Ensemble methods

CS 446 / ECE 449

2022-03-01 18:37:08 -0600 (1566808)

"`pytorch` **meta-algorithm**".

⋮

6. Tweak 1-5, possibly reducing model complexity, until testing error is small.

Rather than trying many models and only selecting one,
can we combine them into an **"ensemble"** and do better?

Plan for today

- Bagging.
- Boosting.

Bagging?

Bagging idea:

- If the predictors have **independent errors**,
  a majority vote of their outputs should be good.

Behavior of independent errors

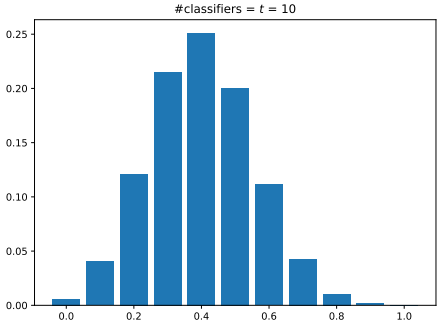
Suppose t classifiers $(f_j)_{j=1}^t$, fixed example $(\boldsymbol{x}, y)$,
independent error probabilities $Z_j := \mathbb{1}[\mathrm{sgn}(f_j(\boldsymbol{x})) \neq y]$,
and $\Pr[Z_j] = 0.4 =: p$ (pretty bad!).

Behavior of independent errors

Suppose t classifiers $(f_j)_{j=1}^t$, fixed example $(\boldsymbol{x}, y)$,
independent error probabilities $Z_j := \mathbb{1}[\mathrm{sgn}(f_j(\boldsymbol{x})) \neq y]$,
and $\Pr[Z_j] = 0.4 =: p$ (pretty bad!).
We can model the distribution of errors with Binom$(t, 0.4)$.

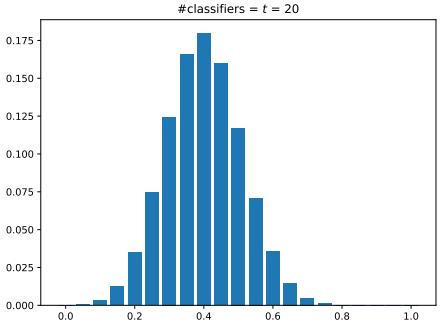
Behavior of independent errors

Suppose t classifiers $(f_j)_{j=1}^t$, fixed example $(\boldsymbol{x}, y)$,
independent error probabilities $Z_j := \mathbb{1}[\mathrm{sgn}(f_j(\boldsymbol{x})) \neq y]$,
and $\Pr[Z_j] = 0.4 =: p$ (pretty bad!).
We can model the distribution of errors with Binom$(t, 0.4)$.



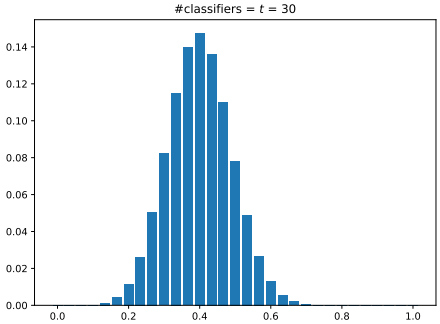
Behavior of independent errors

Suppose t classifiers $(f_j)_{j=1}^t$, fixed example $(\boldsymbol{x}, y)$,
independent error probabilities $Z_j := \mathbb{1}[\mathrm{sgn}(f_j(\boldsymbol{x})) \neq y]$,
and $\Pr[Z_j] = 0.4 =: p$ (pretty bad!).
We can model the distribution of errors with Binom$(t, 0.4)$.



