Machine Learning

Ensembles of classifiers - Lecture VII

Videos for This Lecture

- Repetition Video (Part 0)
- Ensembles of classifiers (Part 1)
- Combining Classifiers (Part 2)
- AdaBoost (Part 3)

Recap: Learning Discriminant Functions

Basic idea

- Directly encode decision boundary
- Minimize misclassification probability directly

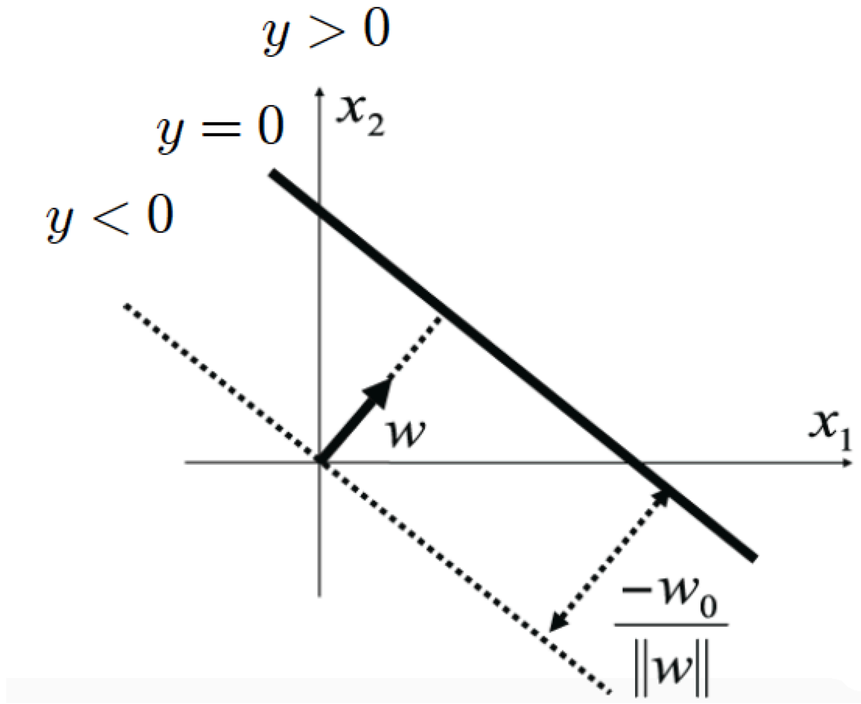
Linear discriminant functions

$$y(\mathrm{x}) = \mathrm{w}^T \mathrm{x} + w_0$$

weight vector

"bias" (=threshold)

- w, w_0 define a hyperplane in $\mathbb{R}^D$



- If a data set can be perfectly classified by a linear discriminant, then we call it **linearly separable**.

Slide source: B. Leibe Slides ML, 2015

Recap: Extension to Nonlinear Basis Functions

Generalization

- Transform vector x with M nonlinear basis functions $\phi_j(\mathrm{x})$:

$$y_k(\mathrm{x}) = \sum_{j=1}^{M} w_{kj}\phi_j(\mathrm{x}) + w_{k0}$$

Advantages

- Allow non-linear decision boundaries.
- By choosing the right ϕ_j, every continuous function can (in principle) be approximated with arbitrary accuracy.

Disadvantages

- The error function can in general no longer be minimised in closed form.

