Ensemble Methods: Classification by Committee

Ensemble Methods

Ensemble methods perform classification by "pooling" or "aggregating" the results of several classifiers.

Setting 1: Hard decisions

Suppose $f_1, \dots, f_M$ are classifiers

$$f_k : \mathbb{R}^d \to \{-1, 1\}$$

that output "hard" decisions, -1 or 1.

The classifiers may be combined by taking a (weighted) majority vote

$$f_{\text{combined}}(x) = \text{sign}\left\{ \sum_{k=1}^{M} w_k f_k(x) \right\}$$

Setting 2: Soft decisions

Write

$$f_k(x) = \text{sign}(g_k(x))$$

where $g_k : \mathbb{R}^d \to \mathbb{R}$ provides a "soft" decision.

Examples

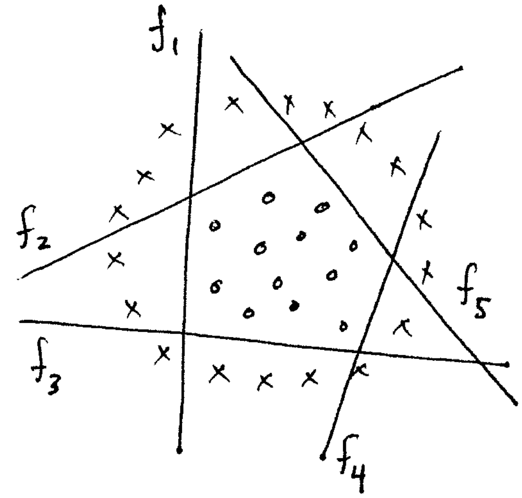
- linear classifier: $g(x) = w^T x + b$
- kernel classifier: $g(x) = \sum_{i=1}^{n} \alpha_i k(x, x_i)$

These soft decisions may be combined as

$$f_{\text{combined}}(x) = \text{sign}\left\{ \sum_{k=1}^{M} w_k g_k(x) \right\}$$

Why do this?

1. $f_1, \dots, f_M$ are too simple to be good classifiers by themselves. But if they were somehow organized to work in unison, the combined classifier could perform very well



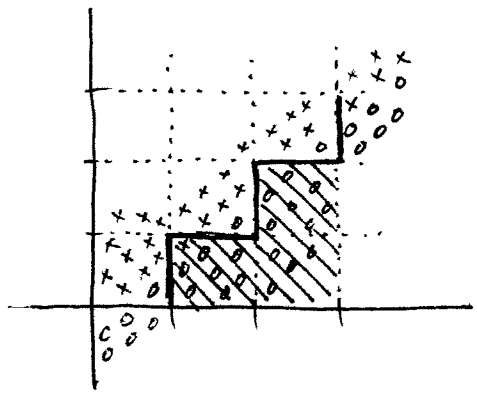
2. Variance reduction:

Suppose a classification algorithm has a high variance, meaning it is highly sensitive to slight perturbations of the training data.

If each f_k is produced by the same algorithm, but on different variations (or reweightings) of the training data, the combined classifier may have smaller variance.

Example | Histograms

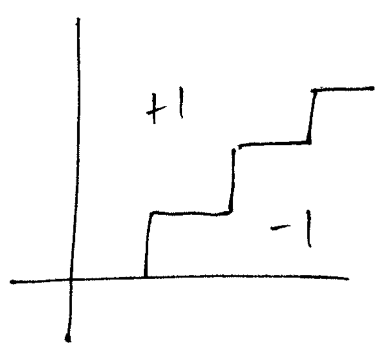
Consider a $d = 2$ dimensional setting. A regular histogram classifier with bin width 1 looks like



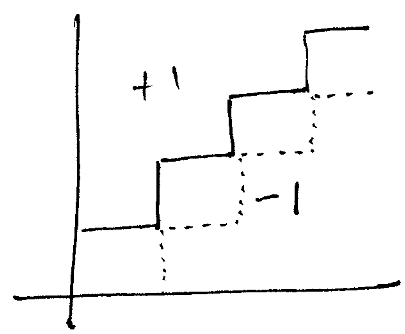
label = majority vote in each cell

The partition is fixed, irrespective of the data.
As the data gradually shift upward,
the classifier changes abruptly

from



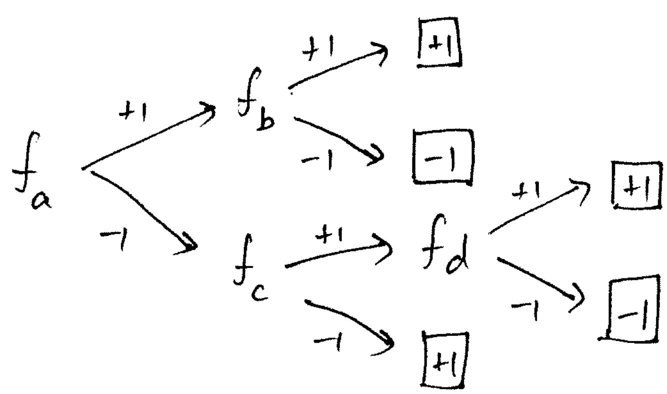
to



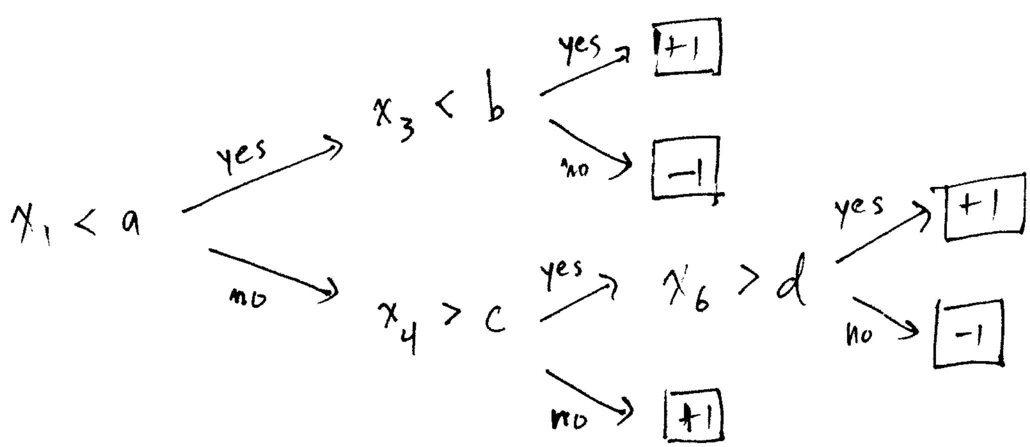
In contrast, a linear classifier would transition smoothly as the data shifts.

