Lecture 10, Machine Learning DD2431

Ensemble Learning

A. Maki
with contributions from J. Sullivan

Autumn 2015

Outline: Ensemble Learning

We will describe and investigate algorithms to

train weak classifiers/regressors and how to combine them

to construct a classifier/regressor more powerful than any of the individual ones.

They are called Ensemble learning, Commitee machine, etc.

Outline: Ensemble Learning

Background/methods:

Wisdom of Crowds

Why combine classifiers?

Bagging: static structure, parallel

Forests: an extension of bagging

Boosting: static structure, serial (Example: face detection)

No Free Lunch

"There is no such thing as **free lunch**"

(in a story of an American restaurant).

NFL Theorem in Machine Learning suggests:
There is no single **classifier** that performs better than *any* other classifiers in *all* classification problems.

No Free Lunch

"There is no such thing as **free lunch**"

(in a story of an American restaurant).

NFL Theorem in Machine Learning suggests:
There is no single **classifier** that performs better than *any* other classifiers in *all* classification problems.

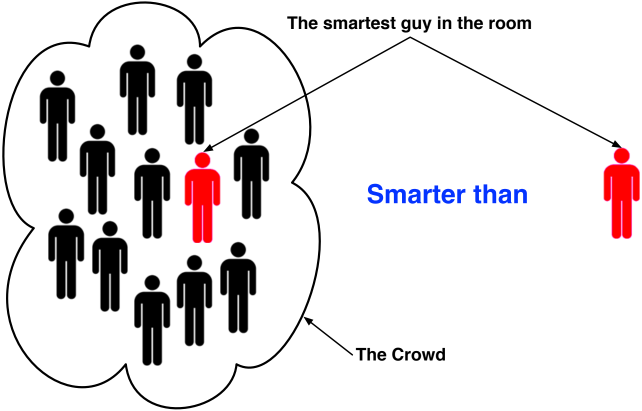
No Free Lunch

"There is no such thing as **free lunch**"

(in a story of an American restaurant).

NFL Theorem in Machine Learning suggests:
There is no single **classifier** that performs better than *any* other classifiers in *all* classification problems.

The Wisdom of Crowds

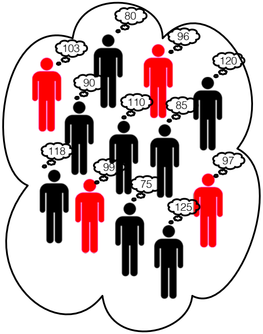


The **collective knowledge** of a *diverse* and *independent* body of people typically **exceeds** the knowledge of **any single individual** and can be harnessed by voting.

Consider this scenario

Asked each person in the same crowd
(in a trade fair in England in early 1900):

How much does the bull weigh?



Crowd's prediction:

AVERAGE of all (800) predictions.

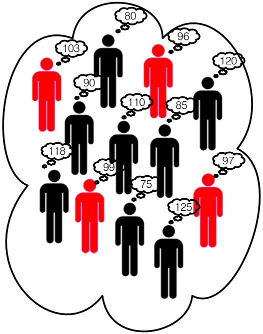
⇐ This crowd predicts **1197** lb.

(The bull weighs 1198 lb.)

Consider this scenario

Asked each person in the same crowd
(in a trade fair in England in early 1900):

How much does the bull weigh?



Crowd's prediction:

AVERAGE of all (800) predictions.

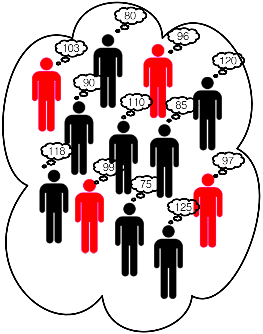
⇐ This crowd predicts **1197** lb.

(The bull weighs 1198 lb.)

Consider this scenario

Asked each person in the same crowd
(in a trade fair in England in early 1900):

How much does the bull weigh?



Crowd's prediction:

AVERAGE of all (800) predictions.

⇐ This crowd predicts **1197** lb.

(The bull weighs 1198 lb.)

The Wisdom of Crowds - Really?

Crowd wiser than **any individual**

- When ?
- For which questions ?

See **The Wisdom of Crowds** by *James Surowiecki* published in 2004 to see this idea applied to business.

What makes a crowd wise?

Four elements required to form a wise crow (*J. Surowiecki*):

- **Diversity of opinion.** People in crowd should have a range of experiences, education and opinions.
- **Independence.** Prediction by person in crowd is not influenced by other people in the crowd.
- **Decentralization** People have specializations and local knowledge.
- **Aggregation.** There is a mechanism for aggregating all predictions into one single prediction.

What makes a crowd wise?

Four elements required to form a wise crow (*J. Surowiecki*):

- **Diversity of opinion.** People in crowd should have a range of experiences, education and opinions.
- **Independence.** Prediction by person in crowd is not influenced by other people in the crowd.
- **Decentralization** People have specializations and local knowledge.
- **Aggregation.** There is a mechanism for aggregating all predictions into one single prediction.

What makes a crowd wise?

Four elements required to form a wise crow (*J. Surowiecki*):

- **Diversity of opinion.** People in crowd should have a range of experiences, education and opinions.
- **Independence.** Prediction by person in crowd is not influenced by other people in the crowd.
- **Decentralization** People have specializations and local knowledge.
- **Aggregation.** There is a mechanism for aggregating all predictions into one single prediction.

What makes a crowd wise?

Four elements required to form a wise crow (*J. Surowiecki*):

- **Diversity of opinion.** People in crowd should have a range of experiences, education and opinions.
- **Independence.** Prediction by person in crowd is not influenced by other people in the crowd.
- **Decentralization** People have specializations and local knowledge.
- **Aggregation.** There is a mechanism for aggregating all predictions into one single prediction.