=> Minimization with Gradient Descent

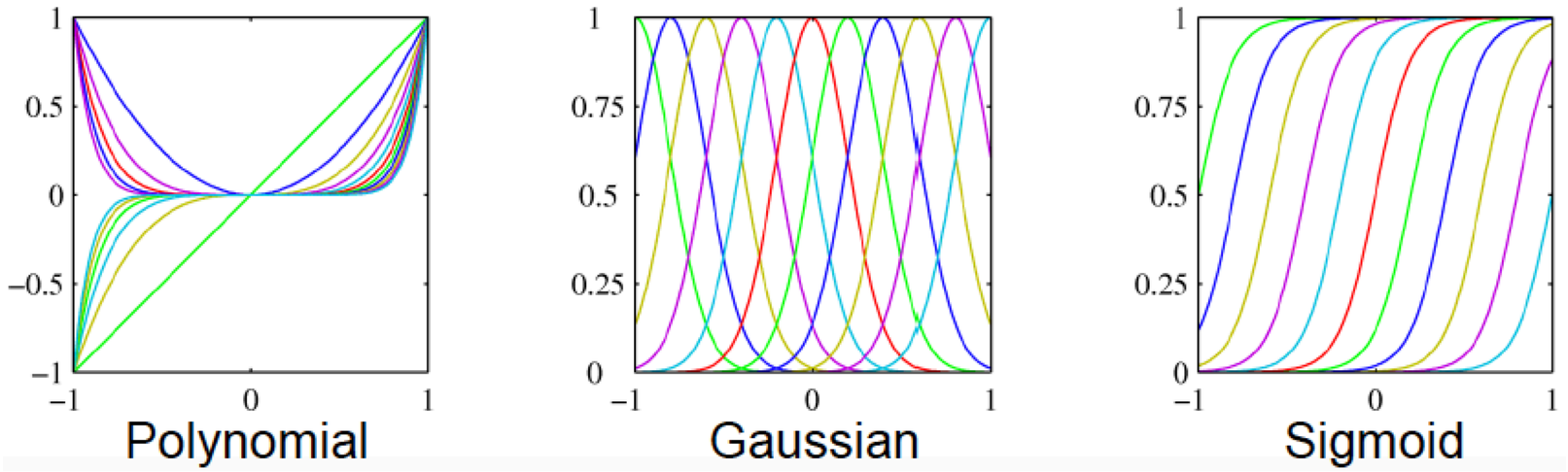
Slide source: B. Leibe Slides ML, 2015

Recap: Basis Functions

Generally, we consider models of the following form

$$y_k(\mathrm{x}) = \sum_{j=0}^{M} w_{kj}\phi_j(\mathrm{x}) = \mathrm{w}^T\phi(\mathrm{x})$$

- where $\phi_j(\mathrm{x})$ are known as *basis functions*.
- Th the simplest case, we use linear basis functions: $\phi_d(\mathrm{x}) = x_d$



Slide source: B. Leibe Slides ML, 2015

Recap: Gradient Descent

Iterative minimisation

$$\mathrm{w}^{(\tau+1)} = \mathrm{w}^{(\tau)} - \eta \nabla E(\mathrm{w})$$

- Start with an initial guess for the parameter values $w_{kj}^{(0)}$
- Move towards a (local) minimum by following the gradient.

Basic strategies

- Batch learning $$w_{kj}^{(\tau+1)} = w_{kj}^{(\tau)} - \eta \left. \frac{\partial E(\mathbf{w})}{\partial w_{kj}} \right|_{\mathbf{w}^{(\tau)}}$$

- Sequential updating learning $$w_{kj}^{(\tau+1)} = w_{kj}^{(\tau)} - \eta \left. \frac{\partial E_n(\mathbf{w})}{\partial w_{kj}} \right|_{\mathbf{w}^{(\tau)}}$$ where $$E(\mathbf{w}) = \sum_{n=1}^{N} E_n(\mathbf{w})$$

Recap: Gradient Descent

Example: Quadratic error function

$$E(\mathbf{w}) = \sum_{n=1}^{N} \left(y(\mathbf{x}_n; \mathbf{w}) - \mathbf{t}_n \right)^2$$

Sequential updating leads to **delta rule** (=**LMS**)

$$\begin{aligned} w_{kj}^{(\tau+1)} &= w_{kj}^{(\tau)} - \eta \left(y_k(\mathbf{x}_n; \mathbf{w}) - t_{kn} \right) \phi_j(\mathbf{x}_n) \\ &= w_{kj}^{(\tau)} - \eta \delta_{kn} \phi_j(\mathbf{x}_n) \end{aligned}$$

where

$$\delta_{kn} = y_k(\mathbf{x}_n; \mathbf{w}) - t_{kn}$$

- Simply feed back the input data point, weighted by the classification error.