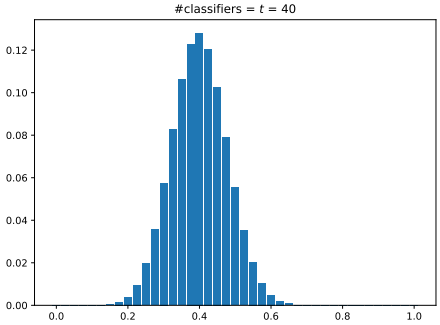
Behavior of independent errors

Suppose t classifiers $(f_j)_{j=1}^t$, fixed example $(\boldsymbol{x}, y)$,
independent error probabilities $Z_j := \mathbb{1}[\mathrm{sgn}(f_j(\boldsymbol{x})) \neq y]$,
and $\Pr[Z_j] = 0.4 =: p$ (pretty bad!).
We can model the distribution of errors with Binom$(t, 0.4)$.



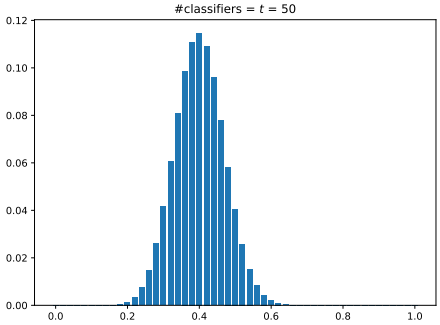
Behavior of independent errors

Suppose t classifiers $(f_j)_{j=1}^t$, fixed example $(\boldsymbol{x}, y)$,
independent error probabilities $Z_j := \mathbb{1}[\mathrm{sgn}(f_j(\boldsymbol{x})) \neq y]$,
and $\Pr[Z_j] = 0.4 =: p$ (pretty bad!).
We can model the distribution of errors with Binom$(t, 0.4)$.



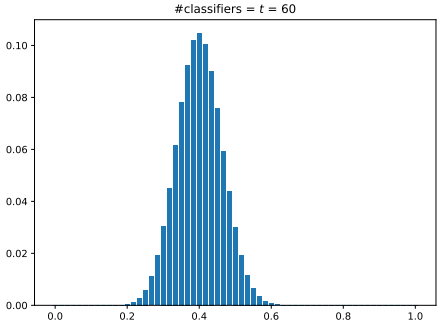
Behavior of independent errors

Suppose t classifiers $(f_j)_{j=1}^t$, fixed example $(\boldsymbol{x}, y)$,
independent error probabilities $Z_j := \mathbb{1}[\mathrm{sgn}(f_j(\boldsymbol{x})) \neq y]$,
and $\Pr[Z_j] = 0.4 =: p$ (pretty bad!).
We can model the distribution of errors with Binom$(t, 0.4)$.



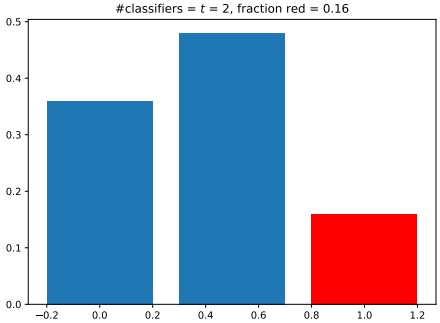
Behavior of independent errors

Suppose t classifiers $(f_j)_{j=1}^t$, fixed example $(\boldsymbol{x}, y)$,
independent error probabilities $Z_j := \mathbb{1}[\mathrm{sgn}(f_j(\boldsymbol{x})) \neq y]$,
and $\Pr[Z_j] = 0.4 =: p$ (pretty bad!).
We can model the distribution of errors with Binom$(t, 0.4)$.



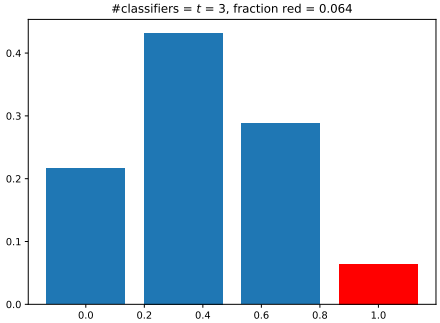
Behavior of independent errors

Suppose t classifiers $(f_j)_{j=1}^t$, fixed example $(\boldsymbol{x}, y)$,
independent error probabilities $Z_j := \mathbb{1}[\mathrm{sgn}(f_j(\boldsymbol{x})) \neq y]$,
and $\Pr[Z_j] = 0.4 =: p$ (pretty bad!).
We can model the distribution of errors with $\mathrm{Binom}(t, 0.4)$.
Red: all classifiers wrong.