Example 1 Decision trees.

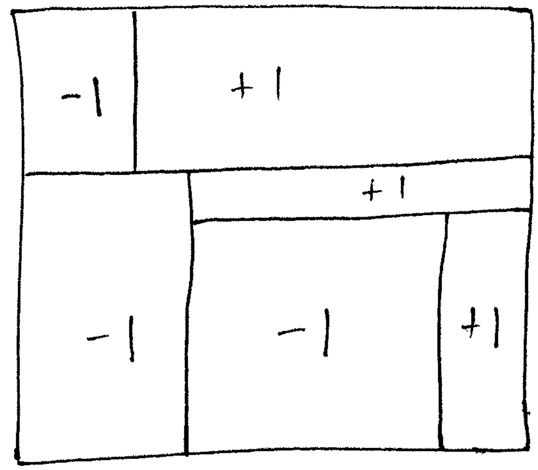
A decision tree is based on a tree-structured hierarchy of classifiers



Often the "nodes" of the tree are simple univariate splits, e.g.



In 2-d:



Like histograms, most algorithms for constructing decision trees have a high variance.

Averaged Shifted Histograms

An ensemble approach can be used to reduce the variance of the histogram classifier based on a fixed partition

Idea:

- Generate M small shifts of the data
- For each shift, form the histogram classifier
- Average the resulting classifiers

In detail

- For $k = 1, \dots, M$, generate

$$\epsilon_k = (\epsilon_{k1}, \epsilon_{k2}, \dots, \epsilon_{kd})^T$$

$\boxed{\text{deterministic or random}}$

For example, if the histogram binwidth is h, consider shifts of the form

$$(0, \dots, 0, \pm \frac{h}{2}, 0, \dots, 0)^T$$

- Let

$$X_0 = \{x_1, \dots, x_n\}, \quad x_i \in \mathbb{R}^d$$

denote the training data.

Define

$$X_k = X_0 + \epsilon_k$$
$$= \{x_1 + \epsilon_k, \dots, x_n + \epsilon_k\}$$

$\boxed{\text{shifted data}}$

- Let $f_{X_k}(x)$ be the histogram classifier based on the data X_k

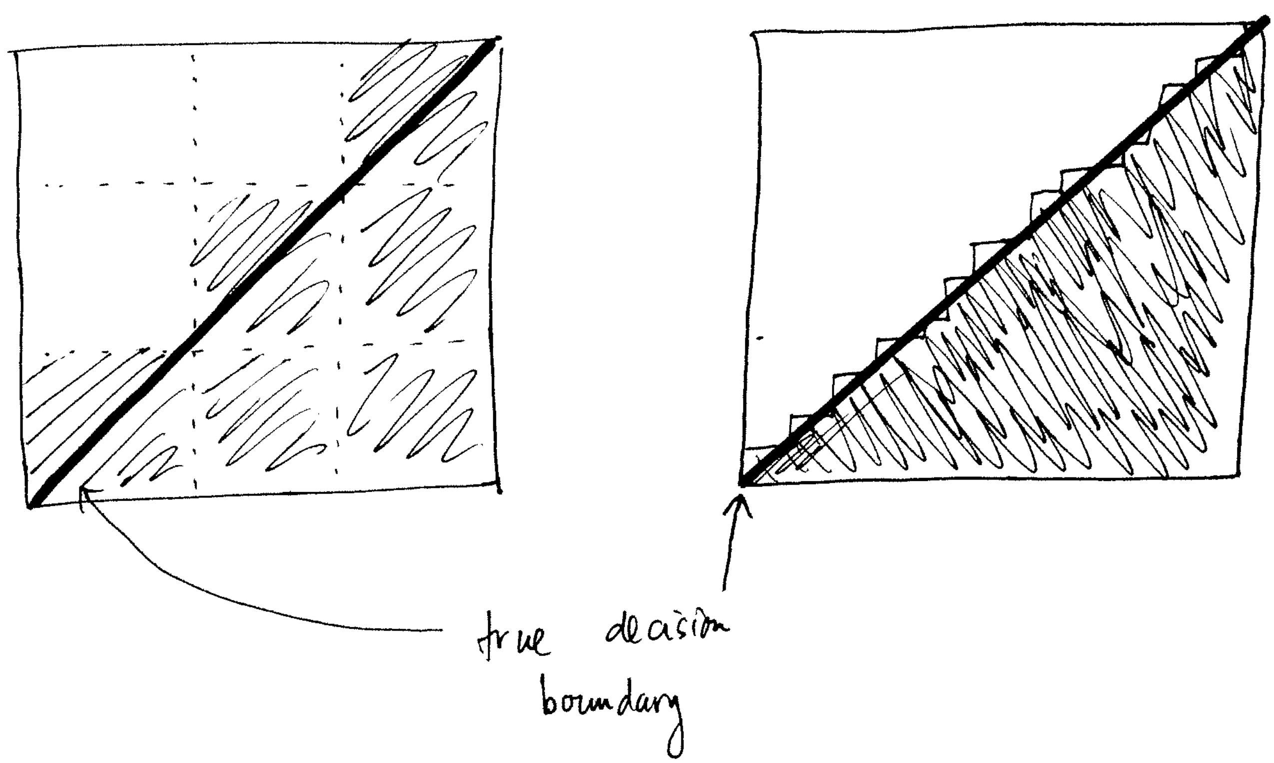
- Majority vote:

$$f_{\text{combined}}(x) = \frac{1}{M} \sum_{k=1}^{M} f_{X_k}(x)$$

The result is a classifier with much lower variance.

