9.520: Class 24

Bagging and Boosting

Alessandro Verri

(based on notes from Theodoros Evgeniou)
About this class

**Theme** We discuss *bagging* and *boosting* and provide some attempts to justify them.

**Web** The slides and all what concerns this class can be found on the web.
Plan

◊ Bagging

◊ Bias-Variance Decomposition

◊ Combinations of Kernel Machines (i.e. SVM)

◊ Boosting and Adaboost

◊ Bounds for Boosting

◊ Additive Logistic Regression
Some motivations for combining Learning Machines

- Suppose you have many “easy rules”: combining them may be a good idea.

- Parameter Estimation: combine many machines each with different parameters?

- Bootstrap: maybe helps with “variance”? 
Bagging (Bootstrap AGGregatING)

Given a training set \( D = \{ (x_1, y_1), \ldots, (x_\ell, y_\ell) \} \),

- sample \( N \) sets of \( \ell \) elements from \( D \) (with replacement) \( D_1, D_2, \ldots D_N \) \( \rightarrow N \) quasi replica training sets;

- train a machine on each \( D_i, i = 1, \ldots, N \) and obtain a sequence of \( N \) outputs \( f_1(x), \ldots, f_N(x) \).
Bagging (cont.)

The final aggregate classifier can be

- for regression

\[ \bar{f}(x) = E\{f_i(x)\}, \]

the average of \( f_i \) for \( i = 1, ..., N; \)

- for classification

\[ \bar{f}(x) = \theta(E\{f_i(x)\}) \]

the majority vote from \( f_i(x) \).
Theoretical analysis

Breiman (1996) studies the average generalization performance of a learning algorithm with respect to the training set. It is possible to relate this quantity to the distance or bias between the Bayes optimal solution and the average solution of the learning algorithm and the variance of the solution.
Bias – Variance for Regression

Let

\[ I[f] = \int (f(x) - y)^2 p(x, y) dx dy \]

be the expected risk and \( f_0 \) the regression function. With \( \bar{f}(x) = E \{ f_i(x) \} \), if we define the bias as

\[ \int (f_0(x) - \bar{f}(x))^2 p(x) dx \]

and the variance as

\[ E \left\{ \int (f_i(x) - \bar{f}(x))^2 p(x) dx \right\}, \]

we have the decomposition

\[ E \{ I[f_i] \} = I[f_0] + \text{bias} + \text{variance}. \]
Bias-Variance for Classification

No unique decomposition for classification exists. In the binary case, with \( \bar{f}(x) = \theta(E\{f_i(x)\}) \), the decomposition suggested by Kong and Dietterich (1995) is

\[
I[\bar{f}] - I[f_0]
\]

for the bias, and

\[
E\{I[f_i]\} - I[\bar{f}]
\]

for the variance, which (again) gives

\[
E\{I[f_i]\} = I[f_0] + bias + variance.
\]
Bagging reduces variance

If each single classifier is unstable – that is, it has high variance, the aggregated classifier \( \bar{f} \) has a smaller variance than a single original classifier.

The aggregated classifier \( \bar{f} \) can be thought of as an approximation to the true average \( f \) obtained by replacing the probability distribution \( p \) with the bootstrap approximation to \( p \) obtained concentrating mass \( 1/\ell \) at each point \((x_i, y_i)\).
Ensembles of Kernel Machines

What happens when combining kernel machines (i.e. SVM)?

- Different subsamples of training data (bagging)
- Different kernels or different features
- Different parameters (i.e. regularization parameter)
Combination of SVM Machines

Let $f_1(x), \ldots, f_N(x)$ be SVM machines we want to combine and let

$$f(x) = \sum_{n=1}^{N} c_n f_n(x)$$

for some fixed $c_n > 0$ with $\sum c_n = 1$.

We want to study the generalization performance of $f(x)$
Leave-one-out error

The leave-one-out error is computed in three steps

1. Leave a training point out

2. Train the remaining points and test the point left out

3. Repeat for each training point and count “errors”

Theorem (Luntz and Brailovski, 1969)

\[ E\{I[f_\ell]\} = E\{CV \text{ error of } f_{\ell+1}\} \]
Leave-one-out of a kernel machine

The leave-one-out error of a kernel machine (without $b$) is upper bounded by

$$\sum_{i=1}^{\ell} \theta(\alpha_i K(x_i, x_i) - y_i f(x_i))$$

(Jaakkola and Haussler, 1998)
Proof (SVM case)