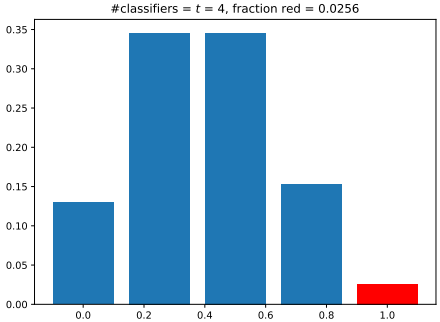
Behavior of independent errors

Suppose t classifiers $(f_j)_{j=1}^t$, fixed example $(\boldsymbol{x}, y)$,
independent error probabilities $Z_j := \mathbb{1}[\mathrm{sgn}(f_j(\boldsymbol{x})) \neq y]$,
and $\Pr[Z_j] = 0.4 =: p$ (pretty bad!).
We can model the distribution of errors with $\mathrm{Binom}(t, 0.4)$.
Red: all classifiers wrong.



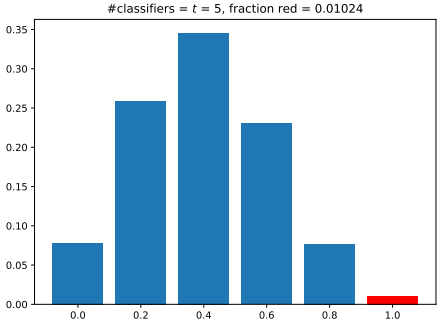
Behavior of independent errors

Suppose t classifiers $(f_j)_{j=1}^t$, fixed example $(\boldsymbol{x}, y)$,
independent error probabilities $Z_j := \mathbb{1}[\mathrm{sgn}(f_j(\boldsymbol{x})) \neq y]$,
and $\Pr[Z_j] = 0.4 =: p$ (pretty bad!).
We can model the distribution of errors with $\mathrm{Binom}(t, 0.4)$.
Red: all classifiers wrong.



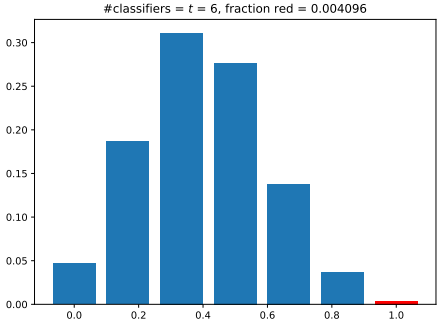
Behavior of independent errors

Suppose t classifiers $(f_j)_{j=1}^t$, fixed example $(\boldsymbol{x}, y)$,
independent error probabilities $Z_j := \mathbb{1}[\mathrm{sgn}(f_j(\boldsymbol{x})) \neq y]$,
and $\Pr[Z_j] = 0.4 =: p$ (pretty bad!).
We can model the distribution of errors with $\mathrm{Binom}(t, 0.4)$.
Red: all classifiers wrong.



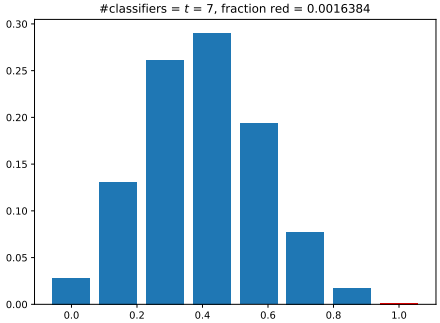
Behavior of independent errors

Suppose t classifiers $(f_j)_{j=1}^t$, fixed example $(\boldsymbol{x}, y)$,
independent error probabilities $Z_j := \mathbb{1}[\mathrm{sgn}(f_j(\boldsymbol{x})) \neq y]$,
and $\Pr[Z_j] = 0.4 =: p$ (pretty bad!).
We can model the distribution of errors with $\mathrm{Binom}(t, 0.4)$.
Red: all classifiers wrong.