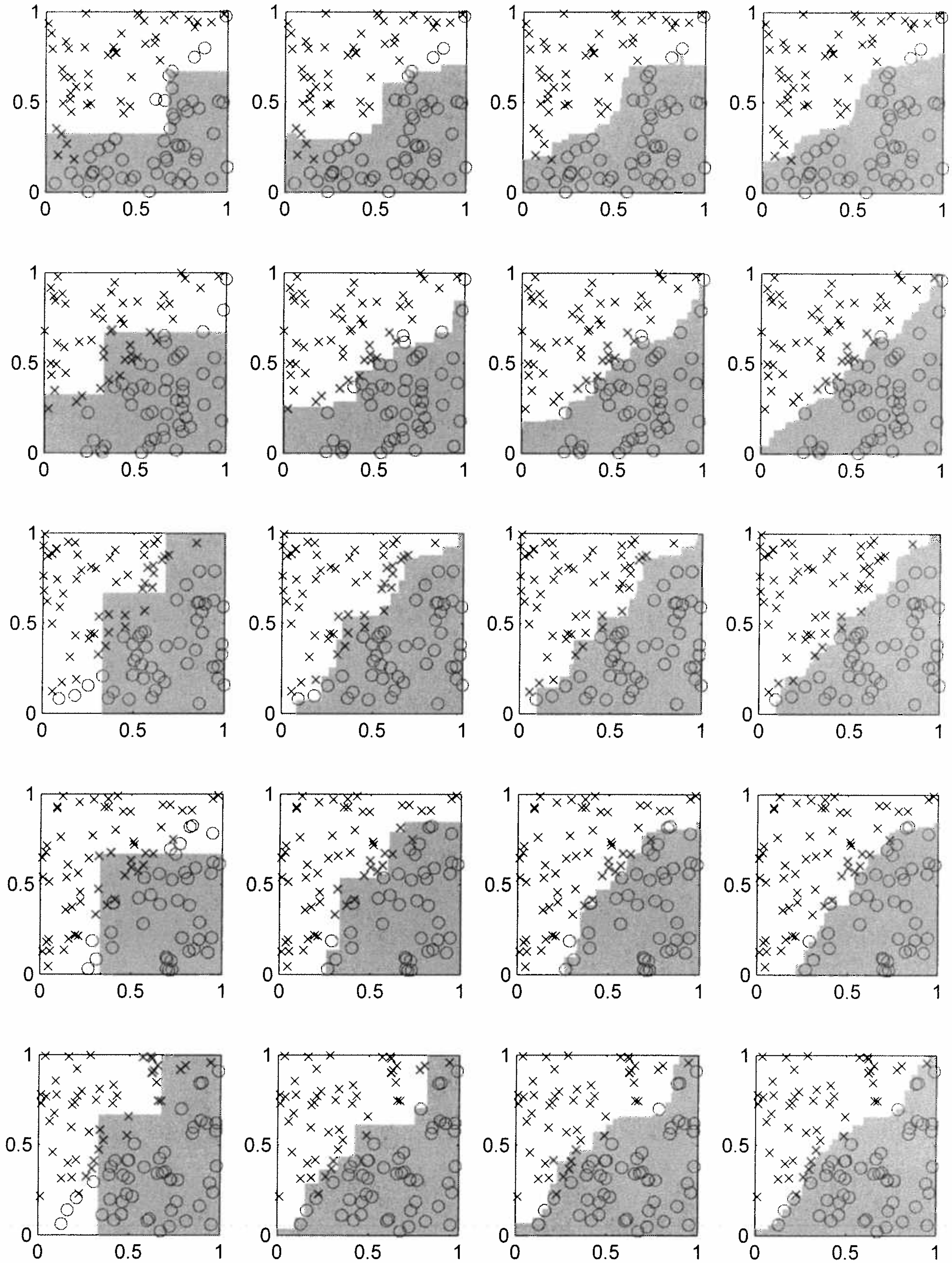
single histogram

average of many histograms



# of votes = 1 5 11 21

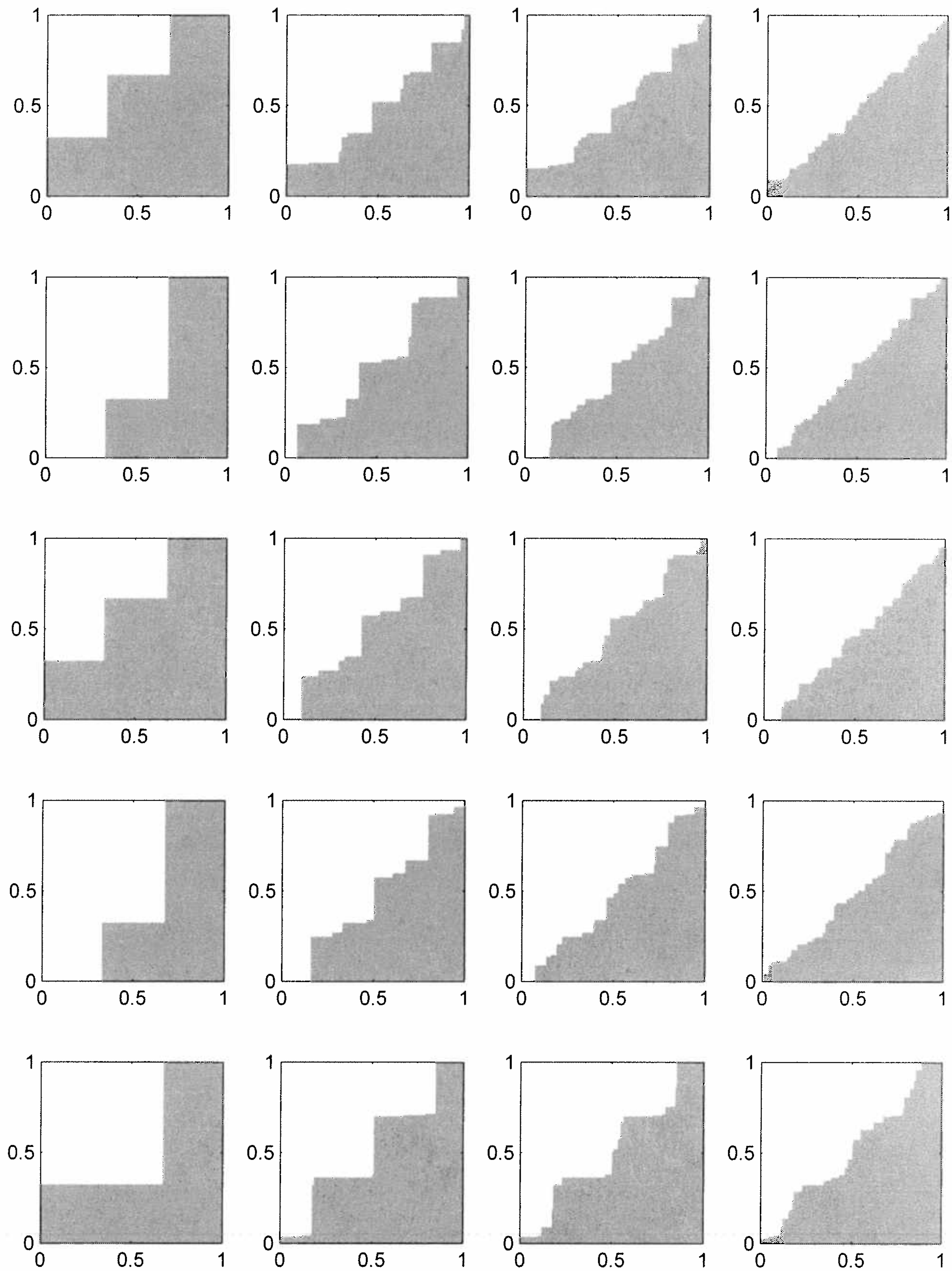
5 realizations of data



$n = 100$ points , bin width $= 1/3$

# of votes = 1   5   11   21

5 realizations of data



$n = 1000$ points, bin width $= 1/3$

$\boxed{\text{Bagging}}$

Bagging is short for bootstrap aggregation

<u>Definition</u> Let $X_0 = \{x_1, \dots, x_n\}$ be a training sample. Let $X^* = \{x_1^*, \dots, x_n^*\}$ be obtained by sampling <u>with replacement</u> from X_0. Then X^* is called a <u>bootstrap</u> sample.

<u>Idea</u>:

- Generate B bootstrap samples $X_1^*, \dots, X_B^*$.
- Let $f_{X_b^*}(x)$ be the classifier trained on X_b^*.
- Vote:

$$f_{combined}(x) = \frac{1}{B} \sum_{b=1}^{B} f_{X_b^*}(x)$$

Both the averaged shifted histograms and bagging combine <u>resampling</u> with a majority vote.

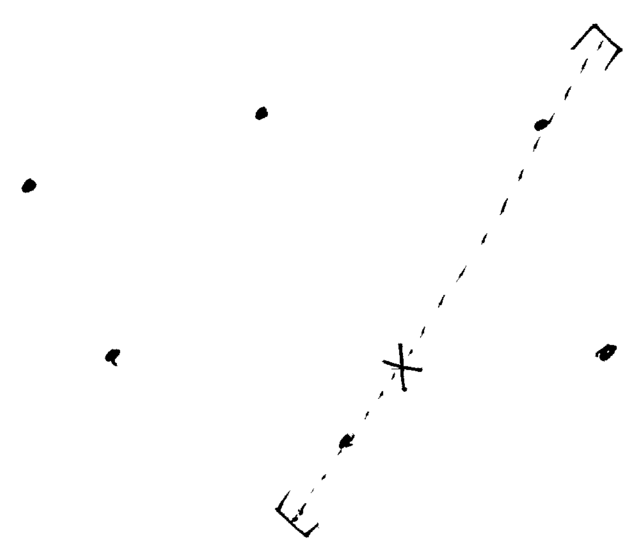
Many other resampling schemes are conceivable.

E.g., random convex combinations:

- Given $X_0 = \{x_1, \dots, x_n\}$, generate $X^* = \{x_1^*, \dots, x_n^*\}$, where x_i^* is obtained as

$$x_i^* = \lambda x_a + (1-\lambda) x_b$$

where a, b are random, $\lambda \in [-\delta, 1+\delta]$ is uniform.



Boosting

Recall there were two reasons that motivated ensemble rules: 1) combing simple rules into a complex rule; 2) variance reduction. Thus far we have only discussed the latter point.

Boosting is an ensemble rule that achieves both. It is based on the notion of a base learner.

Definition | A base learner is any classification rule such that, given any training sample $(x_1, y_1), \ldots, (x_n, y_n)$, and weights $w_1, \ldots, w_n$ $\left(w_i \geq 0, \sum_i w_i = 1\right)$, it produces a classifier f such that

$$\sum_{i=1}^{n} w_i I_{\{f(x_i) \neq y_i\}}$$

is small

In short, a base learner can learn a classifier that respects any possible weighting of the training error.

