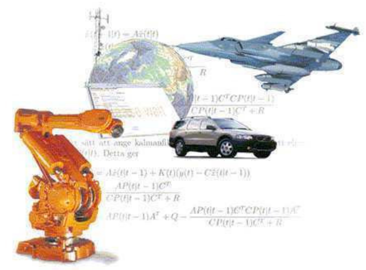
Machine Learning, Lecture 8
Boosting and introduction to graphical models

***"Boosting is one of the most powerful learning ideas introduced in the last twenty years."* HTF**



Thomas Schön

Division of Automatic Control
Linköping University
Linköping, Sweden.

Email: schon@isy.liu.se,
Phone: 013 - 281373,
Office: House B, Entrance 27.

Outline lecture 8

1. Summary of lecture 7
2. Time for projects!
3. Boosting
   - Motivation
   - The AdaBoost algorithm
   - An interpretation of AdaBoost as additive modelling
   - An illustrative example
   - Two recent applications of AdaBoost
4. Introducing graphical models
   - Motivation and some basic facts
   - An example - linear dynamical models

(Chapter 14.3 and Chapter 8.1)

Summary of lecture 7 (I/II)

Variational inference is a type of approximate Bayesian inference where factorial approximations like

$$p(Z|X) \approx q(Z) = \prod_i q_i(Z_i)$$

are used for the posterior.

The Kullback-Leibler (KL) distance is used to measure the distance and hence to find an optimization problem.

Variational Bayes (VB) is a form of variational inference where $\mathrm{KL}(q||p)$ is used for the optimization. We fix all but one of the factors and optimize as follows,

$$\widehat{q}_j(Z_j) = \arg\min_{q_j} \left(q_j(Z_j) \prod_{i \neq j} \widehat{q}_i(Z_i) \middle|\middle| p(X, Z) \right)$$

Summary of lecture 7 (II/II)

Expectation Propagation (EP) is another form of variational inference where $\mathrm{KL}(p||q)$ is used for the optimization. We fix all but one of the factors and optimize.

$$\widehat{q}_j(Z_j) = \arg\min_{q_j} \left(f_j(Z_j) \prod_{i \neq j} \widehat{q}_i(Z_i) \middle|\middle| q_j(Z_j) \prod_{i \neq j} \widehat{q}_i(Z_i) \right)$$

where f_j is the correct factor in $p(X, Z)$.

Time for projects!

An opportunity to learn more and try new ideas!

Three requirements:

1. You must chose something you think is **fun**.
2. Your project must be centered around a data set containing real measurements.
3. Your topic must fit the course.

Date	Action
Mar. 20	Project proposals are due
Mar. 22	Project proposal presentations
Apr. 19	Final reports are due
Apr. 24	Final project presentations

All details are available from the course website.

The AdaBoost algorithm

Algorithm 1 AdaBoost (Adaptive boosting)

1. Initialize training data weights $w_n^{(1)} = 1/N, n = 1, \dots, N$.
2. **For** $m = 1 : M$ **do:**
   (a) Learn a classifier by minimizing the weighted cost function
   $$J_m = \sum_{n=1}^{N} w_n^{(m)} I\left(y_m(x_n) \neq t_n\right).$$
   (b) Compute $\epsilon_m = \sum_{n=1}^{N} w_n^{(m)} I\left(y_m(x_n) \neq t_n\right) / \sum_{n=1}^{N} w_n^{(m)}$.
   (c) Compute $\alpha_m = \ln\left((1-\epsilon_m)/\epsilon_m\right)$.
   (d) Compute new weights
   $$w_n^{(m+1)} = w_n^{(m)} \exp\left(\alpha_m I(y_m(x_n) \neq t_n)\right).$$
3. Result: $Y_M(x) = \text{sign}\left(\sum_{m=1}^{M} \alpha_m y_m(x)\right)$.

Some comments on the AdaBoost algorithm

1. I denotes the indicator function,
$$I(y_m(x_n) \neq t_n) = \begin{cases} 1 & y_m(x_n) \neq t_n \\ 0 & \text{otherwise} \end{cases}$$
2. The cost function J_m minimized in step 2a of the algorithm makes use of the weights $w_n^{(m)}$ to assign a **greater cost** to the training samples that were **previously misclassified**.
3. In step 2b, ϵ_m represents the weighted measure of error rates for each of the weak classifiers. Via α_m in step 2c, this is used to find the final combination of the weak classifiers, resulting in a strong classifier.
4. In step 2d the weights $w_n^{(m)}$ corresponding to misclassified data are increased and weights $w_n^{(m)}$ corresponding to correctly classified data are not changed.