Recap: Gradient Descent

Cases with differentiable, non-linear activation function

$$y_k(\mathbf{x}) = g(a_k) = g\left(\sum_{j=0}^{M} w_{ki}\phi_j(\mathbf{x}_n)\right)$$

Gradient descent (again with quadratic error function)

$$\frac{\partial E_n(\mathbf{w})}{\partial w_{kj}} = \frac{\partial g(a_k)}{\partial w_{kj}} \left(y_k(\mathbf{x}_n; \mathbf{w}) - t_{kn}\right) \phi_j(\mathbf{x}_n)$$

$$w_{kj}^{(\tau+1)} = w_{kj}^{(\tau)} - \eta\delta_{kn}\phi_j(\mathbf{x}_n)$$

$$\delta_{kn} = \frac{\partial g(a_k)}{\partial w_{kj}} \left(y_k(\mathbf{x}_n; \mathbf{w}) - t_{kn}\right)$$

Recap: Probabilistic Discriminative Models

Consider models of the form

$$p(C_1|\phi) = y(\phi) = \sigma(\mathrm{w}^T\phi)$$
$$p(C_2|\phi) = 1 - p(C_1|\phi)$$

This model is called **logistic regression**.

Properties

- Probabilistic interpretation
- But discriminative method: only focus on decision hyperplane
- Advantageous for high-dimensional spaces, requires less parameters than explicitly modelling likelihood and prior

Recap: Logistic Regression

Let's consider a data set $\{\phi_n, t_n\}$ with $n = 1, ..., N$,
where $\phi_n = \phi(\mathrm{x}_n)$ and $t_n \in \{0, 1\}, \mathrm{t} = (t_1, ..., t_N)^T$

With $y_n = p(C_1|\phi_n)$, we can write likelihood as

$$p(\mathrm{t}|\mathrm{w}) = \prod_{n=1}^{N} y_n^{t_n}\{1 - y_n\}^{1-t_n}$$

Define the error function as the negative log-likelihood

$$\begin{aligned} E(\mathrm{w}) &= -\ln p(\mathrm{t}|\mathrm{w}) \\ &= -\sum_{n=1}^{N}\{t_n \ln y_n + (1 - t_n)\ln(1 - y_n)\} \end{aligned}$$

- This is the so-called **cross-entropy error function**.

Recap: Gradient Descent

Gradient for logistic regression

$$\nabla E(\mathrm{w}) = \sum_{n=1}^{N} (y_n - t_n)\phi_n$$

This is the same result as for the Delta (=LMS) rule

$$w_{kj}^{(\tau+1)} = w_{kj}^{(\tau)} - \eta(y_k(\mathbf{x}_n; \mathbf{w}) - t_{kn})\phi_j(\mathbf{x}_n)$$

Second-order Newton-Raphson gradient descent scheme

$$\mathrm{w}^{(\tau+1)} = \mathrm{w}^{(\tau)} - \mathrm{H}^{-1}\nabla E(\mathrm{w})$$

where $\mathrm{H} = \nabla\nabla E(\mathrm{w})$ is the Hessian matrix, i.e. the matrix of second derivatives.

$$\mathbf{H} = \nabla\nabla E(\mathbf{w}) = \sum_{n=1}^{N} y_n(1 - y_n)\boldsymbol{\phi}_n\boldsymbol{\phi}_n^T = \boldsymbol{\Phi}^T\mathbf{R}\boldsymbol{\Phi}$$

Part 1, Video Ensembles_p1

Ensembles of classifiers

- Bagging
- Bayesian Model Averaging

Course Outline

Basic Concepts

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

Classification Approaches

- Linear Discriminants
- Ensemble Methods and Boosting
- Randomized Trees, Forest