The Boosting principle

Choose an initial weighting $w^{(1)}$

- Given a weighting $w^{(t)}$, apply the base learner to generate a classifier f_t
- Upweight $w_i^{(t)}$ if $f(x_i) \neq y_i$
  Downweight $w_i^{(t)}$ if $f(x_i) = y_i$

Repeat while $t \leq T$.

Output

$$f(x) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t f_t(x) \right)$$

Where $\alpha_t > 0$ reflects the confidence in f_t.

Examples of base learners

- Decision trees. As an averaging procedure, boosting will reduce their variance.
- Decision stumps: trees consisting of a single split

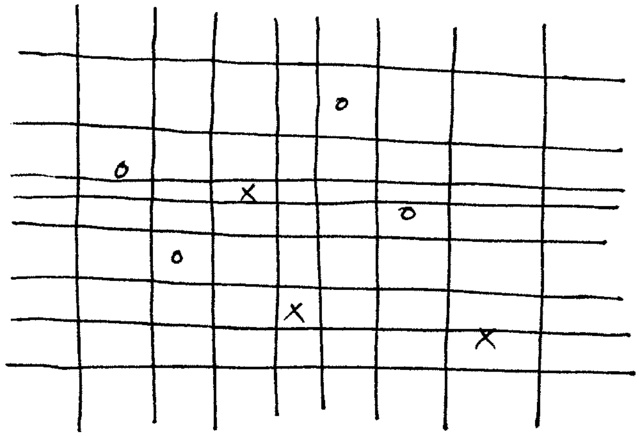
$$f(x) = \mathrm{sign}\{x_j \geq c\}$$

- Radial basis functions

$$f(x) = \mathrm{sign}\{k_\sigma(x, x_i) + b\}$$

Exercise Describe an algorithm (a base learner) that chooses the decision stump with minimal weighted training error. Ditto for RBFs.

Solution



Decision stumps

We only care how the classifier performs on the training data. There are $(n+1)\cdot d$ total stumps to consider, so we can minimize the weighted error over this finite collection by exhaustive search.

A similar strategy applies to RBFs. If σ is fixed (choosing σ is a separate problem), the total number of classifiers to consider is ______.

Adaboost

The first successful boosting algorithm was introduced by Yoav Freund and Robert Schapire, called Adaboost.

Adaboost

Given $(x_1, y_1), \ldots, (x_n, y_n)$, $y_i \in \{-1, +1\}$

Initialize $w_i^1 = \frac{1}{n}$.

For $t = 1, \ldots, T$

- Apply base learner with weights w^t to produce classifier f_t

- Set
$$r_t = \sum_{i=1}^{n} w_i^t \, I_{\{f_t(x_i) \neq y_i\}}$$

- Set
$$\alpha_t = \frac{1}{2} \ln\left(\frac{1 - r_t}{r_t}\right)$$

- Update
$$w_i^{t+1} = \frac{w_i^t \cdot \exp\{-\alpha_t y_i f_t(x_i)\}}{Z_t}$$
where Z_t is a normalization constant

End

Output
$$f(x) = \text{sign}\left\{\sum_{t=1}^{T} \alpha_t f_t(x)\right\}$$

The success of Adaboost is reflected in the following result.

<u>Theorem</u> Suppose $r_t = \frac{1}{2} - \gamma_t$, $\gamma_t \geq 0$, for each t. Then

$$\frac{1}{n}\sum_{i=1}^{n} I_{\{f(x_i) \neq y_i\}} \leq \exp\left(-2\sum_{t=1}^{T} \gamma_t^2\right)$$

In particular, if $\gamma_t \geq \gamma > 0$ for all t, then

$$\frac{1}{n}\sum_{i=1}^{n} I_{\{f(x_i) \neq y_i\}} \leq \exp\left(-2\gamma^2 \cdot T\right)$$

The assumption $\gamma_t \geq \gamma > 0 \;\; \forall t$ is sometimes called the <u>weak learning hypothesis</u>, and the base learner is called a <u>weak learner</u>.

In words, the theorem tells us if our base learner does slightly better than random guessing, the final combined classifier can separate the training data perfectly for T large enough. In fact, the error goes to zero <u>exponentially fast</u>!

Adaboost details and comments

1. If $r_t = 0$, then $\alpha_t = 0$ and the algorithm breaks down. On the other hand, if $r_t = 0$, then f_t classifies every point perfectly and there is no need to boost.

2. T must be set in some manner. Unfortunately, no satisfactory theory or method for setting T is known. In practice a couple of options are

   - Set T by cross-validation
   - Let T_0 be the number of iterations until the training error is zero. set $T = (1.1) * T_0$.

3. Empirical evidence suggests that Adaboost using decision trees for base learners is one of the best "off-the-shelf" methods for classification.

Proof of Theorem | The proof is broken down into some lemmas.