But....

Why not just asking a bunch of experts??

- Large enough crowd $\implies$ high probability a sufficient number of experts will be in crowd (for any question).
- Random selection $\implies$ don't make a biased choice in experts.
- For some questions it may be hard to identify a diverse set of experts

But....

Why not just asking a bunch of experts??

- Large enough crowd $\implies$ high probability a sufficient number of experts will be in crowd (for any question).
- Random selection $\implies$ don't make a biased choice in experts.
- For some questions it may be hard to identify a diverse set of experts

But....

Why not just asking a bunch of experts??

- Large enough crowd $\implies$ high probability a sufficient number of experts will be in crowd (for any question).
- Random selection $\implies$ don't make a biased choice in experts.
- For some questions it may be hard to identify a diverse set of experts

But....

Why not just asking a bunch of experts??

- Large enough crowd $\implies$ high probability a sufficient number of experts will be in crowd (for any question).
- Random selection $\implies$ don't make a biased choice in experts.
- For some questions it may be hard to identify a diverse set of experts

The crowd must be careful

In the analysis of the crowd it is implicitly assumed:

- each person is not concerned with the opinions of others,
- The non-experts will predict a completely random wrong answer - these will somewhat cancel each other out.

However, there may be a systematic and consistent bias in the non-experts' predictions.

If the crowd does not contain sufficient experts then *truth by consensus*, rather than fact, leads to **Wikiality**!

(Term coined by *Stephen Colbert* in an episode of the *The Colbert Report* in July 2006.)

The crowd must be careful

In the analysis of the crowd it is implicitly assumed:

- each person is not concerned with the opinions of others,
- The non-experts will predict a completely random wrong answer - these will somewhat cancel each other out.

However, there may be a systematic and consistent bias in the non-experts' predictions.

If the crowd does not contain sufficient experts then *truth by consensus*, rather than fact, leads to **Wikiality**!

(Term coined by *Stephen Colbert* in an episode of the *The Colbert Report* in July 2006.)

Combining classifiers

Will exploit *Wisdom of crowd* ideas for specific tasks by

- combining classifier predictions **and**
- aim to combine independent and diverse classifiers.

But will use labelled training data

- to identify the **expert** classifiers in the pool;
- to identify **complementary** classifiers;
- to indicate how to best combine them.

Combining classifiers

Will exploit *Wisdom of crowd* ideas for specific tasks by

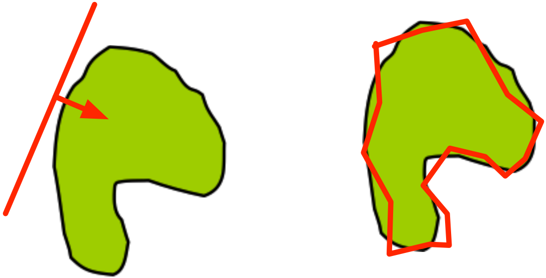
- combining classifier predictions **and**
- aim to combine independent and diverse classifiers.

But will use labelled training data

- to identify the **expert** classifiers in the pool;
- to identify **complementary** classifiers;
- to indicate how to best combine them.

Example:

Voting of oriented hyper-planes can define convex regions.
Green region is the true boundary.



High-bias classifier **Low-bias classifier**

Low model complexity (small # of d.o.f.) $\Longrightarrow$ High-bias
High model complexity (large # of d.o.f.) $\Longrightarrow$ Low-bias

Ensemble Prediction: Voting

A **diverse** and **complementary** set of high-bias classifiers, with performance better than chance, combined by **voting**

$$f_V(\mathbf{x}) = \text{sign}\left(\sum_{t=1}^{T} h_t(\mathbf{x})\right)$$

can produce a classifier with a low-bias.

$h_t \in \mathcal{H}$ where $\mathcal{H}$ is a family of possible weak classifiers functions.

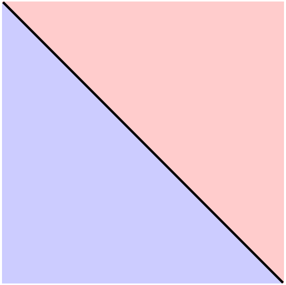
Ensemble method: **Bagging**

Bootstrap **Agg**regat**ing**

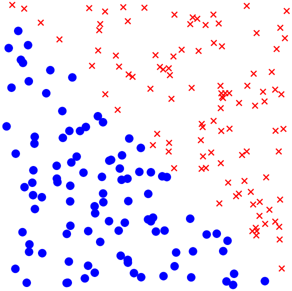
Use bootstrap replicates of training set by sampling with replacement.

On each replicate learn one model – combined altogether.

Binary classification example



True decision boundary



Training data set $\mathcal{S}_i$

Estimate the true decision boundary with a *decision tree* trained from some labeled training set $\mathcal{S}_i$.

High variance, Low bias classifiers

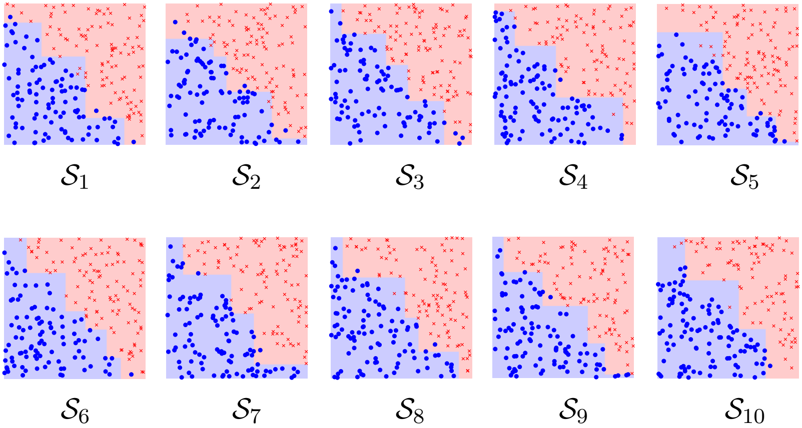
E.g. *decision trees*

High variance classifiers produce differing decision boundaries which are highly dependent on the training data.

