Linear penalty for misclassified points ($z_n < 1$) => robustness.
- Not differentiable around $z_n = 1$ => Cannot be optimized directly

Slide source: B. Leibe Slides ML, 2015

Discussion: AdaBoost Error Functions

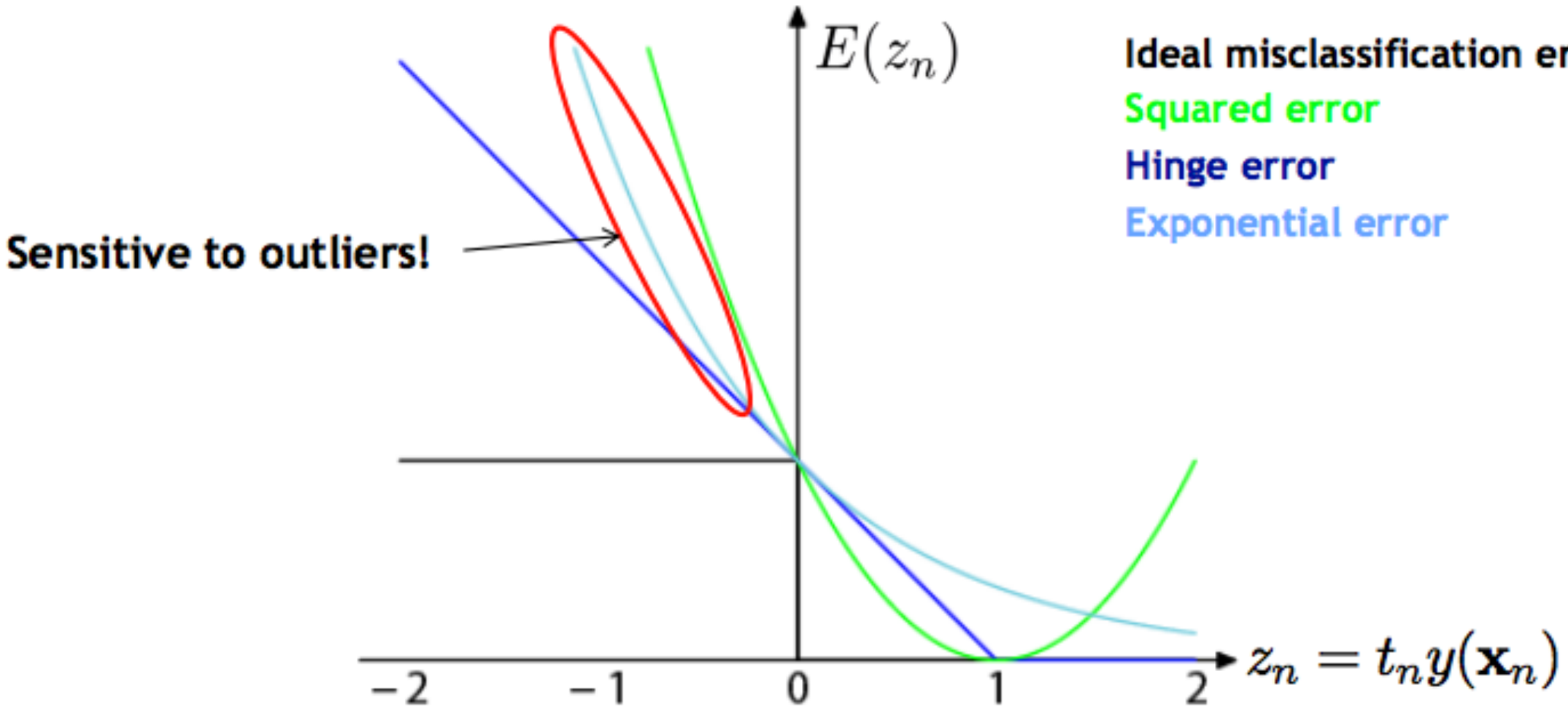


Exponential error used in AdaBoost

- Zero error for points outside the margin ($z_n > 1$) ⇒ sparsity.
- Sequential minimization leads to simple AdaBoost scheme
- Properties?

