Adaboost (resample)

- Boosting by filtering (takes too many examples)
- Adaboost use same training examples, but change distribution on which you sample
- Adaboost is simple to implement and used on many applications and different types of learning machines
Boosting Algorithm

- **Input:** \( S = ((x(i), d(i)), 1 \leq i \leq m) \)
- **Initialization:** Choose distribution \( D_1 \) that picks inputs equally likely and let \( 0 < \gamma < \frac{1}{2} \) be the weak learning rate (i.e. algorithm produces an error rate less than \( \frac{1}{2} - \gamma \))
- **Iterate:** for \( n=1 \ldots T \)
  - Call weak learning algorithm \( L \) with examples chosen from distribution \( D_n \)
  - Get outputs from \( h_n : X \rightarrow Y \)
  - Calculate error \( \varepsilon_n = \frac{1}{2} - \gamma_n \leq \frac{1}{2} - \gamma \)
  - Update \( D_{n+1} \) based on training errors
- **Produce hypothesis** \( h \) from \( h_1, \ldots, h_T \)
Adaboost algorithm

- Choosing $D_n$
  
  $$D_{n+1}(x(i)) = D_n(x(i)) \exp \left( - \alpha_n h_n(x(i))d(i) \right) / Z_n$$

  where $Z_n$ is a normalization constant.

- Output hypothesis
  
  $$h(x) = \text{sgn} \left( \sum_{n=1:T} \alpha_n h_n(x) \right)$$

- Last time showed that empirical error decays exponentially as $T$, the number of times examples are resampled.

- Examine generalization error
Generalization Error using VC bounds

- Using VC bounds can find upper bounds on generalization error

  \[ P(d \neq h(x)) \leq \text{error}(h(S)) + O((Td/m)^{1/2}) \]

  where \( d \) is VC dimension of concept class.

- Error depends on \( T \), the number of times resampling is done, but experimental results indicate that generalization error for boosting decreases as \( T \) increases. Schapire shows how generalization error decreases even when training error is zero.
Generalization error using margin bounds

- Let $\alpha_i$ be a sequence of nonnegative constants with $\sum_{i=1}^{m} \alpha_i = 1$. Let $f(x) = \sum_{n=1}^{T} \alpha_n h_n(x)$, then margin is $\text{mar}_f(x,d) = f(x)d$. Then $|\text{mar}_f(x,d)| \leq 1$.

- Boosting works on examples that have small positive margin or negative margin. It increases the margin on the examples that have the smallest margin.

- $P(d \neq h(x)) \leq P(\text{mar}_f(x,d) \leq \theta) + O((d/(m\theta^2)^{1/2})$

Here generalization error bound is independent of $T$. 

References


Bayesian Methods

- Information about knowledge formulated probabilistically
  - Model defined with unknown parameters
  - Specify prior distribution
- Gather data
- Compute posterior distribution
- Use posterior distribution to: make predictions, make decisions, reach conclusions
Finding Posterior Distribution

- Bayes Rule
  \[ P(\text{param.} | \text{data}) = \frac{P(\text{data} | \text{param.})P(\text{param.})}{P(\text{data})} \]
- Posterior \(\propto\) Likelihood \(\times\) Prior
- To make predictions on new data
  \[ P(\text{new data} | \text{data}) = \int P(\text{new data} | \text{param.})P(\text{param.} | \text{data}) \]
Representing Priors and Posterior Dist.

- Priors and posterior distributions often have complex distributions that are not easily represented
- Represent distributions using samples
  - Obtaining a sample from priors
  - Obtaining a sample from posterior distribution (more difficult)
- Example: A Hard Linear Classifier (Radford Neal, NIPS 2004 tutorial)
Comparison of Bayesian Learning to other methods

- Bayesian learning not tied down to specific network architecture (model, priors, posterior)
- Learning theory: architecture pre-specified with parameters tuned to training data
- Bias versus Variance \((\text{MSE} = (\text{Bias})^2 + \text{Variance})\)
  - Bias is deviation from average value (model not complex enough to realize training data)
  - Variance is average squared error away from average (model too complex and data is overfit)
Choosing a Model and Priors

- Model should be ubiquitous to consider practically all possibilities and may contain latent variables.
- Priors should consider a wide range of possibilities, but avoid being too spread out and may contain hyperparameters.
- The model and priors are tested based on prior knowledge and beliefs. If they do not fit, then model and prior are adjusted.
- Process involves testing model and priors, revisions, and retesting (should not be influenced by data).
Computing Posterior Distributions

Different approaches to computing posterior distribution

- **Analytical integration:** may not be possible or feasible
- **Gaussian approximation:** can work well when there is a lot of data
- **Monte Carlo integration:** simple MC (sample from posterior), importance sampling, Markov Chain Monte Carlo (MCMC).
- **Variational approximations**
Gaussian Priors

- Assume priors are Gaussian, then can assume an underlying network model exists or consider Gaussian processes.
- Underlying network model: multi-layer perceptron, Support Vector Machine. In limit, from Central Limit Theorem outputs may still be approximately Gaussian.
- For Gaussian processes the key is to learn the mean and covariance functions.
Bayesian Framework

- Repeated use of Bayes rule as hyperparameters are adjusted at different levels
- Can use network model or Gaussian processes
- Use Gaussian priors or mixture model
- Regularization incorporated in Bayesian framework
- Procedure can be time consuming as hyperparameters are continually adjusted for different network models and parameters
Bayesian Learning Methodology Example

1) Initialize $\alpha$ and $\beta$ where $J(w) = \frac{1}{2}\alpha w^Tw + \frac{1}{2} \beta e^Te$ based on priors and previous estimate performance

2) Use optimization algorithm to train network to minimize $J(w)$ based on given $\alpha$ and $\beta$ and go to 1)

3) Use different set of model parameters (e.g. modify parameters of kernel) and go to 1)

4) Use different network models and go to 1)
References