Commonly used weak classifier

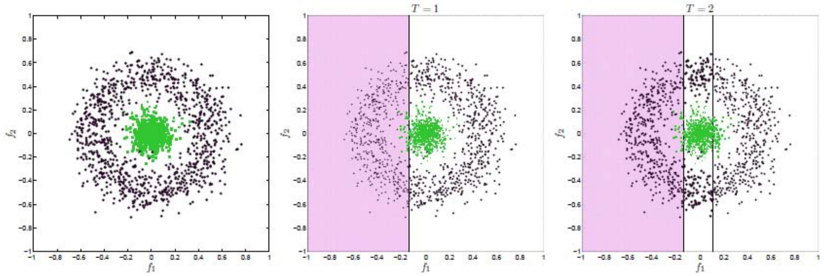
A commonly used weak classifier is the so called **decision stump** (decision tree with one node),
$$y_m(x_n) = \begin{cases} 1 & px_n^j < p\lambda \\ -1 & \text{otherwise} \end{cases}$$
where $j \in \{1, \dots, J\}$ indexes x_n, $p \in \{+1, -1\}$ is the "polarity" and $\lambda \in \mathbb{R}$ is a threshold. Hence, in step 2a we need to estimate $\theta = \begin{pmatrix} j & p & \lambda \end{pmatrix}^T$.

This corresponds to a partitioning of the input space into two half spaces by a decision boundary that is parallel to one of the input axes.

An illustrative example (I/III)

Generate a data set by sampling $\varphi \sim \mathcal{U}[0, 2\pi]$ and

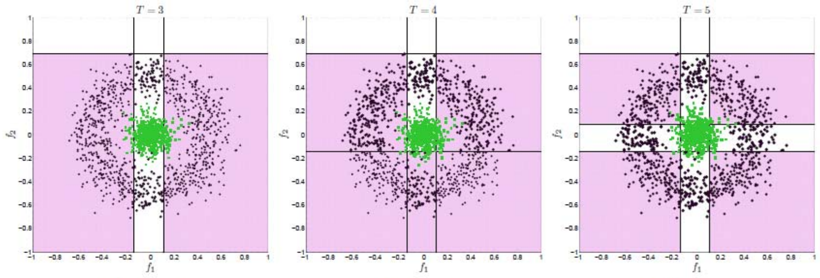
$$r = \begin{cases} \mathcal{N}(r; 0, 0.1) & \text{positive class} \\ \mathcal{N}(r; 0.5, 0.1) & \text{negative class} \end{cases} \qquad \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} r\cos(\varphi) \\ r\sin(\varphi) \end{pmatrix}$$



(a) Polar data (b) T = 1 (c) T = 2

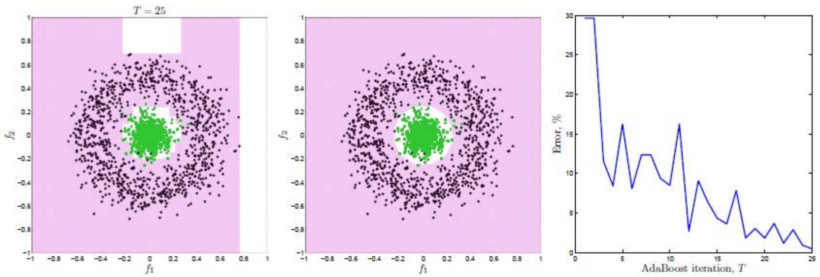
The data and the first two weak classifiers.

An illustrative example (II/III)



(d) T = 3 (e) T = 4 (f) T = 5

An illustrative example (III/III)



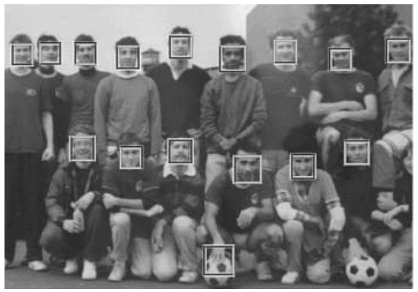
(g) Decision region (h) Probabilistic decision region. (i) Error

This example was borrowed from Karl Granström's PhD thesis (more examples are provided in Chapter 3),

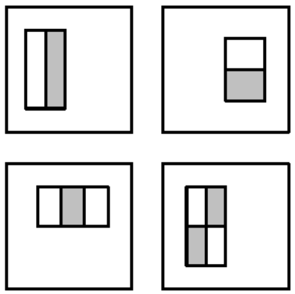
Karl Granström, *Extended target tracking using PHD filters*, Linköping Studies in Science and Technology, Dissertation No. 1476, November, 2012.