Slide source: B. Leibe Slides ML, 2015

Discussion: AdaBoost Error Functions



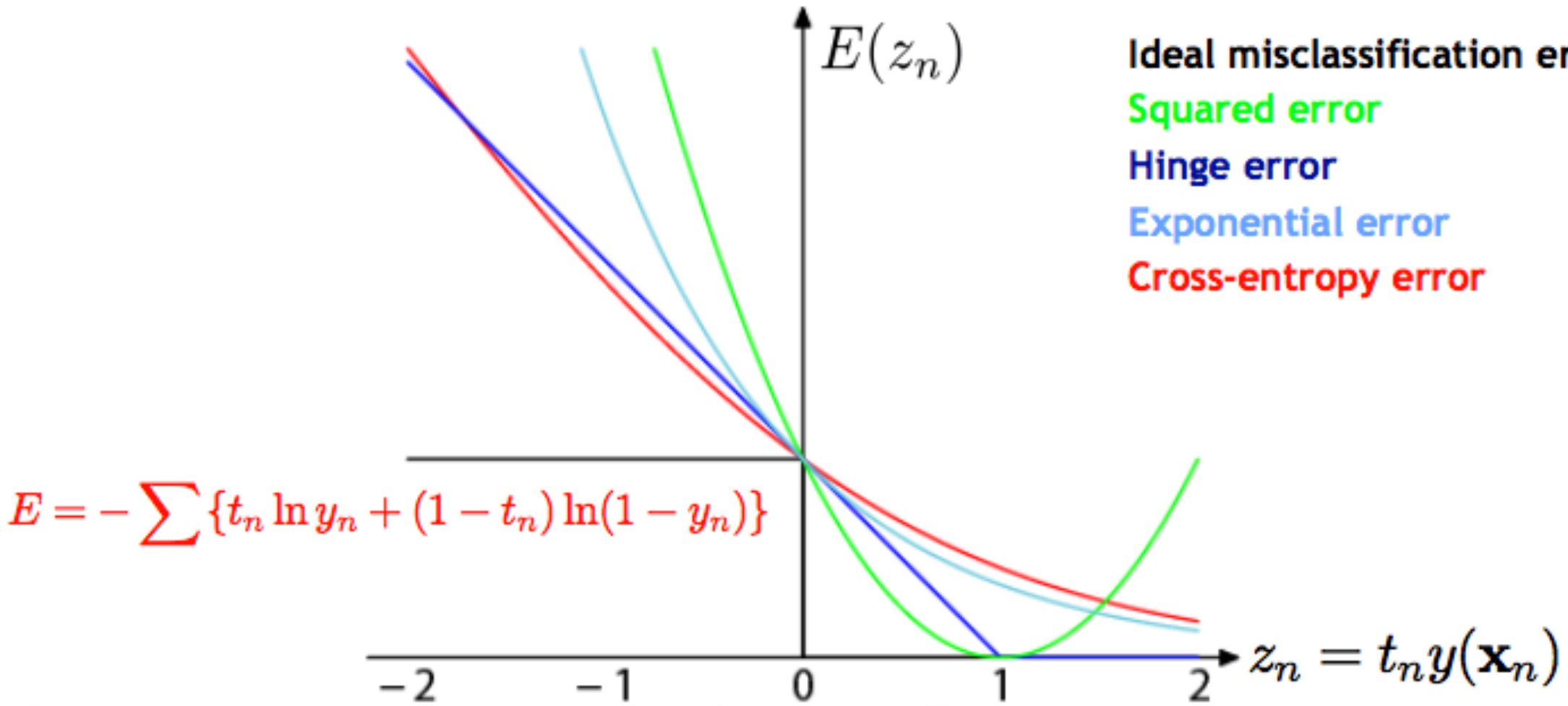
Exponential error used in AdaBoost

- No penalty for too correct data points, fast convergence.
- Disadvantage: exponential penalty for large negative values!

=> Less robust to outliers or misclassified data points!?

Slide source: B. Leibe Slides ML, 2015

Discussion: Other Possible Error Functions



'Cross-entropy error' used in Logistic Regression

- Similar to exponential error for z>0.
- Only grows linearly with large negative values of z

=> Make AdaBoost more robust by switching to this error function.

=> 'GentleBoost'

Slide source: B. Leibe Slides ML, 2015

Summary: AdaBoost

Properties

- Simple combination of multiple classifiers.
- Easy to implement
- Can be used with many different types of classifiers
  - None of them needs to be too good on its own
  - In fact, they only have to be slightly better than chance
- Commonly used in many areas
- Empirically good generalization capabilities

Limitations

- Original AdaBoost sensitive to misclassified training data points.
  - Because of exponential error function
  - Improvement by GentleBoost
- Single-class classifier
  - Multiclass extensions available

Slide source: B. Leibe Slides ML, 2015

Today's Topics

Ensembles of classifiers

- Bagging
- Bayesian Model Averaging

Combining Classifiers

- Stacking
- Bayesian model averaging
- Boosting

AdaBoost

- Intuition
- Algorithm
- Analysis
- Extensions

Applications

Slide source: B. Leibe Slides ML, 2015

Example Application: Face Detection

Frontal faces are a good example of a class where global appearance models + a sliding window detection approach fit well:

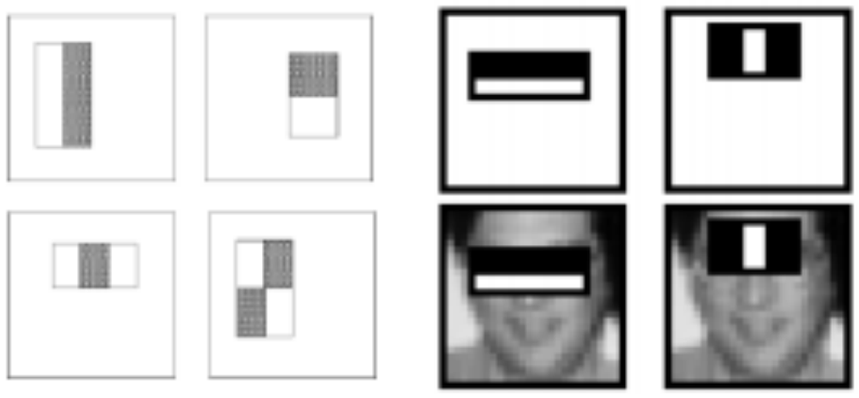
- Regular 2D structure
- Center of face almost shaped like a “patch”/windowt

Now we’ll take AdaBoost and see how the ViolaJones face detector works

Slide source: B. Leibe Slides ML, 2015

Feature extraction

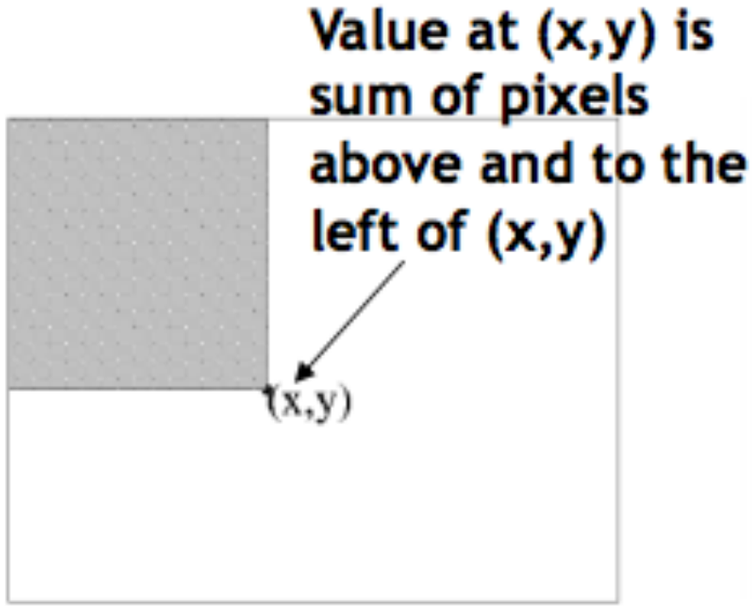
“Rectangular” filters



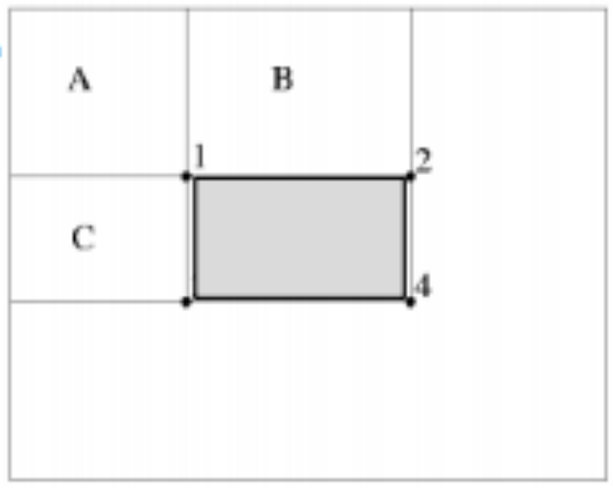
Feature output is difference between adjacent regions

Efficiently computable with integral image: any sum can be computed in constant time