Reinforcement Learning

- Classical Reinforcement Learning
- Deep Reinforcement Learning

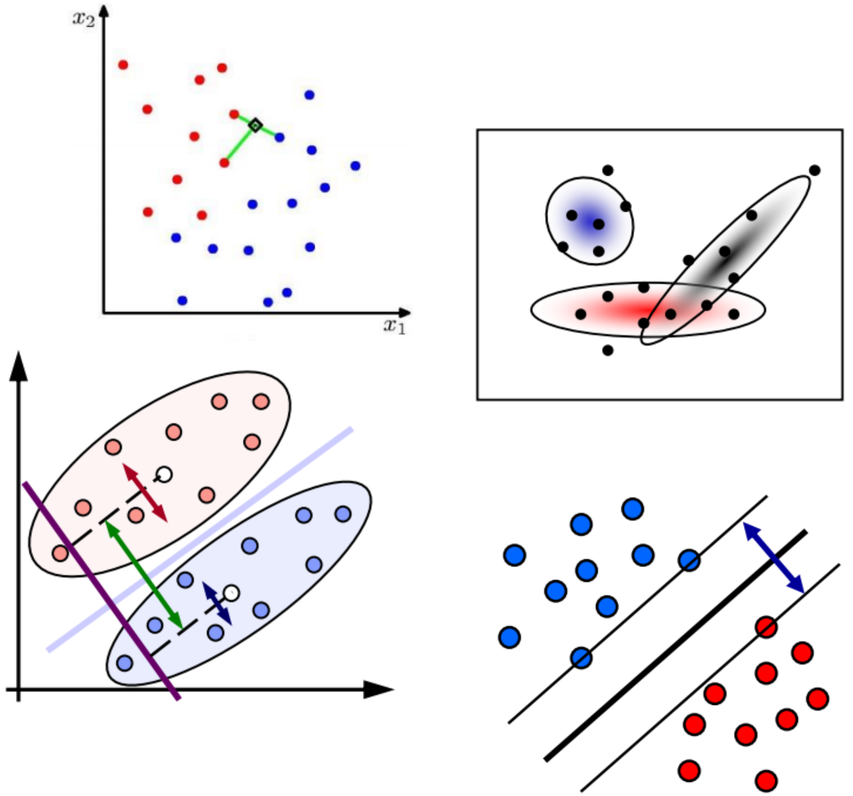
Deep Learning

- Foundations
- Optimization

So Far…

Study:

- k-NN
- Bayes classifiers
- Linear discriminants
- Logistic Regression



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

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

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

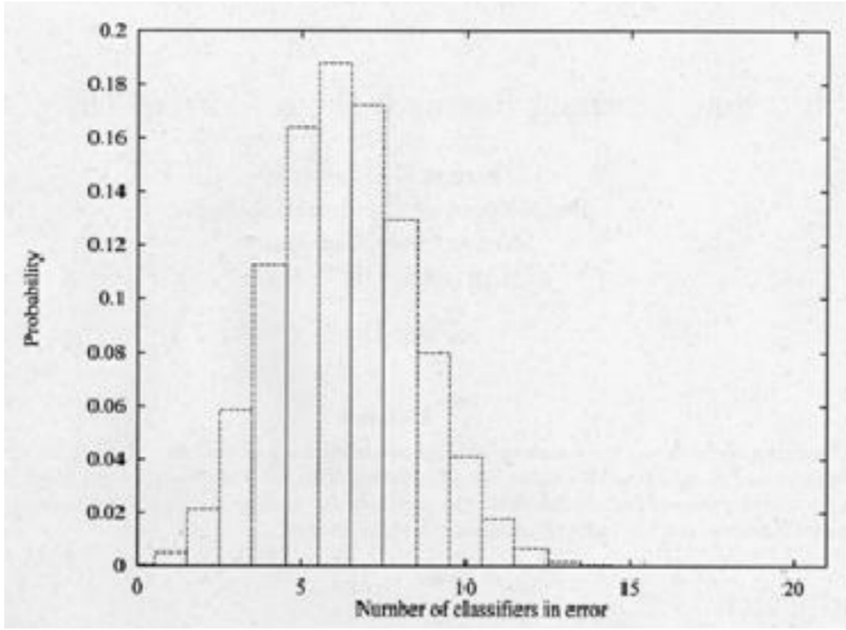
Ensembles of Classifiers

Example

- K classifiers with error probability ***p*=**0.3
- Probability that exactly L classifiers make an error:

