

| Weighted Majority |
| :---: |
| - Assume K experts $f_{1}, f_{2}, \ldots f_{k}$ (base learners) |
| - Each expert makes a decision $f_{k}(\mathbf{x})= \pm 1$ |
| - Improve predictions by making the experts |
| "vote" according to how good they are: |
| $F(\mathbf{x})=\Sigma_{k} \alpha_{k} f_{k}(\mathbf{x})$ |
| $0 \leq \alpha_{k} \leq 1$ |
| - Decision: $\operatorname{sign}[F(\mathbf{x})]$ |



## Simple Examples

- Kernel methods:
- Each example is an "expert"
$f_{i}(\mathbf{x})=y_{i} k\left(\mathbf{x}, \mathbf{x}_{\mathrm{i}}\right)$
- Decision stumps:
- Each variable is an "expert"
$\mathrm{f}_{\mathrm{j}}(\mathbf{x})=\mathrm{x}_{\mathrm{j}} \quad$ ("orient" variables s.t. $\left[\mathrm{x}_{1}, \mathrm{x}_{2}, \ldots \mathrm{x}_{\mathrm{m}}\right] \mathbf{y}>0$ )
- Each feature is an "expert"
$\mathrm{f}_{\mathrm{j}}(\mathbf{x})=\phi_{\mathrm{j}}(\mathbf{x})$


## Bias: $\left[\mathrm{E}_{\mathrm{D}} \mathrm{f}(\mathbf{x}, \mathrm{D})-\mathrm{y}\right]^{2}$

- $\mathrm{E}_{\mathrm{D}} \mathrm{f}(\mathbf{x})$ : your "ideal" committee machine.
- Bias: what your ideal committee can't learn (from $m$ training examples)
- $E_{D} f(\mathbf{x})$ has the same bias as $E_{D} f(x)$ but no variance.
- Note: Each committee member was trained on a different set on $m$ examples...



## Variance: $\mathrm{E}_{\mathrm{D}}\left[\mathrm{f}(\mathbf{x}, \mathrm{D})-\mathrm{E}_{\mathrm{D}} \mathrm{f}(\mathbf{x}, \mathrm{D})\right]^{2}$

- $\mathrm{E}_{\mathrm{D}} \mathrm{f}(\mathrm{x})$ : your "ideal" committee machine.
- Variance : how far apart on average your solution $f(x, D)$ is from your "ideal" committee machine.
- If the variance is high but the bias is low: there is hope that a committee can improve performance.
- Note: Subsampling introduces extra bias...

| Feature Selection |
| :---: |
| 1) Mixture of decision stumps $\Leftrightarrow$ feature selection <br> 2) Merging expert feature rankings: <br> - Average ranking index: $\mathrm{C}_{\mathrm{j}}=\Sigma_{k} \alpha_{k} \mathrm{C}_{\mathrm{jk}}$ <br> - Average rank: $\mathrm{C}_{\mathrm{j}}=\Sigma_{\mathrm{k}} \mathrm{o}_{\mathrm{k}}\left(\mathrm{R}_{\text {max }}-\mathrm{R}_{\mathrm{j} k}\right)$ <br> 3) Merging feature sets selected by experts: <br> - Ranking index: $\mathrm{C}_{\mathrm{j}}=\Sigma_{\mathrm{k}} \alpha_{\mathrm{k}} \delta_{\mathrm{k}} \quad\left(\delta_{\mathrm{k}}=1\right.$ if feat j selected by expert $k$, 0 otherwise) <br> - $\mathrm{S}^{*}=\operatorname{argmax}_{k} \min _{k}\left\|\mathrm{~S}_{\mathrm{k}} \cap \mathrm{S}_{\mathrm{k}}\right\|$ <br> (most "stable" subset) <br> - $\mathrm{R}^{*}=\operatorname{argmin}_{k}$ max $_{k} \operatorname{dist}\left(\mathrm{R}_{\mathrm{k}}, \mathrm{R}_{\mathrm{k}}\right)$ <br> htto://peopoperevoleducom/kardi/utoria//Similarity/OrdinalVariables. $\mathrm{htm} \mid$ <br> 4) Sensitivity-based (special for bagging) |


|  |
| :---: |
| Bayesian Approach |
|  |
|  |
|  |




## Difficulties

- Continuous case:

Infinitely many experts, we can't try them all!