Given \( \ell \) examples we have

\[
L(\alpha) = \sum_j \alpha_j - \frac{1}{2} \sum_{j,k} \alpha_j \alpha_k y_j y_k K_{jk}.
\]

If \( \alpha^* \) denotes the maximizer we clearly have

\[
L(\alpha^*) \geq L(\alpha) \quad \forall \alpha.
\]

If we leave the \( i \)-th point out, denoting with \( L' \) the new objective function and with \( \alpha' \) the \( \ell - 1 \)-dimensional vector of Lagrangian multipliers, we can write

\[
L'(\alpha') = \sum_{j \neq i} \alpha_j - \frac{1}{2} \sum_{j,k \neq i} \alpha_j \alpha_k y_j y_k K_{jk}.
\]

If \( \bar{\alpha}' \) denotes the maximizer we have

\[
L'(\bar{\alpha}') \geq L'(\alpha') \quad \forall \alpha'.
\]
Proof (cont.)

Consider the objective function $L$ with $\alpha_i = \alpha^*_i$,

$$L(\alpha'; \alpha^*_i) = L'(\alpha') - \alpha^*_i y_i \sum_{j \neq i} \alpha_j y_j K_{ij} + \alpha^*_i.$$ 

Since $\alpha^*$ is a maximizer for $L$ we have that

$$L(\alpha'^*; \alpha^*_i) \geq L(\bar{\alpha'}; \alpha^*_i)$$

which can be rewritten as

$$L'(\alpha'^*) - \alpha^*_i y_i \sum_{j \neq i} \alpha^*_j y_j K_{ij} \geq L'(\bar{\alpha'}) - \alpha^*_i y_i \sum_{j \neq i} \bar{\alpha}_j y_j K_{ij}.$$
Proof (cont.)

Since $\bar{\alpha}'$ is a maximizer for $L'$ we have

$$L'(\bar{\alpha}') \geq L'(\alpha'^*)$$

and hence we obtain

$$y_i f'(x_i) = y_i \sum_{j \neq i} \bar{\alpha}_j y_j K_{ij} \geq y_i \sum_{j \neq i} \alpha^*_j y_j K_{ij}. \quad (1)$$

On the other hand we have

$$y_i f(x_i) = y_i \sum_{j \neq i} \alpha^*_j y_j K_{ij} + \alpha^*_i K_{ii},$$

and therefore we can conclude that

$$y_i f'(x_i) \geq y_i f(x_i) - \alpha^*_i K_{ii}. \quad (2)$$
Leave-one-out bound for an SVM

For SVM classification we can also write

$$\sum_{i=1}^{\ell} \theta(\alpha_i K(x_i, x_i) - y_i f(x_i)) \leq \frac{r^2}{\rho^2}$$

where $r$ is the radius of the smallest sphere containing the Support Vectors and $\rho$ the true margin (different from the boosting margin!)
Leave-one-out bound for a kernel machine ensemble

The leave-one-out error of a kernel machine ensemble

\[ f(x) = c_1 f_1(x) + c_2 f_2(x) + \ldots + c_N f_N(x) \]

is upper bounded by

\[ \ell \sum_{i=1}^{\ell} \theta \left( \sum_{n=1}^{N} (\alpha_i K^{(n)}(x_i, x_i)) - y_i f(x_i) \right) \]
Leave-one-out bound for an SVM ensemble (Evgeniou et al., 2000)

For an SVM ensemble, the leave-one-out error can be bounded using geometry

$$\sum_{i=1}^{\ell} \theta(\sum_{n=1}^{N} (\alpha_i K^{(n)}(x_i, x_i)) - y_i f(x_i)) \leq \sum_{n=1}^{N} \frac{r^2(n)}{\rho^2(n)}$$

where $r(n)$ is the radius of the smallest sphere containing the SVs of machine $n$ and $\rho(n)$ the margin of SVM $n$. This suggests that bagging SVMs can be a good idea!
Recent developments (Evgeniou et al, 2001)

Through a modified version of the notion of stability, it is possible to study conditions under which **bagging** should or should not improve performances...
The original Boosting (Schapire, 1990)

1. Train a first classifier $f_1$ on a training set drawn from a probability $p(x, y)$. Let $\epsilon_1$ be the obtained performance;

2. Train a second classifier $f_2$ on a training set drawn from a probability $p_2(x, y)$ such that it has half its measure on the event that $h_1$ makes a mistake and half on the rest. Let $\epsilon_2$ be the obtained performance;