$$p^L(1-p)^{K-L}$$

- The probability that 11 or more classifiers make an error is 0.026



Slide source: B. Leibe Slides ML, 2015

Today's Topics

Ensembles of classifiers

Constructing Ensembles

- Bagging
- Cross-validation

Combining Classifiers

- Stacking
- Bayesian model averaging
- Boosting

AdaBoost

- Intuition
- Algorithm
- Analysis
- Extensions

Methods for obtaining a set of classifiers

Methods for combining different classifiers

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

Part 2, Video Ensembles_p2

Combining Classifiers

- Stacking
- Bayesian model averaging
- Boosting

Today's Topics

Ensembles of classifiers

- Bagging
- Cross-validation

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)

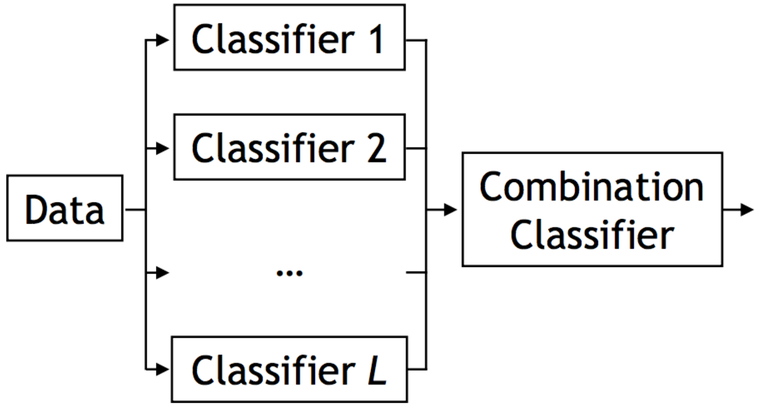
Find a meta-classifier that takes as input the output of the L first-level classifiers.

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



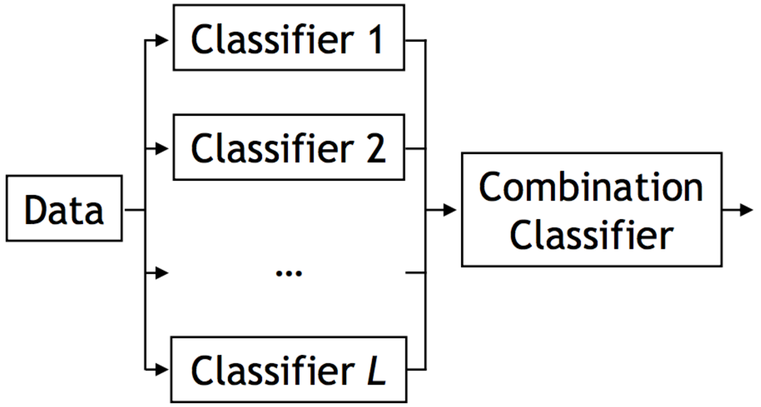
Slide source: B. Leibe Slides ML, 2015

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:

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

Slide source: B. Leibe Slides ML, 2015

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:

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

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:

$$\mathbb{E}_{COM} = \mathbb{E}_{\mathbf{x}}\left[\left\{\frac{1}{M}\sum_{m=1}^{M}y_m(\mathbf{x}) - h(\mathbf{x})\right\}^2\right] = \mathbb{E}_{\mathbf{x}}\left[\left\{\frac{1}{M}\sum_{m=1}^{M}\epsilon_m(\mathbf{x})\right\}^2\right]$$

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

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.