Low bias classifiers produce decision boundaries which on average are good approximations to the true decision boundary.

Ensemble predictions using diverse high-variance, low-bias classifiers reduce the variance of the ensemble classifier.

Estimated decision boundaries found using bootstrap replicates:



Property of instability

See how the decision boundaries on the previous slide differ from the

expected decision boundary of the decision tree classifier (with $m = 200$ training points).

Bagging - Bootstrap Aggregating

Input: Training data

$$\mathcal{S} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m)\}$$

of inputs $\mathbf{x}_j \in \mathbb{R}^d$ and their labels or real values y_j.

Iterate: for $b = 1, \ldots, B$

1. Sample training examples, *with replacement*, m times from $\mathcal{S}$ to create $\mathcal{S}_b$.
2. Use this bootstrap sample $\mathcal{S}_b$ to estimate the regression or classification function f_b.

Bagging - Bootstrap Aggregating

Input: Training data

$$\mathcal{S} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m)\}$$

of inputs $\mathbf{x}_j \in \mathbb{R}^d$ and their labels or real values y_j.

Iterate: for $b = 1, \ldots, B$

1. Sample training examples, *with replacement*, m times from $\mathcal{S}$ to create $\mathcal{S}_b$.
2. Use this bootstrap sample $\mathcal{S}_b$ to estimate the regression or classification function f_b.

Output: The bagging estimate for

Regression:

$$f_{\text{bag}}(\mathbf{x}) = \frac{1}{B}\sum_{b=1}^{B} f_b(\mathbf{x})$$

Classification:

$$f_{\text{bag}}(\mathbf{x}) = \arg\max_{1\le k\le K} \sum_{b=1}^{B} \text{Ind}\left(f_b(\mathbf{x}) = k\right)$$

Note: $\text{Ind}\,(x) = 1$ if $x = \text{TRUE}$ otherwise $\text{Ind}\,(x) = 0$

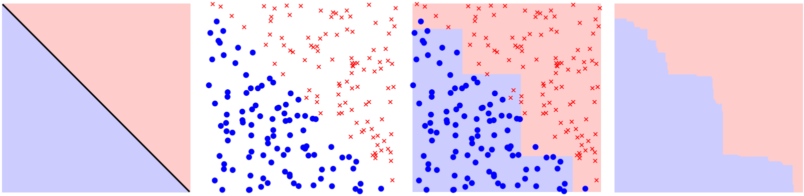
Bagging

is a procedure to reduce the variance of our classifier when labelled training data is limited.

Bias of **bagged classifier** may be marginally less than the base classifiers.

Note: it only produces good results for **high variance**, **low bias** classifiers.

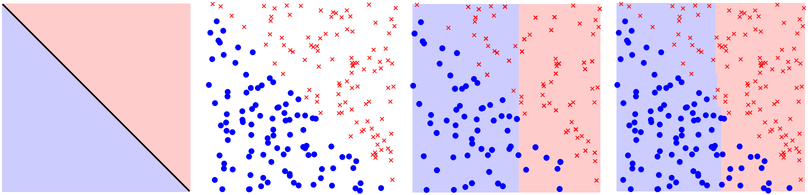
Apply bagging to the original example



Ground truth $\mathcal{S}_1$ **Classifier from** $\mathcal{S}_1$ **Bagged classifier** $B = 100$.

Apply bagging to the original example (cont.)

If we bag a **high bias, low variance** classifier - *oriented horizontal and vertical lines* - we don't get any benefit.



Ground truth | $\mathcal{S}_1$ | **Classifier from $\mathcal{S}_1$** | **Bagged classifier $B = 100$.**

Ensemble method: **Forest**

Decision/Random/Randomized Forest

Bagging + Random *feature selection* at each node

Decision Forests / Random Forests

Two kind of randomnesses involved in:
- Sampling training data (the same as in Bagging)
- Feature selection at each node

Trees are less correlated, i.e. even **higher variance** between weak learners.

A classier suited to multi-class problem.

Decision Forests / Random Forests

Two kind of randomnesses involved in:
- Sampling training data (the same as in Bagging)
- Feature selection at each node

Trees are less correlated, i.e. even **higher variance** between weak learners.

A classier suited to multi-class problem.

Decision Forests / Random Forests

Two kind of randomnesses involved in:
- Sampling training data (the same as in Bagging)
- Feature selection at each node

Trees are less correlated, i.e. even **higher variance** between weak learners.

A classier suited to multi-class problem.

Ensemble method: **Boosting**

Started from a question:

Can a set of weak learners create a single strong classifier where a weak learner performs only slightly better than a chance? (Kearns, 1988)

Loop:

- Apply learner to weighted samples
- Increase weights of misclassified examples

Ensemble Method: Boosting

Input: Training data $\mathcal{S} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m)\}$ of inputs $\mathbf{x}_i$ and their labels $y_i \in \{-1, 1\}$ or real values.

$\mathcal{H}$: a family of possible weak classifiers/regression functions.