Lemma |

$$\frac{1}{n}\sum_{i=1}^{n} I_{\{f(x_i)\neq y_i\}} \leq \prod_{t=1}^{T} Z_t$$

Proof | By unraveling the update rule we find

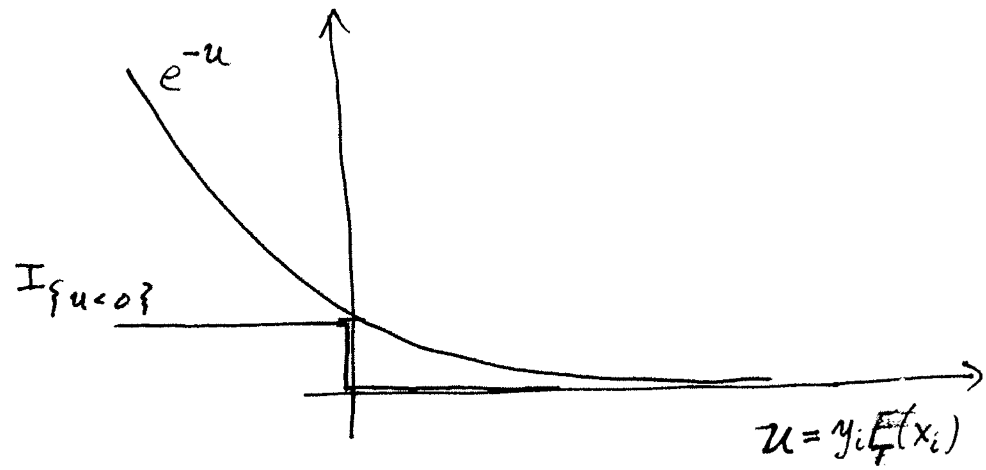
$$\begin{aligned}
W_i^{T+1} &= \frac{W_i^{T}\exp\left(-\alpha_T y_i f_T(x_i)\right)}{Z_T} \\
&= \frac{W_i^{T-1}\exp\left(-y_i\left[\alpha_{T-1} f_{T-1}(x_i) + \alpha_T f_T(x_i)\right]\right)}{Z_{T-1}\cdot Z_T} \\
&\;\;\vdots \\
&= \frac{\underbrace{\frac{1}{n}}_{\boxed{W_i^1}}\cdot \exp\left(-y_i \sum_{t=1}^{T}\alpha_t f_t(x_i)\right)}{Z_1\cdot Z_2\cdots Z_T} \\
&= \frac{\exp\left(-y_i F_T(x_i)\right)}{n\prod_{t=1}^{T} Z_t}
\end{aligned}$$

where $\quad F_t = \sum_{s=1}^{t} \alpha_s f_s$

and $\quad f(x) = \operatorname{sign}\{F_T(x)\}$

Now use the bound

$$I_{\{f(x_i) \neq y_i\}} = I_{\{y_i F_T(x_i) < 0\}} \leq \exp\left(-y_i F_T(x_i)\right)$$



Then

$$1 = \sum_{i=1}^{n} w_i^{T+1}$$

$$= \sum_{i=1}^{n} \frac{\exp\left(-y_i F_T(x_i)\right)}{n \cdot (\pi Z_t)}$$

$$\geq \frac{1}{(\pi Z_t)} \cdot \frac{1}{n} \sum_{i=1}^{n} I_{\{f(x_i) \neq y_i\}}$$

and the lemma follows. ∎

Lemma $\quad Z_t = \sqrt{1-4\gamma_t^2}$

Proof

$$Z_t = \sum_{i=1}^{n} w_i^t \exp\left(-\alpha_t y_i f_t(x_i)\right)$$

$$= \sum_{\substack{i=1 \\ f_t(x_i)=y_i}}^{n} w_i^t e^{-\alpha_t} + \sum_{\substack{i=1 \\ f_t(x_i)\neq y_i}}^{n} w_i^t e^{\alpha_t}$$

$$= (1-r_t)e^{-\alpha_t} + r_t e^{\alpha_t}$$

Now recall

$$\alpha_t = \frac{1}{2}\ln\left(\frac{1-r_t}{r_t}\right)$$

Then

$$Z_t = (1-r_t)\sqrt{\frac{r_t}{1-r_t}} + r_t\sqrt{\frac{1-r_t}{r_t}}$$

$$= 2\sqrt{r_t(1-r_t)}$$

Now substitute

$$r_t = \frac{1}{2} - \gamma_t$$

$$\Longrightarrow \quad Z_t = 2\sqrt{\left(\tfrac{1}{2}-\gamma_t\right)\left(\tfrac{1}{2}+\gamma_t\right)}$$