Slide source: B. Leibe Slides ML, 2015

Part 3, Video Ensembles_p3

AdaBoost

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, re-weight 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

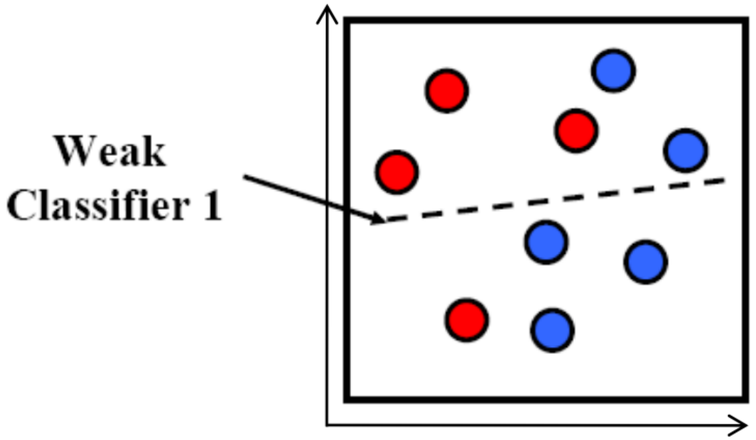
AdaBoost:

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



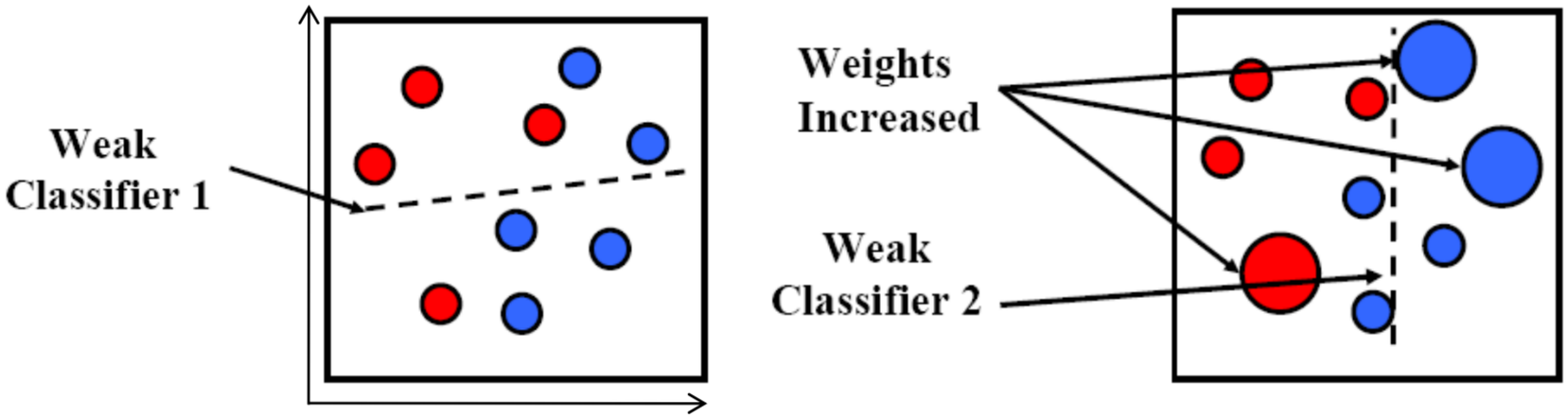
Consider a 2D feature space with **positive** and **negative** examples.

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.