Avoid scaling images → scale features directly for same cost



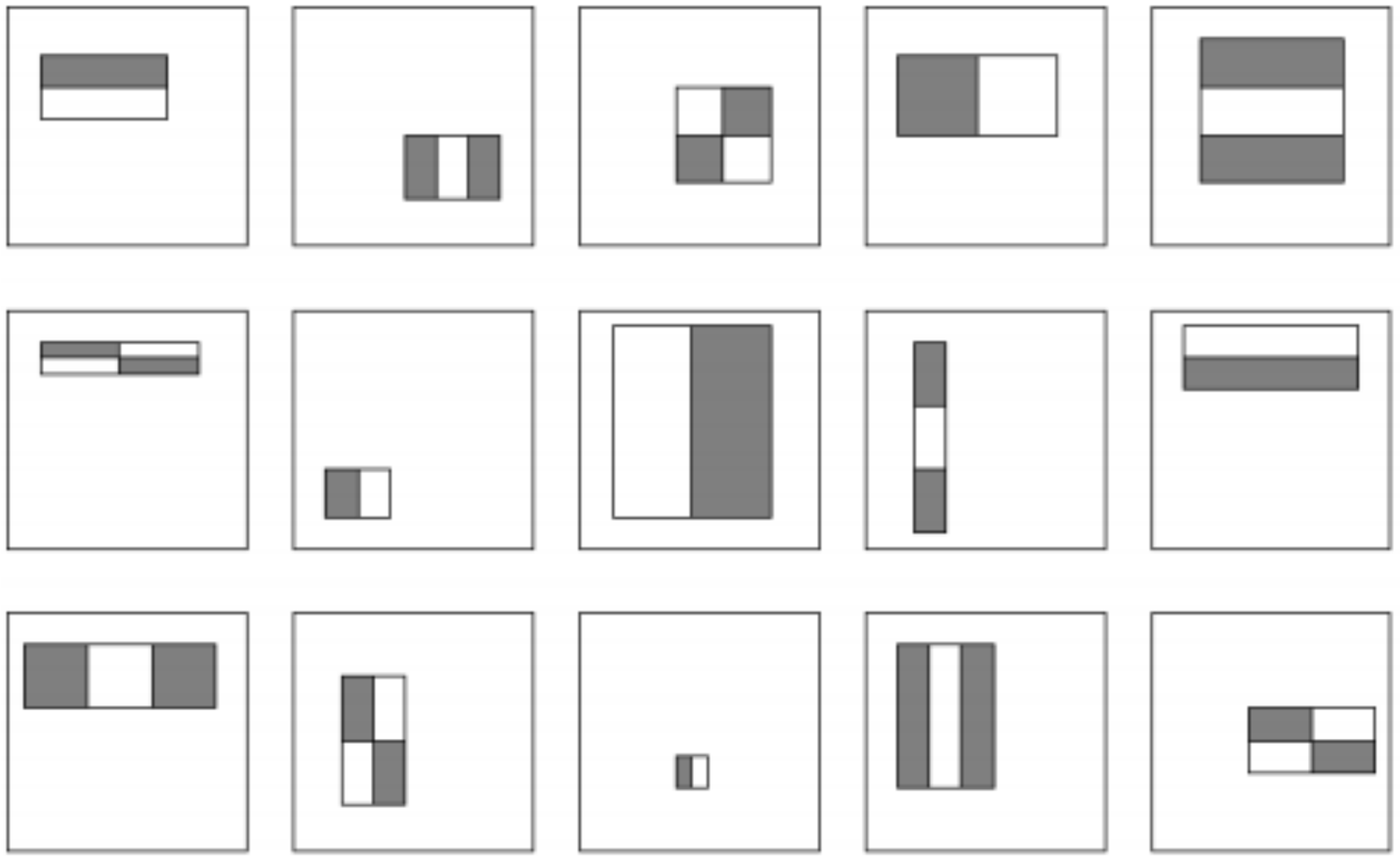
Integral image



$$D = 1+4-(2+3)$$
$$= A+(A+B+C+D)-(A+C+A+B)$$
$$= D$$

Slide source: B. Leibe Slides ML, 2015

Large Library of Filters



Considering all possible filter parameters: position, scale, and type:

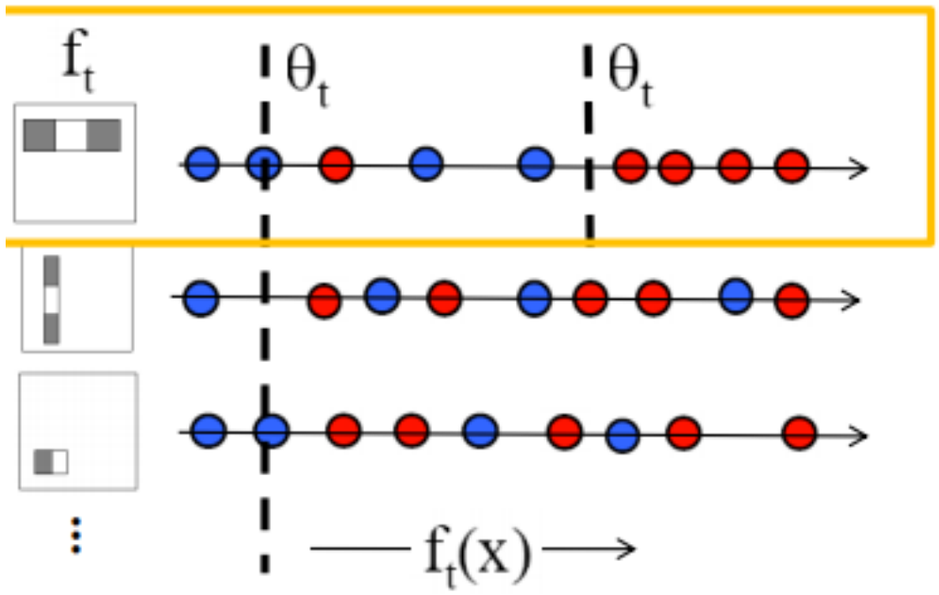
180,000+ possible features associated with each 24 x 24 window