- Idea:

Let's take a sample...

- How?

Grid, heuristic search, stochastic search

- Important:

Avoid sampling "poor" experts or "redundant" experts.


Variable-dimension MCMC

Vehtari and Lampinen, 2002

- Some steps include removal or addition of a feature
- We obtain P(model,feature-subset|D) for some samples of models and feature subsets
- Subset relevance can be computed by marginalization (averaging over the functions using the same subset)
- Feature relevance can also be computed by marginalization (averaging over all subsets containing that feature)

| Performance Gain? |
| :--- |
| - If we draw M classifiers $f_{k}$ according to $P(f \mid D)$, |
| we can approximate |
| $P(y \mid \mathbf{x}, \mathrm{D})=\int_{\mathrm{f}} \mathrm{P}(\mathrm{f} \mid \mathrm{D}) \mathrm{P}(\mathrm{y} \mid \mathbf{x}, \mathrm{f}, \mathrm{D})$ df |
| by |
| $\mathrm{P}(\mathrm{y} \mid \mathbf{x}, \mathrm{D}) \sim \sum_{\mathrm{k}=1: \mathrm{M}} \mathrm{P}\left(\mathrm{y} \mid \mathbf{x}, \mathrm{f}_{\mathrm{k}}, \mathrm{D}\right)$ |
| - Relative error difference with optimum Bayes |
| classifier decays with $\mathrm{O}(1 / \mathrm{M})(\mathrm{Ng}$, Jordan, 2001) |


| Bagging |
| :---: |
| Breiman, 1996 |
| - Bootstrap Aggregation: |
| - Draw with replacement m samples from |
| the original training set of size m |
| - Train a learning machine |
| - Repeat many time |
| - On average, each example appears in the |
| training set $(1-1 / \mathrm{m})^{\mathrm{m} \sim 1-\mathrm{e}^{-1} \sim 0.632 \text { times }}$ |

Relative error difference with optimum Bayes classifier decays with $\mathrm{O}(1 / \mathrm{M})(\mathrm{Ng}$, Jordan, 2001)

Non-Bayesian Approaches

- Parallel ensembles: bagging
- Serial ensembles: boosting

|  |
| :---: |
| Non-Bayesian Approaches |

- Bootstrap Aggregation:
- Draw with replacement m samples from the original training set of size m
- Train a learning machine
- Repeat many time
- On average, each example appears in the training set $(1-1 / \mathrm{m})^{\mathrm{m}} \sim 1-\mathrm{e}^{-1 \sim} \sim 0.632$ times


## Random Forests

Breiman, 2001

1. A number n is specified much smaller than the total number N of variables (typically $\mathrm{n} \sim$ sqrt(N))
2. Each tree of maximum depth is grown on a bootstrap sample of the training set
3. At each node, $n$ variables are selected at random out of the $N$
4. The split used is the best split on these $n$ variables


## Embedded Variable Scoring

- $I G_{t}(f)=$ Information gain due to splitting with feature $f$ at node $t$
- Ranking index: $\mathrm{R}(\mathrm{f})=\Sigma_{\mathrm{t}} \mathrm{IG}_{\mathrm{t}}(\mathrm{f})$
- Surrogate variables (detect masking)
- Use of $M$ trees:

$$
R(\mathrm{f})=\Sigma_{\mathrm{T}} \Sigma_{\mathrm{t} \in \mathrm{~T}} \mathrm{IG}_{\mathrm{t}}(\mathrm{f})
$$