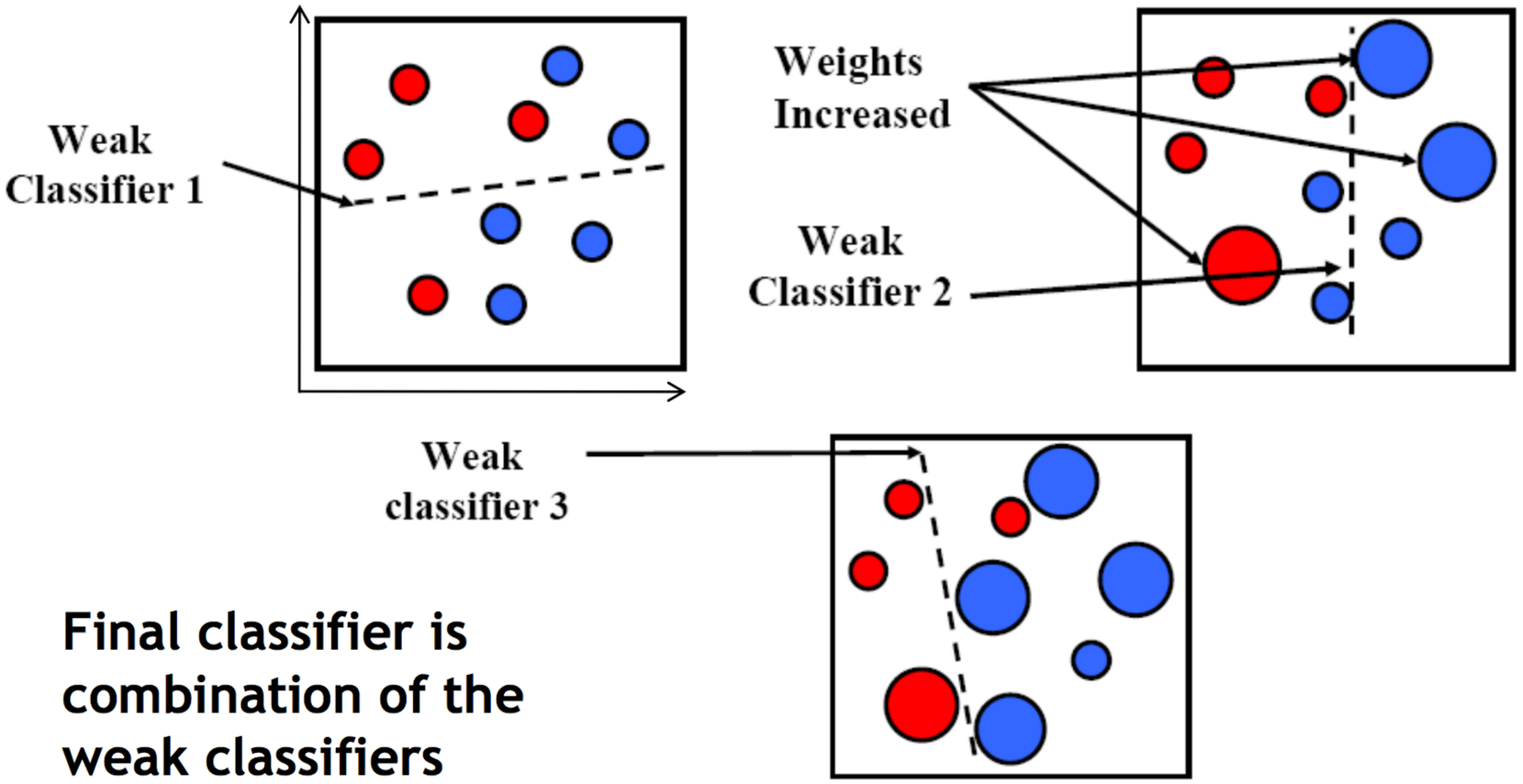
Slide source: B. Leibe Slides ML, 2015

AdaBoost: Intuition



Slide source: B. Leibe Slides ML, 2015

AdaBoost: Intuition



Final classifier is combination of the weak classifiers

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

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

$$I(A) = \begin{cases} 1 & \text{if } A \text{ is true} \\ 0 & \text{else} \end{cases}$$

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.

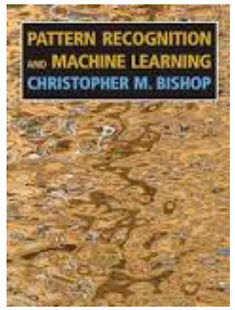
Different explanation (Friedman, Hastie, Tibshirani, 2000)

- 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

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

You should be able to answer …

- Which methods for obtaining a set of classifiers do you know? Explain the idea behind them how do they work.
- Which Methods for combining different classifiers do you know? Explain the idea behind them how do they work.
- What is the difference between model combination and Bayesian Model Averaging.
- What is the problem with the formula for average error of committee, why we cannot use it?
- AdaBoost: Main idea, components, Intuition