Output: A strong classifier/regression function

$$f_T(\mathbf{x}) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(\mathbf{x})\right) \quad \text{or} \quad f_T(\mathbf{x}) = \sum_{t=1}^{T} \alpha_t h_t(\mathbf{x})$$

weighted sum of weak classifiers

$h_t \in \mathcal{H} \quad t = 1, ..., T$

α_t: confidence/reliability

Ensemble Method: Boosting

Input: Training data $\mathcal{S} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m)\}$ of inputs $\mathbf{x}_i$ and their labels $y_i \in \{-1, 1\}$ or real values.

$\mathcal{H}$: a family of possible weak classifiers/regression functions.

Output: A strong classifier/regression function

$$f_T(\mathbf{x}) = \mathrm{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(\mathbf{x})\right) \quad \text{or} \quad f_T(\mathbf{x}) = \sum_{t=1}^{T} \alpha_t h_t(\mathbf{x})$$

weighted sum of weak classifiers

$h_t \in \mathcal{H} \quad t = 1, ..., T$

α_t: confidence/reliability

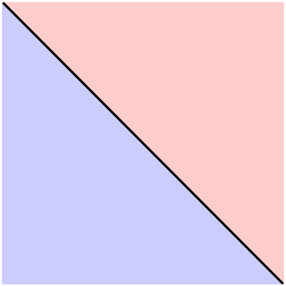
Ensemble Method: Boosting

How ?? (Just consider case of classification.)

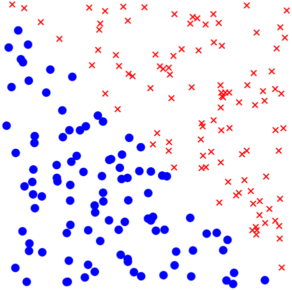
- Performance of classifiers $h_1, \ldots, h_t$ helps define h_{t+1}.
- Maintain **weight** $w_i^{(t)}$ for each training example in $\mathcal{S}$.
- Large $w_i^{(t)} \implies \mathbf{x}_i$ has greater influence on choice of h_t.
- Iteration t: $w_i^{(t)}$ increased if $\mathbf{x}_i$ wrongly classified by h_t.
- Iteration t: $w_i^{(t)}$ decreased if $\mathbf{x}_i$ correctly classified by h_t.

Remember: Each $h_t \in \mathcal{H}$

Binary classification example



True decision boundary

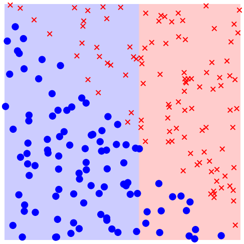


Training data

$\mathcal{H}$ is the set of all possible oriented vertical and horizontal lines.

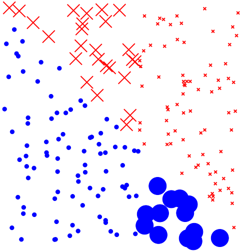
Example

Round 1



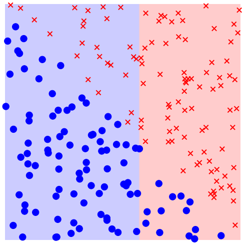
Chosen weak classifier

$\epsilon_1 = 0.19, \ \alpha_1 = 1.45$



Re-weight training points

$w_i^{(2)}$**'s**

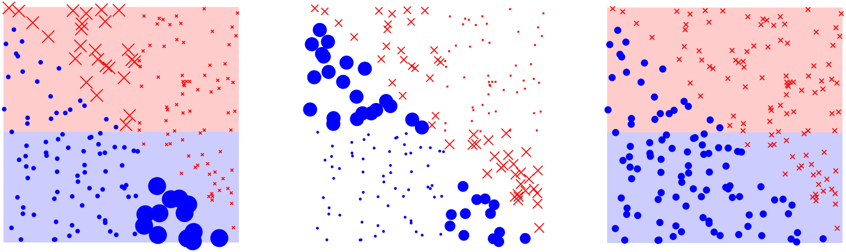


Current strong classifier

$f_2(\mathbf{x})$

Example

Round 2



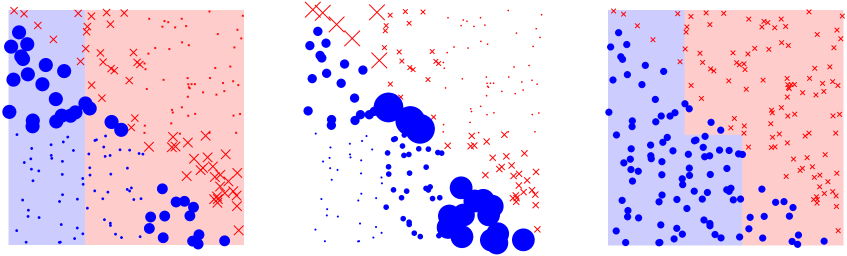
Chosen weak classifier — $\epsilon_2 = 0.1512$, $\alpha_2 = 1.725$