Use AdaBoost both to select the informative features and to form the classifier

Slide source: B. Leibe Slides ML, 2015

AdaBoost for Feature+Classifier Selection

Want to select the single rectangle feature and threshold that best separates positive (faces) and negative (nonfaces) training examples, in terms of *weighted* error



Outputs of a possible rectangle feature on a face and non-face

Resulting weak classifier:

$$h_t(x) = \begin{cases} +1 & \text{if } f_t(x) > \theta_t \\ -1 & \text{otherwise} \end{cases}$$

For next round, reweight the examples according to errors, choose another filter/ threshold combo.

Slide source: B. Leibe Slides ML, 2015

AdaBoost for Efficient Feature Selection

Image features = weak classifiers

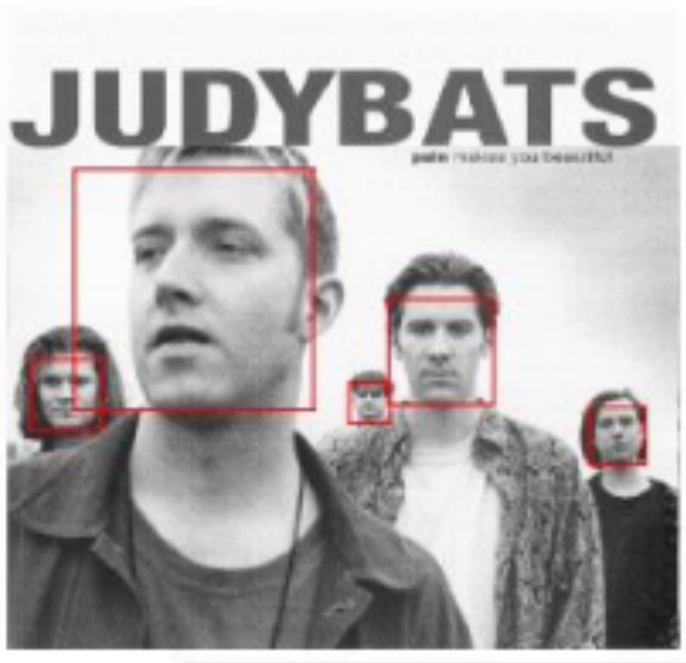
For each round of boosting:

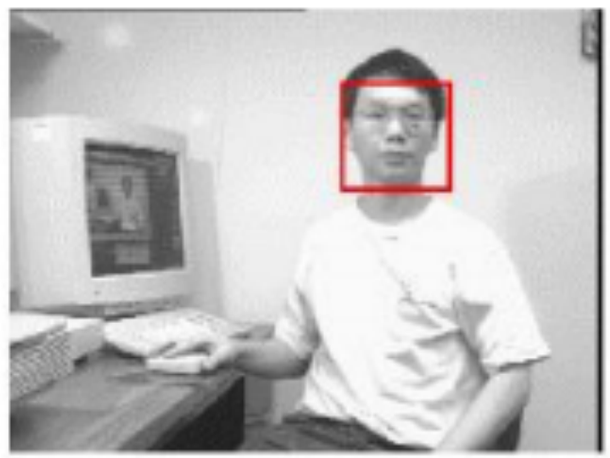
- Evaluate each rectangle filter on each example
- Sort examples by filter values
- Select best threshold for each filter (min error)
  - Sorted list can be quickly scanned for the optimal threshold
- Select best filter/threshold combination
- Weight on this features is a simple function of error rate
- Reweight examples

