COMMITTEE MACHINES

CS6140

Predrag Radivojac
Khoury College of Computer Sciences
Northeastern University

Spring, 2020
**Motivation**

**Given:** a set of observations \( \mathcal{D} = \{(x_i, y_i)\}_{i=1}^{n} \), \( x_i \in \mathcal{X}, y_i \in \mathcal{Y} \)

**Objective:** learn the posterior \( p(y|x, \mathcal{D}) \)

**Optimal Bayes Model:**

\[
p(y|x, \mathcal{D}) = \sum_{f \in \mathcal{F}} p(y|x, \mathcal{D}, f)p(f|x, \mathcal{D})
\]

Finite \( \mathcal{F} \)

**Idea:**

Don’t just train a single model, train multiple models. Average the outputs in some way.
MOTIVATION

Given: a set of observations $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{n}$, $x_i \in \mathcal{X}$, $y_i \in \mathcal{Y}$

Objective: learn $s(x|\mathcal{D})$ to approximate the posterior $p(y|x, \mathcal{D})$

The problem of local optima with rich hypothesis spaces:

- every time we train, we find a different local optimum; i.e. $s'(x|\mathcal{D})$

$$s(x) = \sum_{s' \in \mathcal{S}_{\text{strong}}} w_{(s',x)} s'(x|\mathcal{D})$$  

Finite $\mathcal{S}_{\text{strong}}$

The problem of weak hypothesis space $\mathcal{S}_{\text{weak}}$:

- we can only find a weak learner $s'(x)$

$$s(x) = \sum_{s' \in \mathcal{S}_{\text{weak}}} w_{(s',x)} s'(x|\mathcal{D})$$  

Finite $\mathcal{S}_{\text{weak}}$
TWO WAYS OF AVERAGING

Static structures:
  pre-trained models, applied independently of $x$
  examples: bagging, boosting, random forests

Dynamic structures:
  pre-trained models, applied depending on $x$
  examples: mixtures of experts

STATIC STRUCTURE

One way to average:

\[ s(x) = \frac{1}{M} \sum_{i=1}^{M} s_i(x) \]
**WHY SHOULD IT WORK?**

**Given:** \( \mathcal{D} = \{(x_i, y_i)\}_{i=1}^{n}, \ x_i \in \mathcal{X}, y_i \in \mathcal{Y} \)

**Idea:**

Let \( f_A(x) = \mathbb{E}[f(x|D)] \) be an “averaged” model. Then, for a fixed \( x \)

\[
\mathbb{E}[(y - f(x|D))^2] = y^2 - 2y\mathbb{E}[f(x|D)] + \mathbb{E}[f^2(x|D)]
\]

\[
\mathbb{E}[(y - f(x|D))^2] \geq (y - f_A(x))^2
\]

Jensen’s inequality

\[
\mathbb{E}[Z^2] \geq \mathbb{E}^2[Z]
\]
**Bagging**

**Bagging:** Bootstrap aggregating

**Approach:**

Create $B$ bootstrap samples $D_b$, where $b = 1, 2, ..., B$

Train a model $f_b(x)$ for each $D_b$

Let the final decision $f(x)$ be a majority vote by $\{f_1(x), \ldots, f_B(x)\}$

**Averaging:**

Average soft predictions then threshold the model.

Use majority vote (original idea by Leo Breiman).
Boosting

Idea: Average weak models to create strong models.

Models trained sequentially on different distributions.

Two approaches:

Boosting by filtering.

Boosting by subsampling or reweighting.
BOOSTING BY FILTERING

Idea: Construct a committee of 3 experts.

1) Expert $f_1$ is trained on $n_1$ examples picked randomly

2) Expert $f_2$ is trained as follows

2.1. Flip a coin

Heads $\rightarrow$ pass examples through $f_1$ until one is misclassified

include that example in training set for $f_2$

Tails $\rightarrow$ pass examples through $f_1$ until one is correctly classified

include that example in training set for $f_2$

2.2. Continue until $n_1$ examples are collected, then train $f_2$
**BOOSTING BY FILTERING**

**Idea:** Construct a committee of 3 experts.

3) Expert $f_3$ is trained as follows

3.1. Select $n_1$ examples by keeping those where $f_1$ and $f_2$ disagree

Data distributions are different for training $f_1$, $f_2$, and $f_3$

The final prediction obtained by majority voting.