`http://liu.diva-portal.org/smash/record.jsf?searchId=3&pid=diva2:558084`

AdaBoost example 1 – face detection (I/III)



Viola, P. A. and Jones, M. J. **Robust real-time face detection**. *International Journal of Computer Vision*, 57(2):137-154, 2004.

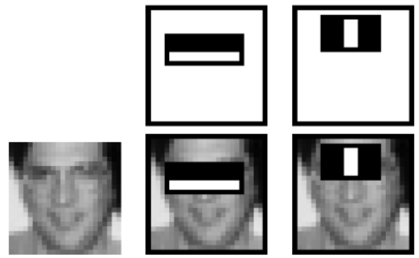


Copyright Springer (2004)

Very simple features (inputs) x_n^j are used. The feature is simply computed by adding the pixel values in the gray area and subtracting the pixel values in the white areas.

This can be done **extremely fast** using so called integral images (a.k.a. summed area table).

Drawback: the training time can be quite long.

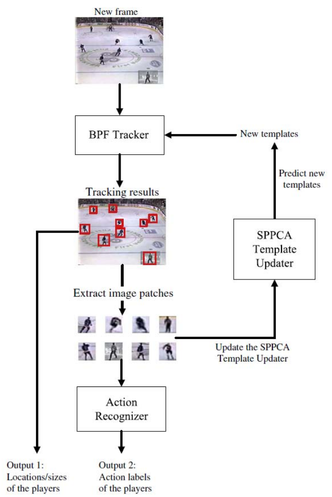


Copyright Springer (2004)

Interpretation of the two first features selected by AdaBoost:

The **first** feature measures the intensity difference between the eyes and the cheeks.

The **second** feature measures the intensity difference between the eyes and the bridge of the nose.



Copyright Springer (2009)

Makes use of many of the methods we have learned about in this course (such as AdaBoost, EM, probabilistic graphical models, logistic regerssion) and particle filtering.

Lu, W.-L., Okuma, K. and Little, J. J. **Tracking and Recognizing Actions of Multiple Hockey Players using the Boosted Particle Filter**. *Image and Vision Computing*, 27(1–2):189–205, 2009.

"Graphical models bring together graph theory and probability theory in a powerful formalism for multivariate statistical modeling." [1]

We can of course always handle probabilistic models using pure algebraic manipulation. Some reasons for using probabilistic graphical models,

1. A simple way to **visualize** the structure of a probabilistic model.
2. Knowledge about **model properties** directly from the graph.
3. A different way of **performing and structuring** calculations.

[1] Wainwright, M. J. and Jordan, M. I. **Graphical models, exponential families, and variational inference**, *Foundations and Trends in Machine Learning*, 1(1-2):1–305, 2008.

Probabilistic graphical models – basic facts (I/II)

A graph $\mathcal{G} = (\mathcal{V}, \mathcal{L})$ consists of

1. a set of **nodes** $\mathcal{V}$ (a.k.a. vertices) representing the random variables and
2. a set of **links** $\mathcal{L}$ (a.k.a. edges or arcs) containing elements $(i, j) \in \mathcal{L}$ connecting a pair of nodes $(i, j) \in \mathcal{V}$.

The links describes the probabilistic relations between the random variables (nodes).

Probabilistic graphical model representations,

1. **Bayesian networks** represents a set of random variables and their conditional dependencies via a directed acyclic graph (DAG).
2. **Markov random fields** represents a set of random variables having a Markov property by an undirected graph.

Probabilistic graphical models – basic facts (II/II)

Define

$$\mathcal{P}(j) \triangleq \{i \in \mathcal{V} \mid (i, j) \in \mathcal{E}\}$$

denoting the set of parents to node j.

The directed graph describes how the joint distribution $p(x)$ **factors** into a product of factors $p(x_i \mid x_{\mathcal{P}(i)})$ only depending on a subset of the variables,

$$p(x_{\mathcal{V}}) = \prod_{i \in \mathcal{V}} p(x_i \mid x_{\mathcal{P}(i)}),$$

where $x_{\mathcal{A}}$ denotes the set $\{x_i \mid i \in \mathcal{A}\}$.

Hence, node's value conditioned on its parents is independent of all other ancestors.

A few concepts to summarize lecture 8

Weak classifier: (a.k.a. base classifier) A classifier that is just slightly better than random guessing.

Boosting: Trains a sequence of M weak classifiers (models), where the error function used to train a certain model depends on the performance of the previous weak classifiers. All weak learners are then combined to a final strong classifier.

Probabilistic graphical model: Offers a compact way of encoding the conditional dependency structure of a set of random variables.

Bayesian network: A probabilistic graphical model that represents a set of random variables and their conditional dependencies via a directed acyclic graph (DAG).

Markov random field: A probabilistic graphical model that represents a set of random variables having a Markov property by an undirected graph.