Re-weight training points — $w_i^{(3)}$**'s**

Current strong classifier — $f_2(\mathbf{x})$

Example

Round 3



Chosen weak classifier

$\epsilon_3 = 0.2324,\ \alpha_3 = 1.1946$

Re-weight training points

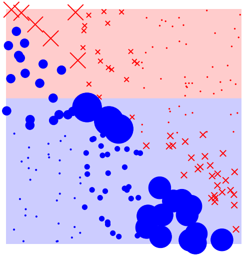
$w_i^{(4)}$**'s**

Current strong classifier

$f_3(\mathbf{x})$

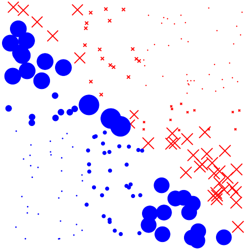
Example

Round 4



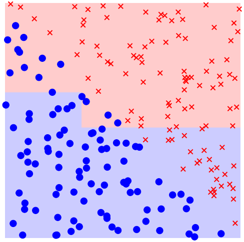
Chosen weak classifier

$\epsilon_4 = 0.2714$, $\alpha_4 = 0.9874$



Re-weight training points

$w_i^{(5)}$**'s**

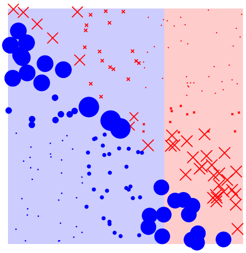


Current strong classifier

$f_4(\mathbf{x})$

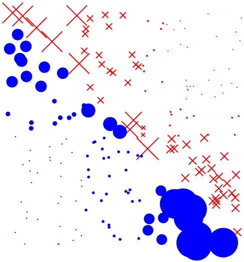
Example

Round 5



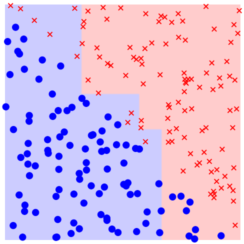
Chosen weak classifier

$\epsilon_5 = 0.2616$, $\alpha_5 = 1.0375$



Re-weight training points

$w_i^{(6)}$**'s**

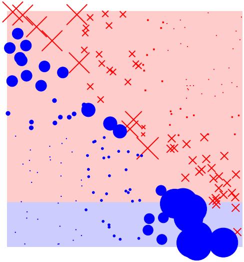


Current strong classifier

$f_5(\mathbf{x})$

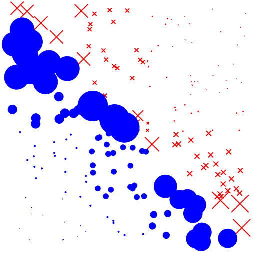
Example

Round 6



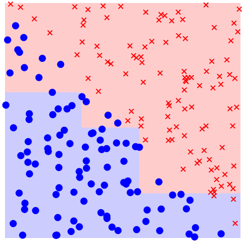
Chosen weak classifier

$\epsilon_6 = 0.2262$, $\alpha_6 = 1.2298$



Re-weight training points

$w_i^{(7)}$**'s**

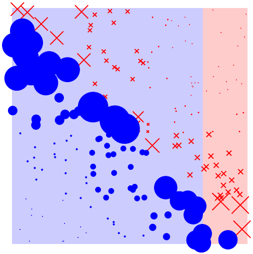


Current strong classifier

$f_6(\mathbf{x})$

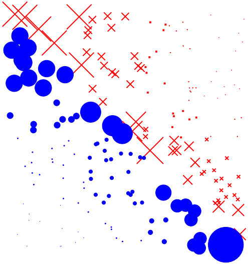
Example

Round 7



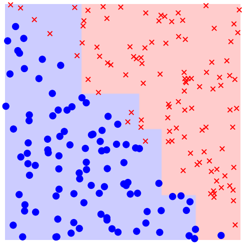
Chosen weak classifier

$\epsilon_7 = 0.2680,\ \alpha_7 = 1.0049$



Re-weight training points

$w_i^{(8)}$**'s**

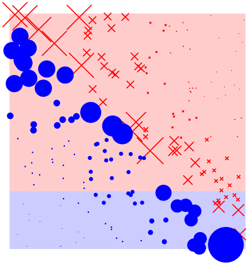


Current strong classifier

$f_7(\mathbf{x})$

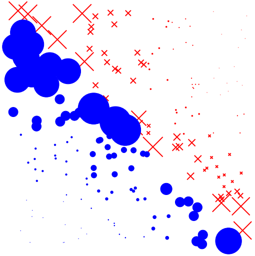
Example

Round 8



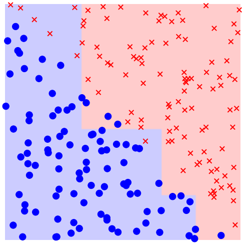
Chosen weak classifier

$\epsilon_8 = 0.3282, \ \alpha_8 = 0.7165$



Re-weight training points

$w_i^{(9)}$**'s**

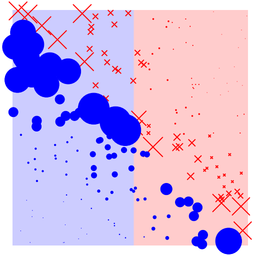


Current strong classifier

$f_8(\mathbf{x})$

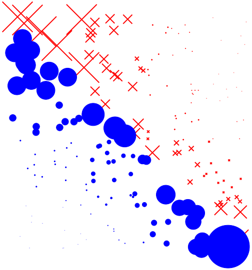
Example

Round 9



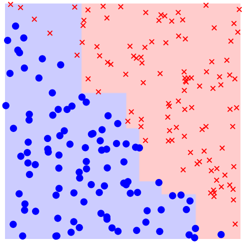
Chosen weak classifier

$\epsilon_9 = 0.3048, \ \alpha_9 = 0.8246$



Re-weight training points

$w_i^{(10)}$**'s**

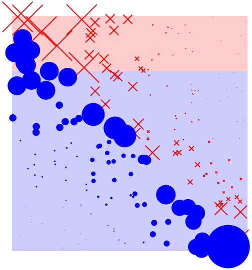


Current strong classifier

$f_9(\mathbf{x})$

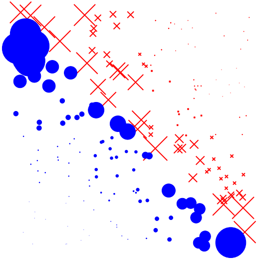
Example

Round 10



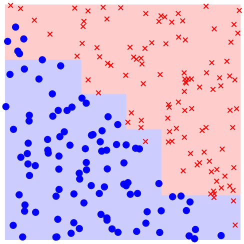
Chosen weak classifier

$\epsilon_{10} = 0.2943, \; \alpha_{10} = 0.8744$



Re-weight training points

$w_i^{(11)}$**'s**

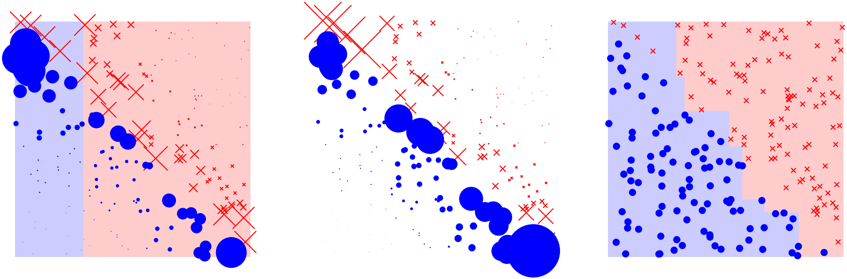


Current strong classifier

$f_{10}(\mathbf{x})$

Example

Round 11



Chosen weak classifier

$\epsilon_{11} = 0.2876$, $\alpha_{11} = 0.9071$

Re-weight training points

$w_i^{(12)}$'s

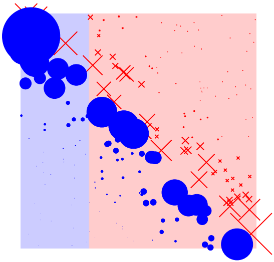
Current strong classifier

$f_{11}(\mathbf{x})$

Example

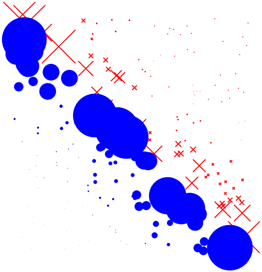
Example

Round 21