Behavior of independent errors

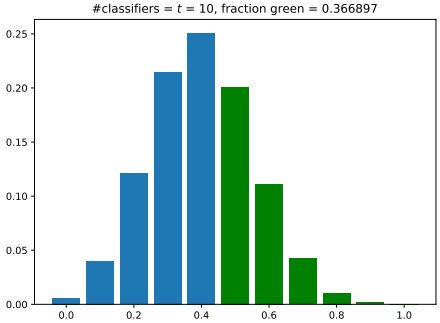
Suppose t classifiers $(f_j)_{j=1}^t$, fixed example $(\boldsymbol{x}, y)$,
independent error probabilities $Z_j := \mathbb{1}[\mathrm{sgn}(f_j(\boldsymbol{x})) \neq y]$,
and $\Pr[Z_j] = 0.4 =: p$ (pretty bad!).
We can model the distribution of errors with $\mathrm{Binom}(t, 0.4)$.
Red: all classifiers wrong.



Behavior of independent errors

Suppose t classifiers $(f_j)_{j=1}^t$, fixed example $(\boldsymbol{x}, y)$,
independent error probabilities $Z_j := \mathbb{1}[\mathrm{sgn}(f_j(\boldsymbol{x})) \neq y]$,
and $\Pr[Z_j] = 0.4 =: p$ (pretty bad!).
We can model the distribution of errors with $\mathrm{Binom}(t, 0.4)$.

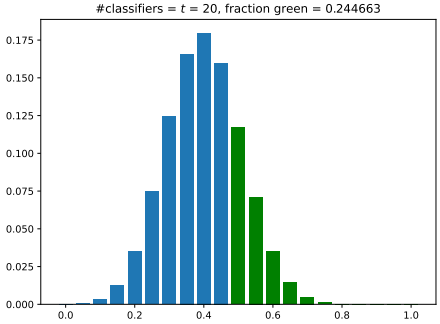
Green: at least half classifiers wrong.



Behavior of independent errors

Suppose t classifiers $(f_j)_{j=1}^t$, fixed example $(\boldsymbol{x}, y)$,
independent error probabilities $Z_j := \mathbb{1}[\mathrm{sgn}(f_j(\boldsymbol{x})) \neq y]$,
and $\Pr[Z_j] = 0.4 =: p$ (pretty bad!).
We can model the distribution of errors with $\mathrm{Binom}(t, 0.4)$.

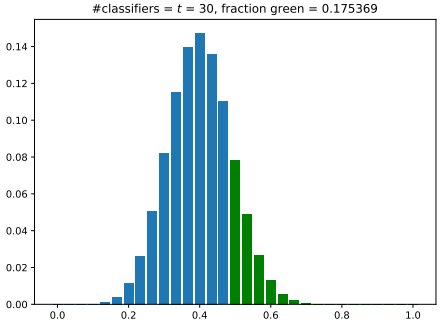
Green: at least half classifiers wrong.



Behavior of independent errors

Suppose t classifiers $(f_j)_{j=1}^t$, fixed example $(\boldsymbol{x}, y)$,
independent error probabilities $Z_j := \mathbb{1}[\mathrm{sgn}(f_j(\boldsymbol{x})) \neq y]$,
and $\Pr[Z_j] = 0.4 =: p$ (pretty bad!).
We can model the distribution of errors with $\mathrm{Binom}(t, 0.4)$.