## Sensitivity-based Scoring

Breiman, 2001

- Classify the OOB cases and count the number of votes cast for the correct class in every tree grown in the forest
- Randomly permute the values of feature $f$ in the OOB cases and classify these cases down the trees
- Subtract the number of votes for the correct class in the feature-f permuted OOB data from the untouched OOB data
- Average this number over all trees in the forest to obtain the importance score $\mathrm{R}(\mathrm{f})$


## Cross-validated Committee

Parmanto et al., 1996

- Any learning machine
- Any method of splitting the (training) data many times into training set and validation set (vset)
- Perturb feature f randomly in vset (pvset)
- $\mathrm{R}(\mathrm{f})=$ mean[ num-correct-class(vset) -num-correct-class(pvset) ]
- Zscore = R(f)/stderror

| Boosting |
| :---: |
| - Adaboost (Freund and Schapire, 1996): At every step add a new base learner that is forced (by re-weighting the training data) to concentrate on misclassified examples. <br> - Forward stagewise boosting (Breiman, 1997, Friedman et al., 2000) <br> 1. Initialize $F(\mathbf{x})=0$ <br> 2. For $k=1$ to $M$ $\begin{aligned} & \mathrm{F}(\mathbf{x}) \leftarrow \mathrm{F}(\mathbf{x})+\alpha \mathrm{f}(\mathbf{x}) \\ & \left(\alpha_{\mathrm{k}}, \mathrm{f}_{\mathrm{k}}\right)=\operatorname{argmin}_{\alpha, \mathrm{f}} \Sigma_{i} \exp \left(-\mathrm{y}_{\mathrm{i}} \mathrm{~F}\left(\mathbf{x}_{\mathrm{i}}\right)\right) \end{aligned}$ <br> 3. Output $F(x)=\sum_{k=1: M} \alpha_{k} f_{k}(\mathbf{x})$ |



| Conclusion |
| :---: |
| - Ensemble methods help reducing the |
| "variance" |
| - They benefit most to "low bias" base |
| learners |
| - One should not confuse feature set |
| variability and variance in predictions |
| - CV committees allow to rank features |
| according to sensitivity ans compute |
| zscores. |


| Exercise Class |
| :---: | :---: |
| Arcene |
| Boosting |


| Arcene |
| :---: |
|  |
| Best performances: $\mathbf{1 1 . 9 \%} \pm \mathbf{1 . 2}$ (use training set + validation set in training) with $11 \%$ of the features ( 1100 features) |
| Baseline model: $\mathbf{1 4 . 7 \%} \pm \mathbf{1 . 4}, 1100$ features <br> my_svc=svc(f'coef0=1', 'degree=3', 'gamma=0', 'shrinkage=0.1'\}); |
| my_model=chain(fstandardize, s2n('f_max=1100'), normalize, my_svct) |


| Forward Stagewise Boosting |
| :---: |
| 1. Initialize $F(\mathbf{x})=0$ <br> 2. For $k=1$ to $M$ $\begin{aligned} & F(\mathbf{x}) \leftarrow F(\mathbf{x})+\alpha f(\mathbf{x}) \\ & \left(\alpha_{k}, f_{k}\right)=\operatorname{argmin}_{\alpha, f} \Sigma_{i} \exp \left(-y_{i} F\left(\mathbf{x}_{i}\right)\right) \end{aligned}$ <br> 3. Output $\mathrm{F}(\mathrm{x})=\sum_{\mathrm{k}=1: \mathrm{M}} \alpha_{k} \mathrm{f}_{k}(\mathbf{x})$ |
| At step t: $\left(\alpha_{k}, f_{k}\right)=\operatorname{argmin}_{\alpha, \mathrm{f}} \sum_{i} \exp \left[-y_{i}\left(F_{\mathrm{t}-1}\left(\mathbf{x}_{\mathrm{i}}\right)+\alpha \mathrm{f}\left(\mathbf{x}_{\mathrm{i}}\right)\right)\right]$ <br> Compute $\alpha_{k}, \mathrm{f}_{\mathrm{k}}$ for decision stumps |