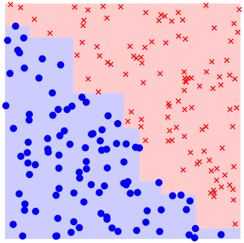
Chosen weak classifier

$\epsilon_{21} = 0.3491$, $\alpha_{21} = 0.6232$



Re-weight training points

$w_i^{(22)}$**'s**



Current strong classifier

$f_{21}(\mathbf{x})$

Adaboost Algorithm (Freund & Schapire, 1997)

Given:

- Labeled training data

$$\mathcal{S} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m)\}$$

  of inputs $\mathbf{x}_j \in \mathbb{R}^d$ and their labels $y_j \in \{-1, 1\}$.
- A set/class $\mathcal{H}$ of T possible weak classifiers.

Initialize:

- Introduce a weight, $w_j^{(1)}$, for each training sample.
- Set $w_j^{(1)} = \frac{1}{m}$ for each j.

Adaboost Algorithm (cont.)

Iterate: for $t = 1, \ldots, T$

1. Train weak classifier $h_t \in \mathcal{H}$ using $\mathcal{S}$ and $w_1^{(t)}, \ldots, w_m^{(t)}$; select the one that minimizes the training error:

$$\epsilon_t = \sum_{j=1}^{m} w_j^{(t)} \text{ Ind}\left(y_j \neq h_t(\mathbf{x}_j)\right)$$

(sum of the weights for misclassified samples)

2. Compute the reliability coefficient:

$$\alpha_t = \log\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$$

ϵ_t must be less than 0.5. Break out of loop if $\epsilon_t \approx .5$

3. Update weights using:

$$w_j^{(t+1)} = w_j^{(t)} exp(-\alpha_t y_j h_t(\mathbf{x}_j))$$

4. Normalize the weights so that they sum to 1.

Adaboost Algorithm (cont.)

Iterate: for $t = 1, \ldots, T$

1. Train weak classifier $h_t \in \mathcal{H}$ using $\mathcal{S}$ and $w_1^{(t)}, \ldots, w_m^{(t)}$; select the one that minimizes the training error:

$$\epsilon_t = \sum_{j=1}^{m} w_j^{(t)} \ \mathrm{Ind}\left(y_j \neq h_t(\mathbf{x}_j)\right)$$

(sum of the weights for misclassified samples)

2. Compute the reliability coefficient:

$$\alpha_t = \log\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$$

ϵ_t must be less than 0.5. Break out of loop if $\epsilon_t \approx .5$

3. Update weights using:

$$w_j^{(t+1)} = w_j^{(t)} exp(-\alpha_t y_j h_t(\mathbf{x}_j))$$

4. Normalize the weights so that they sum to 1.

Adaboost Algorithm (cont.)

Iterate: for $t = 1, \ldots, T$

1. Train weak classifier $h_t \in \mathcal{H}$ using $\mathcal{S}$ and $w_1^{(t)}, \ldots, w_m^{(t)}$; select the one that minimizes the training error:

$$\epsilon_t = \sum_{j=1}^{m} w_j^{(t)} \text{ Ind}\left(y_j \neq h_t(\mathbf{x}_j)\right)$$

(sum of the weights for misclassified samples)

2. Compute the reliability coefficient:

$$\alpha_t = \log\left(\frac{1 - \epsilon_t}{\epsilon_t}\right)$$

ϵ_t must be less than 0.5. Break out of loop if $\epsilon_t \approx .5$

3. Update weights using:

$$w_j^{(t+1)} = w_j^{(t)} exp(-\alpha_t y_j h_t(\mathbf{x}_j))$$

4. Normalize the weights so that they sum to 1.

Properties of the Boosting algorithm

Training Error: Training error $\rightarrow 0$ exponentially.