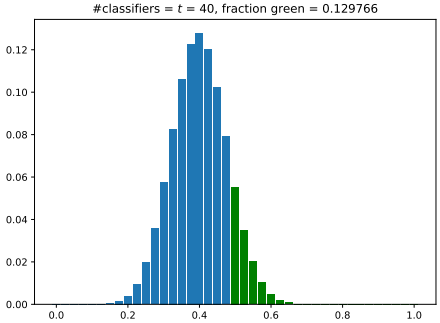
Green: at least half classifiers wrong.



Behavior of independent errors

Suppose t classifiers $(f_j)_{j=1}^t$, fixed example $(\boldsymbol{x}, y)$,
independent error probabilities $Z_j := \mathbb{1}[\mathrm{sgn}(f_j(\boldsymbol{x})) \neq y]$,
and $\Pr[Z_j] = 0.4 =: p$ (pretty bad!).
We can model the distribution of errors with $\mathrm{Binom}(t, 0.4)$.

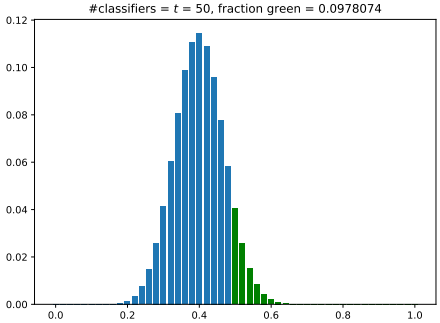
Green: at least half classifiers wrong.



Behavior of independent errors

Suppose t classifiers $(f_j)_{j=1}^t$, fixed example $(\boldsymbol{x}, y)$,
independent error probabilities $Z_j := \mathbb{1}[\mathrm{sgn}(f_j(\boldsymbol{x})) \neq y]$,
and $\Pr[Z_j] = 0.4 =: p$ (pretty bad!).
We can model the distribution of errors with $\mathrm{Binom}(t, 0.4)$.

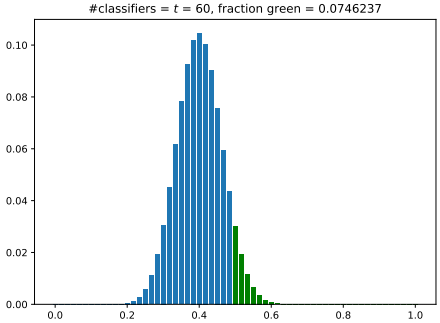
Green: at least half classifiers wrong.



Behavior of independent errors

Suppose t classifiers $(f_j)_{j=1}^t$, fixed example $(\boldsymbol{x}, y)$,
independent error probabilities $Z_j := \mathbb{1}[\mathrm{sgn}(f_j(\boldsymbol{x})) \neq y]$,
and $\Pr[Z_j] = 0.4 =: p$ (pretty bad!).
We can model the distribution of errors with $\mathrm{Binom}(t, 0.4)$.

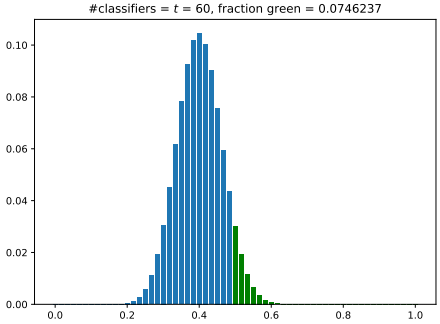
Green: at least half classifiers wrong.



Behavior of independent errors

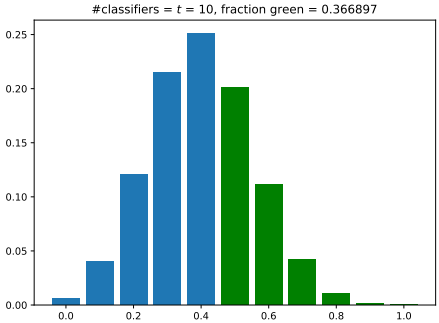
Suppose t classifiers $(f_j)_{j=1}^t$, fixed example $(\boldsymbol{x}, y)$,
independent error probabilities $Z_j := \mathbb{1}[\mathrm{sgn}(f_j(\boldsymbol{x})) \neq y]$,
and $\Pr[Z_j] = 0.4 =: p$ (pretty bad!).
We can model the distribution of errors with $\mathrm{Binom}(t, 0.4)$.