It can be proved that if $f_1, f_2, f_3$ all have error rate of $\epsilon < \frac{1}{2}$

then the committee machine has error $g(\epsilon) = 3\epsilon^2 - 2\epsilon^3$

**Idea:** Repeat the process recursively.
Algorithm 1 AdaBoost algorithm. Typically, $T = 100$.

Input:
- $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^n$, where $x_i \in \mathcal{X}$ and $y_i \in \mathcal{Y}$.
- Weak learning algorithm $a$ that maps $\mathcal{X}$ to $\mathcal{Y}$
- Positive integer $T$

Initialization:
- Initialize sampling distribution $p^{(1)}(i) = \frac{1}{n}$ for $\forall i \in \{1, 2, \ldots, n\}$

Loop:
- for $t = 1$ to $T$
  - Sample data set $\mathcal{D}^{(t)}$ from $\mathcal{D}$ according to $p^{(t)}(i)$
  - Learn model $f_t(x)$ from $\mathcal{D}^{(t)}$
  - Calculate error $\epsilon_t$ on training data $\mathcal{D}$ as $\epsilon_t = \sum_{i: f_t(x_i) \neq y_i} p^{(t)}(i)$
  - Set $\beta_t = \frac{\epsilon_t}{1 - \epsilon_t}$
  - Set $w_t = \ln \frac{1}{\beta_t}$
  - Set $p^{(t+1)}(i) = \frac{p^{(t)}(i)}{Z} \cdot \begin{cases} 
\beta_t & \text{if } f_t(x_i) = y_i \\
1 & \text{otherwise}
\end{cases}$, where $Z$ is a normalizer
- end

Output:
- $f(x) = \arg \max_{y \in \mathcal{Y}} \left( \sum_{t=1}^{T} w_t \cdot I(f_t(x) = y) \right)$
**AdaBoost**

**Theorem.** Assume $\epsilon_t < \frac{1}{2}$ and let $\gamma_t = \frac{1}{2} - \epsilon_t$. Then the following bound holds

$$
\frac{1}{n} |\{i : f(x_i) \neq y_i\}| \leq \prod_{t=1}^{T} \sqrt{1 - 4\gamma_t^2} \leq e^{-2 \sum_{t=1}^{T} \gamma_t^2}
$$

**In practice:**

Training error $\to 0$, but the test error continues to decrease

AdaBoost theory shows relationship to SVMs

Minimizing $\epsilon_t$ is equivalent to minimizing $E = \sum_{i=1}^{n} e^{-y_i f(x_i)}$
Random Forests

Idea. Construct an ensemble of trees, 100 to 1000.

In practice:

Randomize dataset
   ○ bootstrap the dataset

Randomize features upon which we split
   ○ at each node, keep e.g. $F = \log_2 d + 1$ features, then split
EXPERIMENT: BAGGING

Data set:
- binary classification
- a unit-radius circle within a $4 \times 4$ square
- 200 examples, added a small amount of noise

Models:
- neural networks
- regression trees
- w/ and w/o bagging

$x = [-2, 2] \times [-2, 2]$
$Y = \{-, +\}$
**EXPERIMENT: BAGGING**

Data set:
- a unit-radius circle within a $4 \times 4$ square
- $n_+ = 100$, $n_- = 100$
- 5% error in positives, 10% error in negatives

Models:
- single-output two-layer neural networks
- $h$ hidden neurons, tanh($x$) activation
- RPROP optimization

![Three graphs showing $s(x)$ for different $h$ and $B$ values.](image-url)
EXPERIMENT: BAGGING

Data set:
- a unit-radius circle within a $4 \times 4$ square
- $n_+ = 100$, $n_- = 100$
- 5% error in positives, 10% error in negatives

Models:
- single-output two-layer neural networks
- $h$ hidden neurons, tanh($x$) activation
- RPROP optimization
EXPERIMENT: BAGGING

Data set:
- a unit-radius circle within a $4 \times 4$ square
- $n_+ = 100$, $n_- = 100$
- 5% error in positives, 10% error in negatives

Models:
- single-output two-layer neural networks
- $h$ hidden neurons, tanh($x$) activation
- RPROP optimization

$s(x)\, h = 4$

$s(x)\, h = 4, B = 100$
**EXPERIMENT: BAGGING**

Data set:
- a unit-radius circle within a $4 \times 4$ square
- $n_+ = 100$, $n_- = 100$
- 5% error in positives, 10% error in negatives

Models:
- regression trees

![Graphs showing $s(x)$ and $s(x), B = 100$]
PROBABILISTIC GENERATIVE MIXTURE MODELS

Model:
- $x$ drawn according to $p(x)$
- pick model $k$ to generate target according to $p(k|x)$
- generate target using a linear model with additive zero-mean error
  e.g., $Y = \sum w_{kj} X_j + \epsilon_k$, where $\epsilon_k \sim \mathcal{N}(0, \sigma_k^2)$

$$p(y|x, \theta) = \sum_{k=1}^{K} p(y|x, w_k)p(k|x)$$
MIXTURE OF EXPERTS