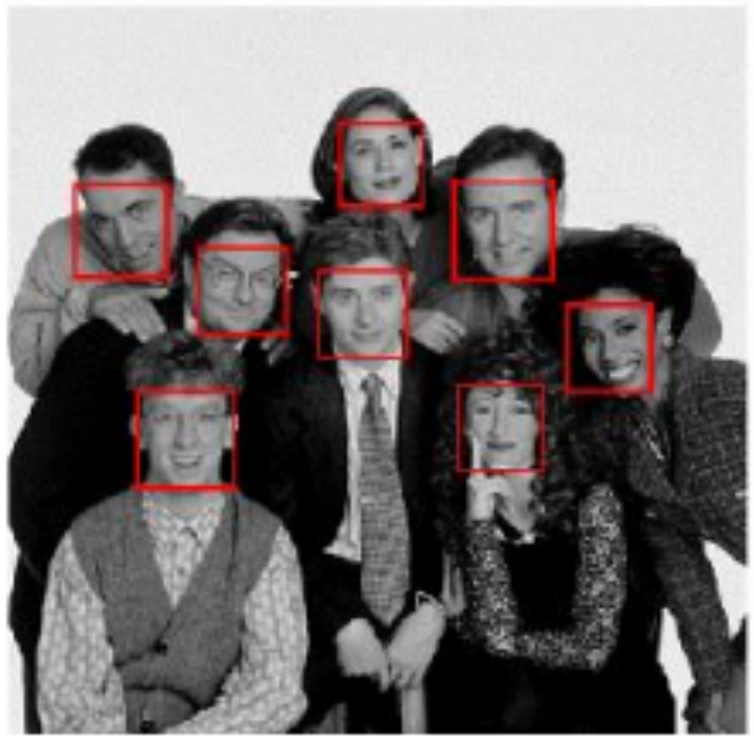
P. Viola, M. Jones, Robust Real-Time Face Detection, IJCV, Vol. 57(2), 2004.(first version appeared at CVPR 2001)