Green: at least half classifiers wrong.



Green region is error of majority vote! $0.075 \ll 0.4$!!!

Majority vote



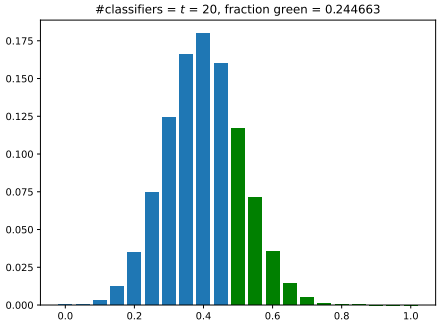
Green region is error of majority vote! Defining $\hat{y}_j := \mathrm{sgn}(f_j(\boldsymbol{x}))$,

$$\mathsf{MAJ}(\hat{y}_1, \ldots, \hat{y}_t) := \begin{cases} +1 & \text{when } \sum_j \hat{y}_j \geq 0, \\ -1 & \text{when } \sum_j \hat{y}_j < 0. \end{cases}$$

Error rate of majority classifier (with individual error probability p):

$$\Pr[\mathsf{MAJ}(\hat{y}_1, \ldots, \hat{y}_t) \neq y] \leq \Pr[\mathsf{Binom}(t, p) \geq t/2] \leq \exp\left(-2t(1/2 - p)^2\right).$$

Majority vote



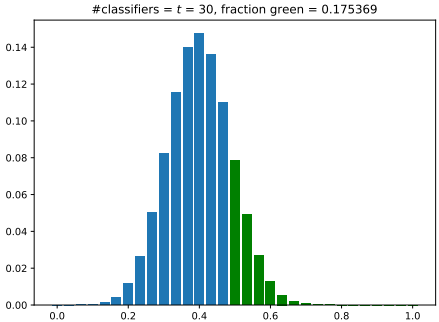
Green region is error of majority vote! Defining $\hat{y}_j := \mathrm{sgn}(f_j(\boldsymbol{x}))$,

$$\mathsf{MAJ}(\hat{y}_1, \ldots, \hat{y}_t) := \begin{cases} +1 & \text{when } \sum_j \hat{y}_j \geq 0, \\ -1 & \text{when } \sum_j \hat{y}_j < 0. \end{cases}$$

Error rate of majority classifier (with individual error probability p):

$$\Pr[\mathsf{MAJ}(\hat{y}_1, \ldots, \hat{y}_t) \neq y] \leq \Pr[\mathsf{Binom}(t, p) \geq t/2] \leq \exp\left(-2t(1/2 - p)^2\right).$$

Majority vote



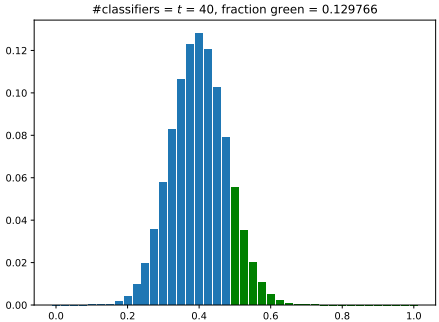
Green region is error of majority vote! Defining $\hat{y}_j := \mathrm{sgn}(f_j(\boldsymbol{x}))$,

$$\mathsf{MAJ}(\hat{y}_1, \ldots, \hat{y}_t) := \begin{cases} +1 & \text{when } \sum_j \hat{y}_j \geq 0, \\ -1 & \text{when } \sum_j \hat{y}_j < 0. \end{cases}$$