Good Generalization Properties: Would expect over-fitting but even when training error vanishes the test error asymptotes

Why? Boosting tries to increase the margin of the training examples even when the training error is zero:

$$f_T(\mathbf{x}) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(\mathbf{x})\right) = \text{sign}\left(\phi_T(\mathbf{x})\right)$$

Margin of a correctly classified example is: $y_i \, \phi_T(\mathbf{x}_i)$

The larger the margin $\implies$ further example is from the decision boundary $\implies$ better generalization ability.

Properties of the Boosting algorithm

Training Error: Training error $\rightarrow 0$ exponentially.

Good Generalization Properties: Would expect over-fitting but even when training error vanishes the test error asymptotes

Why? Boosting tries to increase the margin of the training examples even when the training error is zero:

$$f_T(\mathbf{x}) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(\mathbf{x})\right) = \text{sign}\left(\phi_T(\mathbf{x})\right)$$

Margin of a correctly classified example is: $y_i\, \phi_T(\mathbf{x}_i)$

The larger the margin $\Longrightarrow$ further example is from the decision boundary $\Longrightarrow$ better generalization ability.

Properties of the Boosting algorithm

Training Error: Training error $\to 0$ exponentially.

Good Generalization Properties: Would expect over-fitting but even when training error vanishes the test error asymptotes

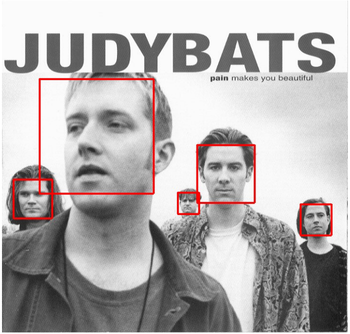
Why? Boosting tries to increase the margin of the training examples even when the training error is zero:

$$f_T(\mathbf{x}) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(\mathbf{x})\right) = \text{sign}\left(\phi_T(\mathbf{x})\right)$$

Margin of a correctly classified example is: $y_i \, \phi_T(\mathbf{x}_i)$

The larger the margin $\implies$ further example is from the decision boundary $\implies$ better generalization ability.

Example: Viola & Jones Face Detection



- Most state-of-the-art face detection on mobile phones, digital cameras etc. are based on this algorithm.
- Example of a classifier constructed using the Boosting algorithm.

Viola & Jones: Training data

Positive training examples: Image patches corresponding to faces - $(\mathbf{x}_i, 1)$.

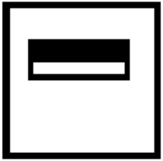
Negative training examples: Random image patches from images not containing faces - $(\mathbf{x}_j, -1)$.

Note: All patches are re-scaled to have same size.

$\Uparrow$

Positive training examples

Viola & Jones: Weak classifier



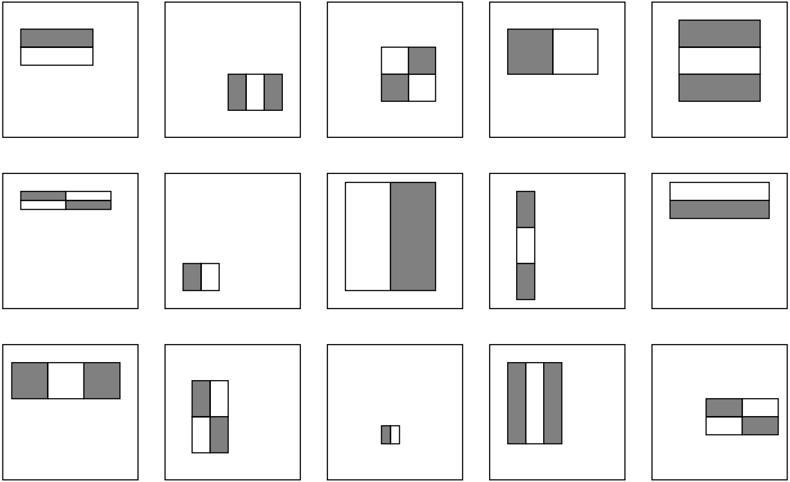
FACE or NON-FACE

Input: $\mathbf{x}$ **Apply filter:** $f^j(\mathbf{x})$ **Output:** $h(\mathbf{x}) = (f^j(\mathbf{x}) > \theta)$

Filters used compute differences between sums of pixels in adjacent rectangles. (These can be computed very quickly using **Integral Images**.)

Viola & Jones: Filters Considered

Huge **library** of possible Haar-like filters, $f^1, \ldots, f^n$ with $n \approx 16,000,000$.



Viola & Jones: AdaBoost training

Recap: define weak classifier as

$$h_t(\mathbf{x}) = \begin{cases} 1 & \text{if } f^{j_t}(\mathbf{x}) > \theta_t \\ -1 & \text{otherwise} \end{cases}$$

Use AdaBoost to efficiently choose the **best weak classifiers** and to **combine** them.

Remember: a weak classifier corresponds to a filter type and a threshold.

Viola & Jones: AdaBoost training (cont.)

For $t = 1, \ldots, T$

- for each filter type j
  1. Apply filter, f^j, to each example.
  2. Sort examples by their filter responses.
  3. Select best threshold for this classifier: θ_{tj}.
  4. Keep record of error of this classifier: ϵ_{tj}.
- Select the filter-threshold combination (weak classifier j^*) with minimum error. Then set $j_t = j^*$, $\epsilon_t = \epsilon_{tj^*}$ and $\theta_t = \theta_{tj^*}$.
- Re-weight examples according to the AdaBoost formualae.

Note: (There are many tricks to make this implementation more efficient.)

Viola & Jones: Sliding window

Remember: Better classification rates if use a classifier, f_T, with large T.