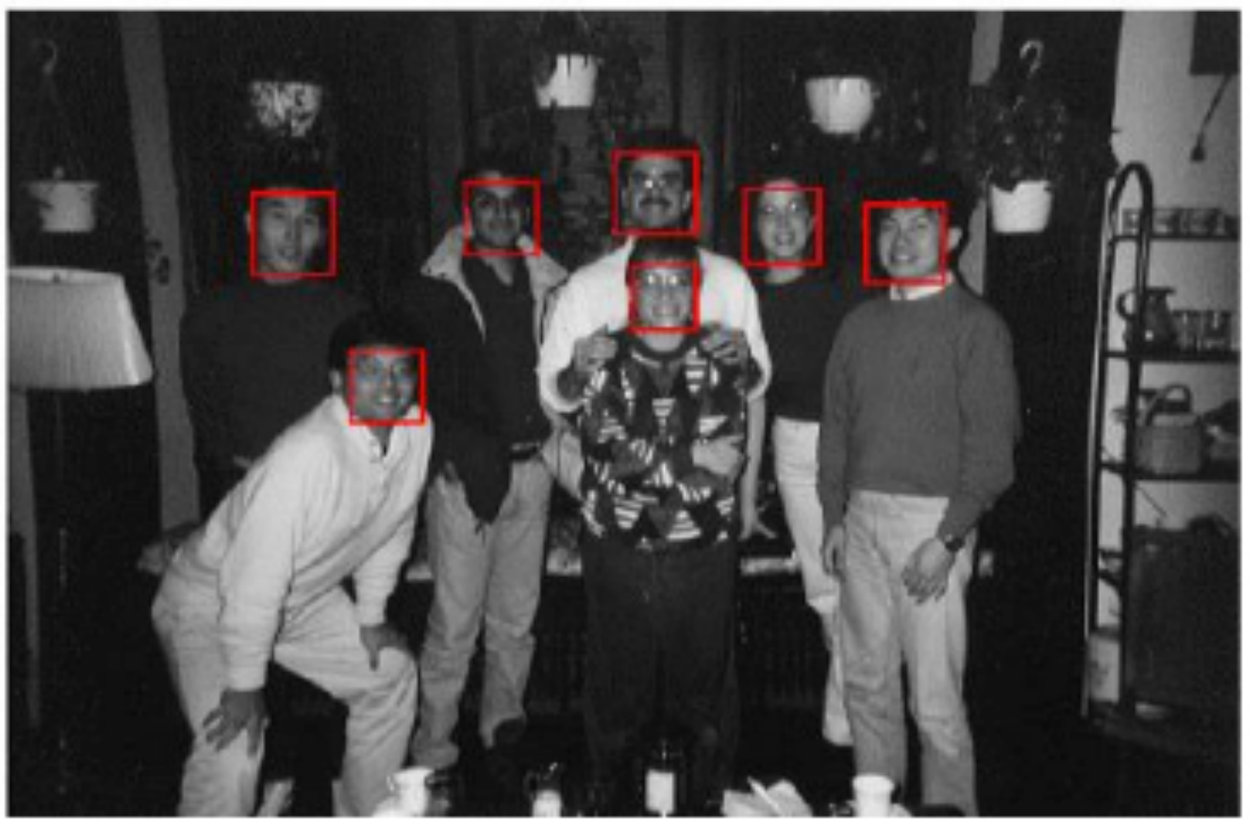
Slide source: B. Leibe Slides ML, 2015

Viola-Jones Face Detector: Results



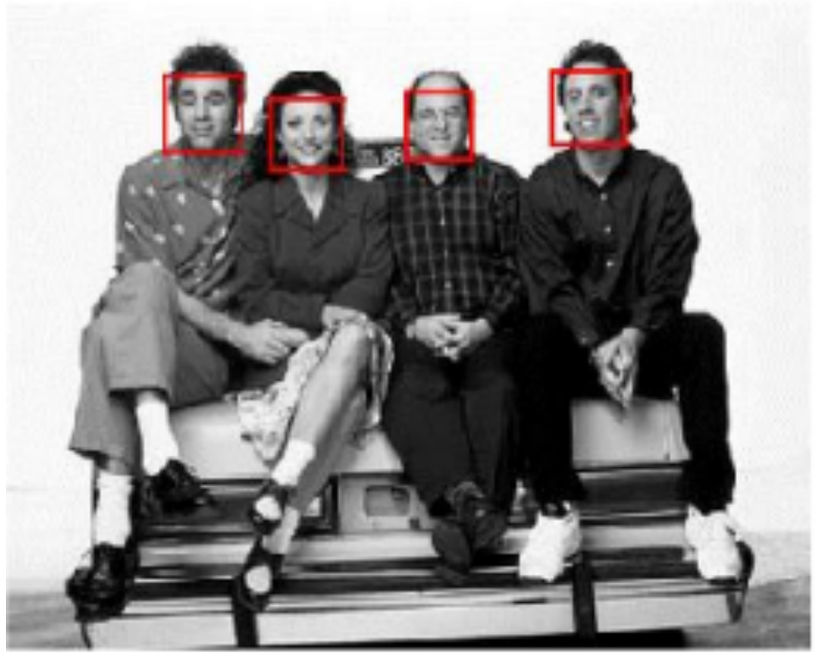


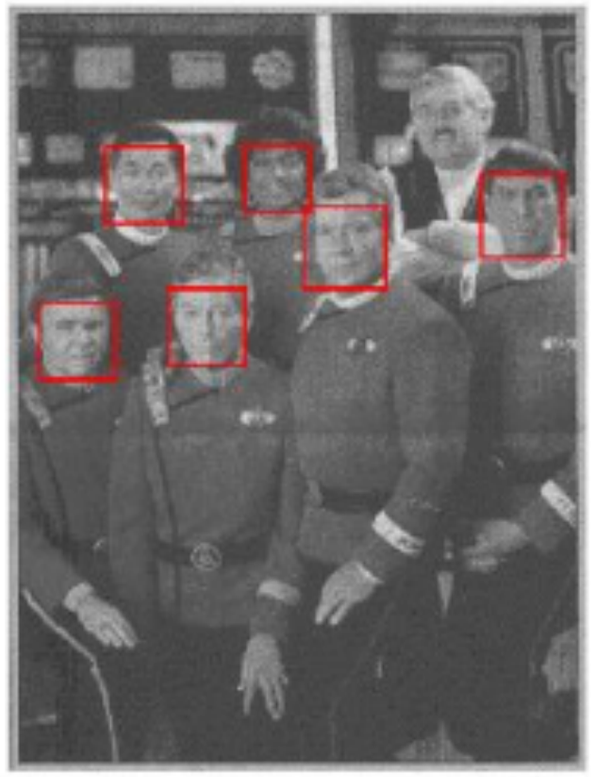


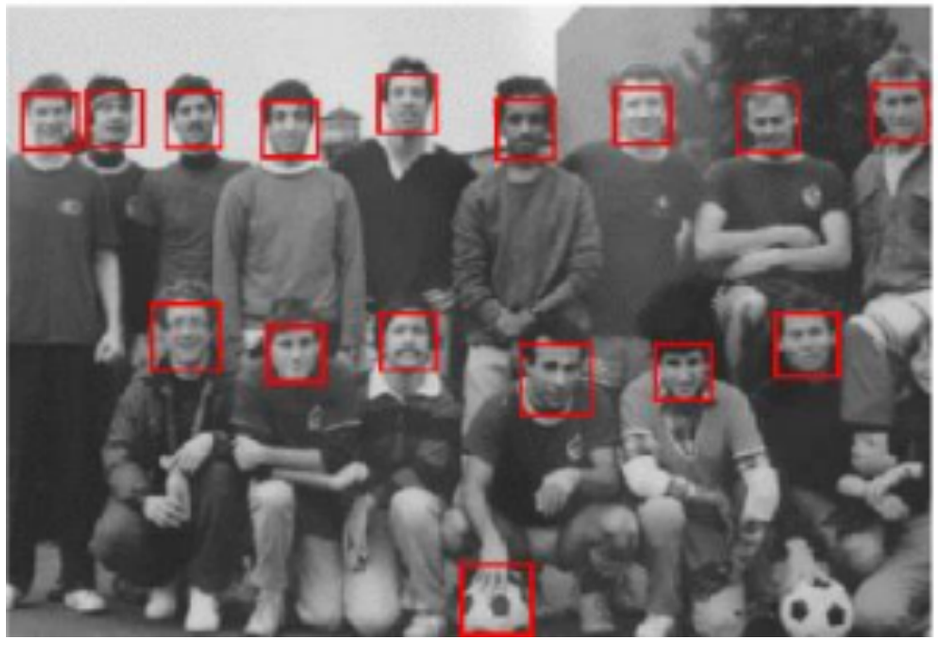


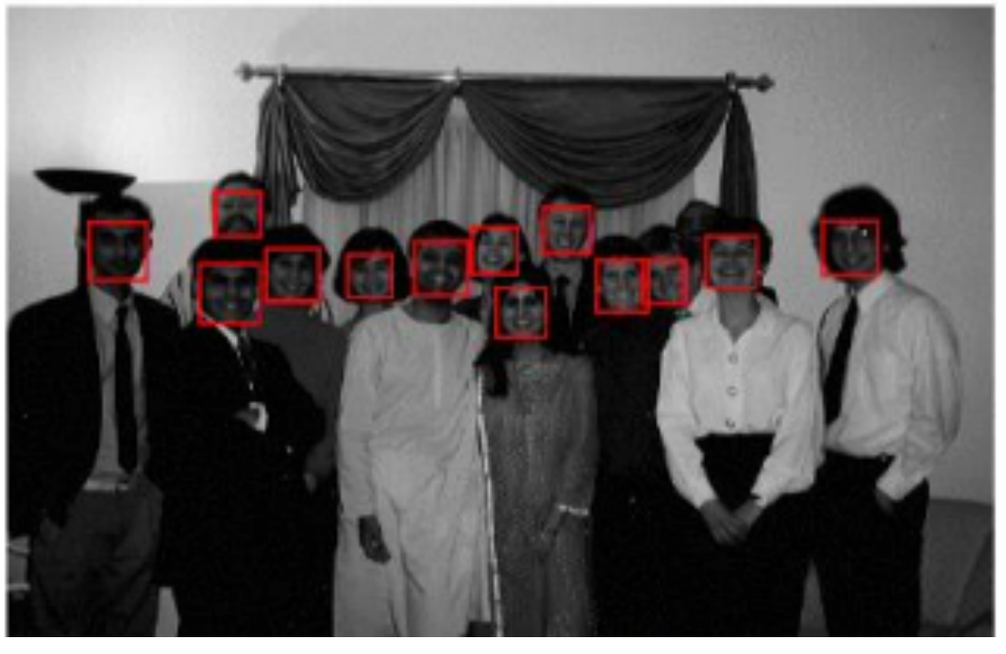
Slide source: B. Leibe Slides ML, 2015

Viola-Jones Face Detector: Results









Slide source: B. Leibe Slides ML, 2015

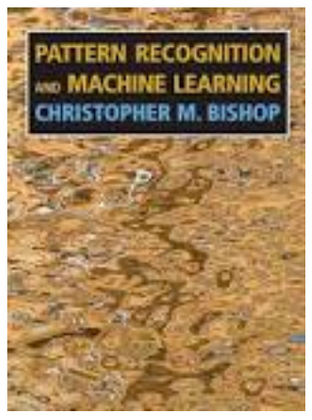
Viola-Jones Face Detector: Results

Slide source: B. Leibe Slides ML, 2015

Readings

Bishop's book

Chapter 14.1 - 14.3 9 (More information on Classifier Combination and Boosting)



Additional information

A more in-depth discussion of the statistical interpretation of AdaBoost is available in the following paper:

J. Friedman, T. Hastie, R. Tibshirani, Additive Logistic Regression: a Statistical View of Boosting, The Annals of Statistics, Vol. 38(2), pages 337-374, 2000

Slide is taken from B. Leibe, ML, 2017