Error rate of majority classifier (with individual error probability p):

$$\Pr[\mathsf{MAJ}(\hat{y}_1, \ldots, \hat{y}_t) \neq y] \leq \Pr[\mathsf{Binom}(t, p) \geq t/2] \leq \exp\left(-2t(1/2 - p)^2\right).$$

Majority vote



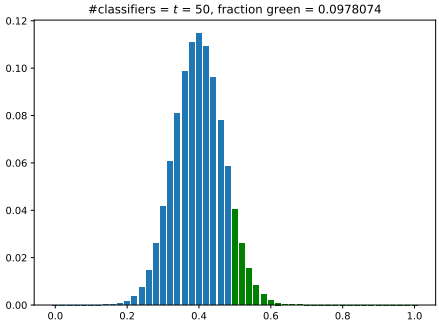
Green region is error of majority vote! Defining $\hat{y}_j := \mathrm{sgn}(f_j(\boldsymbol{x}))$,

$$\mathsf{MAJ}(\hat{y}_1, \ldots, \hat{y}_t) := \begin{cases} +1 & \text{when } \sum_j \hat{y}_j \geq 0, \\ -1 & \text{when } \sum_j \hat{y}_j < 0. \end{cases}$$

Error rate of majority classifier (with individual error probability p):

$$\Pr[\mathsf{MAJ}(\hat{y}_1, \ldots, \hat{y}_t) \neq y] \leq \Pr[\mathsf{Binom}(t, p) \geq t/2] \leq \exp\left(-2t(1/2 - p)^2\right).$$

Majority vote



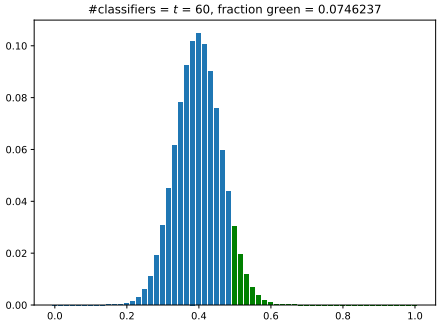
Green region is error of majority vote! Defining $\hat{y}_j := \mathrm{sgn}(f_j(\boldsymbol{x}))$,

$$\mathsf{MAJ}(\hat{y}_1, \ldots, \hat{y}_t) := \begin{cases} +1 & \text{when } \sum_j \hat{y}_j \geq 0, \\ -1 & \text{when } \sum_j \hat{y}_j < 0. \end{cases}$$

Error rate of majority classifier (with individual error probability p):

$$\Pr[\mathsf{MAJ}(\hat{y}_1, \ldots, \hat{y}_t) \neq y] \leq \Pr[\mathsf{Binom}(t, p) \geq t/2] \leq \exp\left(-2t(1/2 - p)^2\right).$$

Majority vote



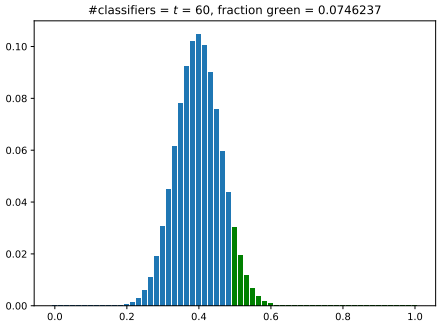
Green region is error of majority vote! Defining $\hat{y}_j := \mathrm{sgn}(f_j(\boldsymbol{x}))$,

$$\mathsf{MAJ}(\hat{y}_1, \ldots, \hat{y}_t) := \begin{cases} +1 & \text{when } \sum_j \hat{y}_j \geq 0, \\ -1 & \text{when } \sum_j \hat{y}_j < 0. \end{cases}$$

Error rate of majority classifier (with individual error probability p):

$$\Pr[\mathsf{MAJ}(\hat{y}_1, \ldots, \hat{y}_t) \neq y] \leq \Pr[\mathsf{Binom}(t, p) \geq t/2] \leq \exp\left(-2t(1/2 - p)^2\right).$$

Bottom line



Green region is error of majority vote!