Given a new image, I, detect the faces in the image by:

- for each plausible face size s
  - for each possible patch centre c
    1. Extract sub-patch of size s at c from I.
    2. Re-scale patch to size of training patches.
    3. Apply detector to patch.
    4. Keep record of s and c if the detector returns positive.

This is a **lot** of patches to be examined. If T is very large processing an image will be very slow!

Viola & Jones: Sliding window

Remember: Better classification rates if use a classifier, f_T, with large T.

Given a new image, I, detect the faces in the image by:

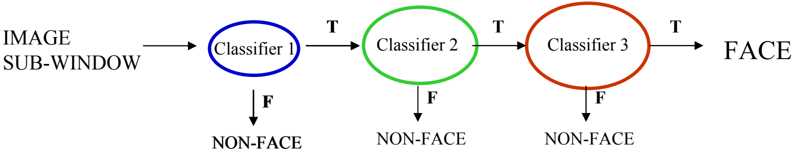
- for each plausible face size s
  - for each possible patch centre c
    1. Extract sub-patch of size s at c from I.
    2. Re-scale patch to size of training patches.
    3. Apply detector to patch.
    4. Keep record of s and c if the detector returns positive.

This is a **lot** of patches to be examined. If T is very large processing an image will be very slow!

Viola & Jones: Cascade of classifiers

But:
only a tiny proportion of the patches will be faces **and** many of them will not look anything like a face.

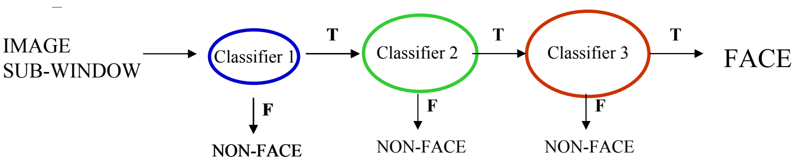
Exploit this fact: Introduce a cascade of increasingly strong classifiers



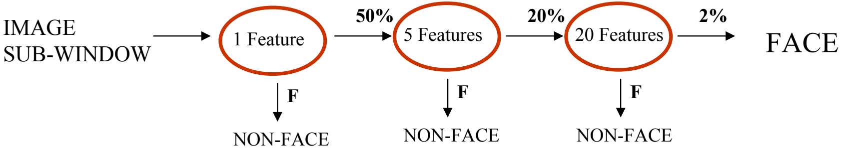
Viola & Jones: Cascade of classifiers

But:
only a tiny proportion of the patches will be faces **and** many of them will not look anything like a face.

Exploit this fact: Introduce a cascade of increasingly strong classifiers

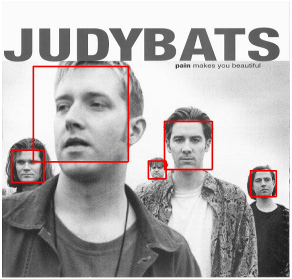


Viola & Jones: Cascade of classifiers



- A 1 feature classifier achieves 100% detection rate and about 50% false positive rate.
- A 5 feature classifier achieves 100% detection rate and 40% false positive rate (20% cumulative) - using data from previous stage.
- A 20 feature classifier achieves 100% detection rate with 10% false positive rate (2% cumulative).

Viola & Jones: Typical Results



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

Summary

Summary: Ensemble Prediction

Can combine many weak classifiers/regressors into a stronger classifier; voting, averaging, bagging

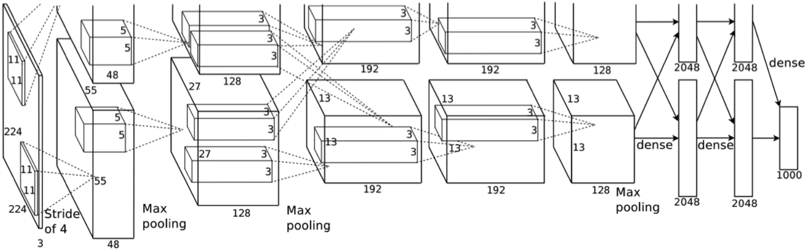
- if weak classifiers/regressors are better than random.
- if there is sufficient de-correlation (independence) amongst the weak classifiers/regressors.

Summary: Ensemble Prediction & Learning

Can combine many (high-bias) weak classifiers/regressors into a **strong** classifier; boosting

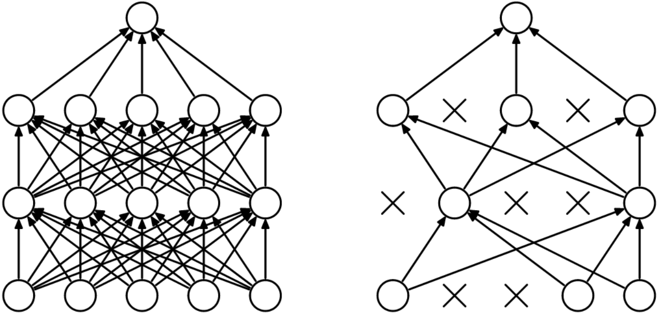
- if weak classifiers/regressors are **chosen** and **combined** using knowledge of how well they and others performed on the task on training data.
- The selection and combination encourages the weak classifiers to be complementary, diverse and de-correlated.

Modern Convolutional Neural Networks (ConvNets)



Krizhevsky et al. NIPS 2012.

Dropout: ensemble learning in ConvNets



(a) Standard Neural Net (b) After applying dropout.

Srivastava, Hinton, Krizhevsky, Sutskever and Salakhutdinov, **Dropout: A Simple Way to Prevent Neural Networks from Overtting.** *Journal of Machine Learning Research* 15: 1929-1958, 2014.