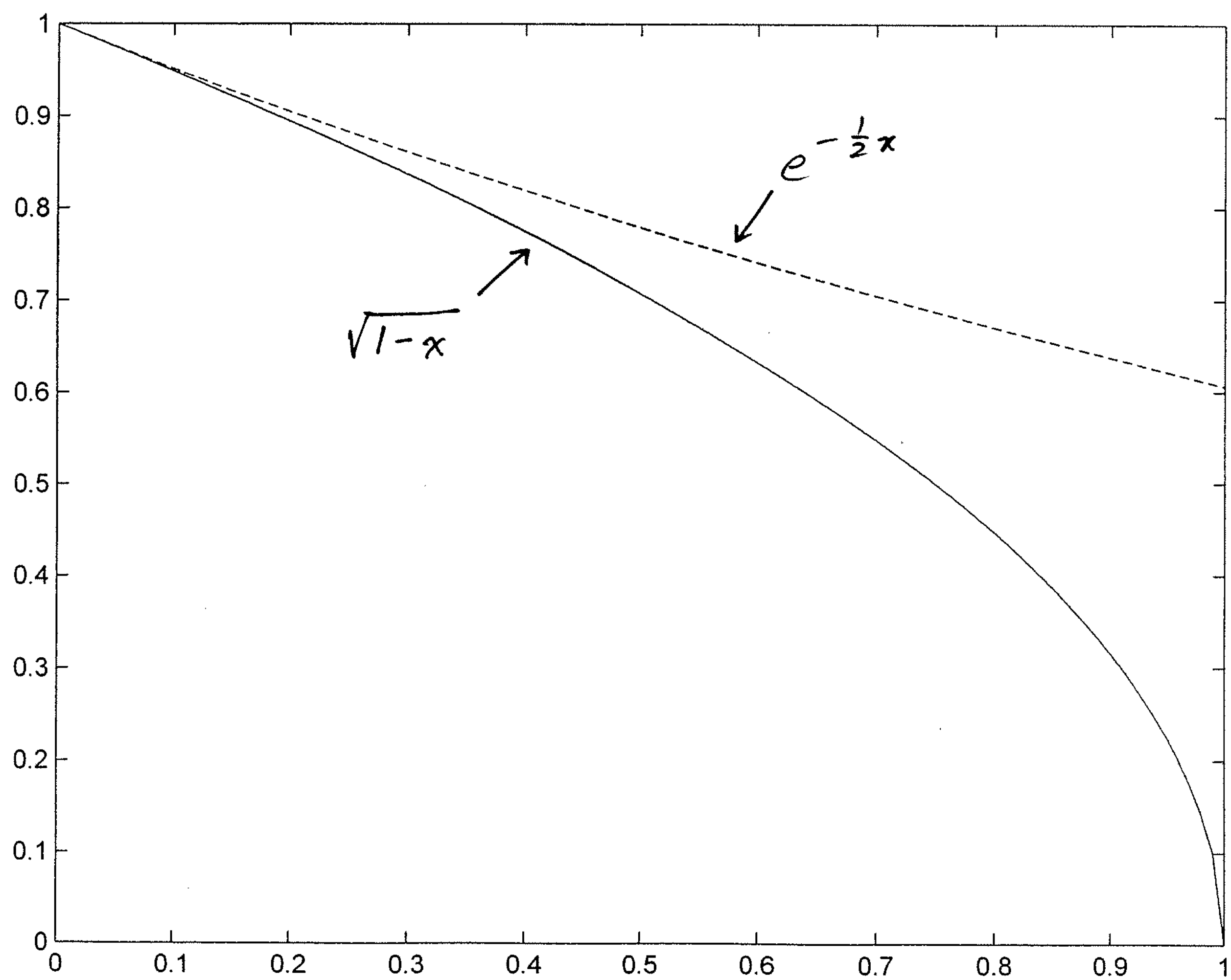
$$= 2\sqrt{\tfrac{1}{4}-\gamma_t^2}$$

$$= \sqrt{1-4\gamma_t^2} \qquad \blacksquare$$

Lemma

$$\sqrt{1-x} \leq e^{-\frac{1}{2}x}$$

Proof



Formally, $\sqrt{1-x}$ is concave, $e^{-\frac{1}{2}x}$ is convex, so it suffices to show their slopes (derivatives) are both $= -\frac{1}{2}$ at 0. $\blacksquare$

Putting it all together, we obtain

$$\frac{1}{n}\sum_{i=1}^{n} I_{\{f(x_i) \neq y_i\}} \leq \frac{1}{n}\sum_{i=1}^{n} \exp(-y_i F_T(x_i))$$
$$= \prod_{t=1}^{T} Z_t$$
$$= \prod_{t=1}^{T} \sqrt{1-4\gamma_t^2}$$
$$\leq e^{-2\sum_{t=1}^{T}\gamma_t^2} \qquad \blacksquare$$

Exercise | View Z_t as a function of α_t, and find the value of α_t that minimizes Z_t.

Solution | Earlier we showed

$$Z_t = (1-r_t)e^{-\alpha_t} + r_t e^{\alpha_t}.$$

This is a convex, differentiable function of α_t.

It is minimized by setting

$$0 = \frac{\partial Z_t}{\partial \alpha_t} = -(1-r_t)e^{-\alpha_t} + r_t e^{\alpha_t}$$

$$\Rightarrow \quad e^{2\alpha_t} = \frac{1-r_t}{r_t}$$

$$\Rightarrow \quad \alpha_t = \frac{1}{2}\ln\left(\frac{1-r_t}{r_t}\right)$$

In conclusion, each α_t is choosen to minimize the corresponding term Z_t in the bound $\prod_{t=1}^{T} Z_t$.

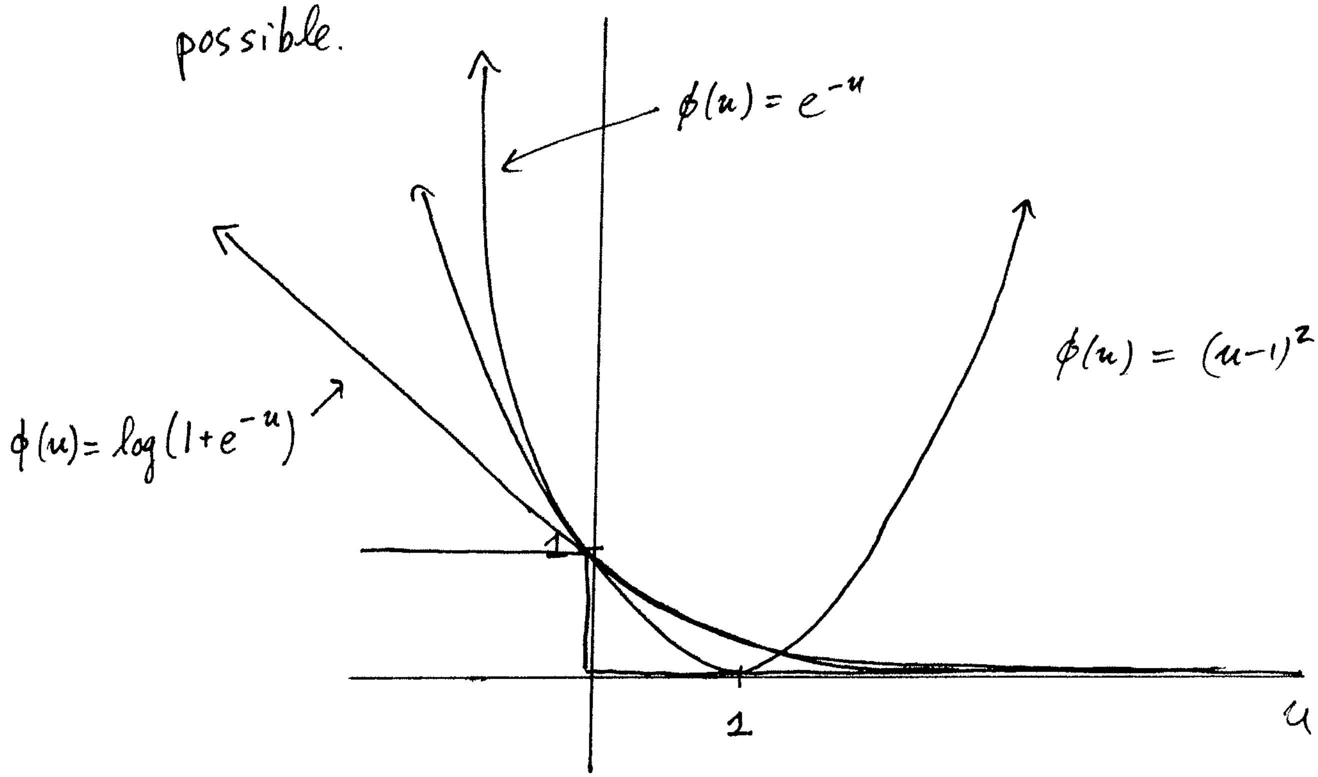
That is, the bound is minimized <u>incrementally</u> (not <u>globally</u>)