3. Train a third classifier $f_3$ on disagreements of the first two – that is, drawn from a probability $p_3(x, y)$ which has its support on the event that $h_1$ and $h_2$ disagree. Let $\epsilon_3$ be the obtained performance.
Main result: If $\epsilon_i < p$ for all $i$, the boosted hypothesis

$$f = \text{MajorityVote} (f_1, f_2, f_3)$$

has performance no worse than $\epsilon = 3p^2 - 2p^3$
Adaboost (Freund and Schapire, 1996)

The idea is of **adaptively** resampling the data

- Maintain a probability distribution over training set;

- Generate a sequence of classifiers in which the “next” classifier focuses on sample where the “previous” classifier failed;

- **Weigh** machines according to their performance.
Adaboost (pseudocode)

Given a learning method that can use weights on the data, initialize the distribution as $P_1(i) = 1/\ell$. Then, for $n = 1, \ldots, N$: 

1. Train a machine with weights $P_n(i)$ and get $f_n$; 

2. Compute the weighted error $\epsilon_n = \sum_{i=1}^{\ell} P_n(i) \theta( -y_i f_n(x_i) )$; 

3. Compute the importance of $f_n$ as $\alpha_n = 1/2 \ln \left( \frac{1-\epsilon_n}{\epsilon_n} \right)$; 

4. Update the distribution $P_{n+1}(i) \propto P_n(i) e^{-\alpha_n y_i f_n(x_i)}$. 
Adaboost (cont.)

Adopt as final hypothesis

\[ f(x) = \text{sign} \left( \sum_{n=1}^{N} \alpha_n f_n(x) \right) \]
Theory of Boosting

We define the margin of \((x_i, y_i)\) according to the\textit{ real value} function \(f\) to be

\[
\text{margin}(x_i, y_i) = y_i f(x_i).
\]

Note that this notion of margin is \textbf{different} from the SVM margin. This defines a margin for each training point!
A first theorem on boosting

**Theorem** *(Schapire et al, 1997)*

If running adaboost generates functions with errors:

\[ \epsilon_1, \ldots, \epsilon_N \]

Then for \( \forall \gamma \)

\[
\ell \sum_{i=1}^{\ell} \theta(\gamma - y_i f(x_i)) \leq \prod_{n=1}^{N} \sqrt{4\epsilon_n^{1-\gamma}(1 - \epsilon_n)^{1+\gamma}}.
\]

Thus, the training margin error drops exponentially fast if \( \epsilon_n < 0.5 \)
A second theorem on boosting

Let $H$ be an hypothesis space with VC-dimension $d$ and $C$ the convex hull of $H$

\[
C = \left\{ f : f(x) = \sum_{h \in H} \alpha_h h(x) \mid \alpha_h \geq 0; \sum h \alpha_h = 1 \right\}
\]

**Theorem (Schapire et al, 1997)**

For $\forall f \in C$ and $\forall \gamma > 0$:

\[
I[f] \leq \sum_{i=1}^{\ell} \theta(\gamma - y_i f(x_i)) + O\left(\frac{d}{\ell \gamma}\right).
\]

This holds for any voting method!
Are these theorems really useful?

- The first theorem simply ensures that the training error goes to zero...

- The second gives a loose bound which does not account for the success of boosting as a learning technique...
Additive Logistic Regression  
(Friedman, Hastie, Tibshirani 1999)

A possibly better insight can be gained by interpreting particular versions of adaboost as fitting an additive model using a certain loss function.

For example, in the discrete case \( f_n \in \{-1, 1\} \), it can be shown that adaboost builds an additive logistic regression model via Newton-like updates for approximately minimizing the functional

\[
I[f] = \int e^{-yf(x)} p(x, y) dx dy.
\]
Additive Logistic Regression (cont.)

The functional $I[f]$ is minimized at

$$f(x) = \frac{1}{2} \log \frac{P(y = 1|x)}{P(y = -1|x)}.$$  

Hence,

$$P(y = 1|x) = \frac{e^{f(x)}}{e^{-f(x)} + e^{f(x)}}$$

$$P(y = -1|x) = \frac{e^{-f(x)}}{e^{-f(x)} + e^{f(x)}}$$

Note that the usual logistic transform would not have the factor $1/2$. 
Why this loss? (Shapire and Singer, 1998)

The loss

\[ V(f(x), y) = e^{-yf(x)} \]

- is a differentiable upper bound to the 0 – 1 loss
- it has similar flavor to the SVM loss

Where is the regularizing term in this case?