Alternative Loss Functions

Adaboost uses the loss function

$$\phi(u) = e^{-u}$$

as a convex, differentiable upper bound on $I_{\{u<0\}}$. However, other loss functions are possible.



For example, the "logistic loss" $\phi(u) = \log(1+e^{-u})$ doesn't work as hard on misclassified points, and therefore may be less susceptible to overfitting.

To generaliz Adaboost to other loss functions, recall

$$F_t(x) = \sum_{s=1}^{t} \alpha_t f_t$$

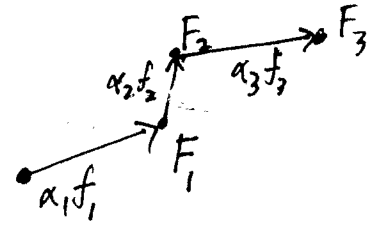
On the t^{th} iteration of boosting, we have the upper bound

$$\frac{1}{n}\sum_{i=1}^{n} I_{\{y_i F_t(x_i) < 0\}} \le \frac{1}{n}\sum_{i=1}^{n} \phi(y_i F_t(x_i)) \qquad (\star)$$

View this bound as an objective to be minimized over function space. That is, view the function F_t as a variable being optimized.

Boosting can be seen as $\boxed{\text{functional gradient descent.}}$

Given that $f_1, \ldots, f_{t-1}$ have been learned, view f_t as the direction of the next step in a gradient descent minimization of the upper bound $(\star)$



We seek f_t that minimizes the slope of B_t at F_{t-1}.

Writing

$$B_t(\alpha_t) = \frac{1}{n}\sum_{i=1}^{n} \phi\left(y_i F_{t-1}(x_i) + y_i \alpha_t f_t(x_i)\right)$$

the slope of B_t in the direction f_t is

$$\left.\frac{\partial B_t}{\partial \alpha_t}\right|_{\alpha_t=0} = \frac{1}{n}\sum_{i=1}^{n} y_i f_t(x_i)\, \phi'(y_i F_{t-1}(x_i))$$

Minimizing this is equivalent to minimizing

$$-\sum_{i=1}^{n} y_i f_t(x_i) \underbrace{\frac{\phi'(y_i F_{t-1}(x_i))}{\sum_{j=1}^{n} \phi'(y_j F_{t-1}(x_j))}}_{=:\, w_i^t}$$

since $\phi' < 0$

$$= \sum_{i: y_i \neq f_t(x_i)} w_i^t - \sum_{i: y_i = f_t(x_i)} w_i^t$$

$$= 2\left(\sum_{i: y_i \neq f_t(x_i)} w_i^t\right) - 1$$

$\Longrightarrow$ We can use the base learner to find f_t

Once the direction f_t is established, the next step is to determine the optimal step-size α_t:

This is achieved by minimizing $B_t(\alpha_t)$ with respect to α_t.

The advantage of the exponential loss is <u>computational</u>

- the weight update has a nice recursive formula since

$$\phi'(a+b) = \phi'(a) \cdot \phi'(b)$$

- $B_t'(\alpha_t) = 0$ has a closed form solution.

However, using other convex losses is not much worse from a computational perspective.

- $\phi'(y_i F_{t-1}(x_i))$ is easy to compute
- α_t is the solution of a univariate, <u>convex</u> optimization problem.

Generalized Boosting Algorithm

Given $(x_1, y_1), \ldots, (x_n, y_n)$, $y_i \in \{-1, 1\}$, convex loss ϕ

Initialize $w_i^1 = \frac{1}{n}$

For $t = 1, \ldots, T$

- Apply base learner with weights w^t to produce classifier f_t

- Set

$$\alpha_t = \arg\min_{\alpha} \frac{1}{n} \sum_{i=1}^{n} \phi\left(y_i F_{t-1}(x_i) + y_i \alpha f_t(x_i)\right)$$

- Update

$$w_i^{t+1} = \frac{\phi'(y_i F_t(x_i))}{\sum_{j=1}^{n} \phi'(y_j F_t(x_j))}$$

End

Output

$$f(x) = \mathrm{sign}\left\{F_T(x)\right\} = \mathrm{sign}\left\{\sum_{t=1}^{T} \alpha_t f_t(x)\right\}$$

Exercise Verify that when $\phi(u) = e^{-u}$, the algorithm reduces to Adaboost.

Solution | If $\phi(u) = e^{-u}$, then $\phi'(u) = -e^{-u}$,

and

$$w_i^t \propto -\phi'(y_i F_t(x_i)) = e^{-y_i F_t(x_i)}$$

$$= \prod_{s=1}^{t} e^{-y_i \alpha_s f_s(x_i)}$$ ✓

To see that α_t is the same as for Adaboost, apply the same argument used to show α_t minimized Z_t.

Remark |

When $\phi(u) = \log_2(1 + e^{-2u})$, and α_t is estimated by a single step of a Newton-Raphson algorithm, the algorithm is called LogitBoost. This is the other common boosting algorithm besides Adaboost.