Error of majority vote classifier goes down **exponentially** in t.

$$\Pr[\mathsf{MAJ}(\hat{y}_1, \ldots, \hat{y}_m) \neq y] \leq \Pr[\mathsf{Binom}(t, p) \geq t/2] \leq \exp\left(-2t(1/2 - p)^2\right).$$

Technical aside: **Hoeffding's inequality**

How did we estimate $\Pr[\text{Binom}(t,p) \geq t/2] \leq \exp\left(-2t(1/2-p)^2\right)$?

Theorem (Hoeffding's inequality). Given IID $Z_i \in [a,b]$ and any $\epsilon > 0$,

$$\Pr\left[\frac{1}{t}\sum_{i=1}^{t} Z_i - \mathbb{E}Z_1 \geq \epsilon\right] \leq \exp\left(\frac{-2t\epsilon^2}{(b-a)^2}\right).$$

Technical aside: **Hoeffding's inequality**

How did we estimate $\Pr[\text{Binom}(t,p) \geq t/2] \leq \exp\left(-2t(1/2-p)^2\right)$?

Theorem (Hoeffding's inequality). Given IID $Z_i \in [a,b]$ and any $\epsilon > 0$,

$$\Pr\left[\frac{1}{t}\sum_{i=1}^{t} Z_i - \mathbb{E}Z_1 \geq \epsilon\right] \leq \exp\left(\frac{-2t\epsilon^2}{(b-a)^2}\right).$$

Our application: set $p := \mathbb{E}Z_1$, we want

$$\begin{aligned}
\Pr\left[\text{MAJ}(\hat{y}_1,\ldots,\hat{y}_m) \neq y\right] &\leq \Pr\left[\sum_{j=1}^{t} Z_j \geq \frac{t}{2}\right] \\
&= \Pr\left[\frac{1}{t}\sum_{j=1}^{t} Z_j - \mathbb{E}Z_1 \geq \frac{1}{2} - p\right] \\
&\leq \exp\left(-2t(1/2-p)^2\right).
\end{aligned}$$

Algorithmically using independent errors

Training algorithm:

1. For $t = 1, 2, \ldots, T$:
   1.1 Obtain IID data $\mathcal{S}_t := ((\boldsymbol{x}_i^{(t)}, y_i^{(t)}))_{i=1}^n$,
   1.2 Train classifier f_t on $\mathcal{S}_t$.
2. Output $\boldsymbol{x} \mapsto \text{MAJ}\left(f_1(\boldsymbol{x}), \ldots, f_T(\boldsymbol{x})\right)$.

Algorithmically using independent errors

Training algorithm:

1. For $t = 1, 2, \ldots, T$:
   1.1 Obtain IID data $\mathcal{S}_t := ((\boldsymbol{x}_i^{(t)}, y_i^{(t)}))_{i=1}^n$,
   1.2 Train classifier f_t on $\mathcal{S}_t$.
2. Output $\boldsymbol{x} \mapsto \mathrm{MAJ}\left(f_1(\boldsymbol{x}), \ldots, f_T(\boldsymbol{x})\right)$.

- **Good news:** errors are independent!
  (Our exponential error estimate from before is valid.)
- **Bad news:** classifiers trained on $1/T$ fraction of data
  (why not just train ResNet on all of it...).

Bagging

Bagging = **B**ootstrap **agg**regat**ing** (Leo Breiman, 1994).

1. Obtain IID data $\mathcal{S} := ((\boldsymbol{x}_i, y_i))_{i=1}^n$.
2. For $t = 1, 2, \ldots, T$:
   2.1 Resample n points uniformly at random with replacement from S, obtaining "Bootstrap sample" S_t.
   2.2 Train classifier f_t on $\mathcal{S}_t$.
3. Output $\boldsymbol{x} \mapsto \text{MAJ}\left(f_1(\boldsymbol{x}), \ldots, f_T(\boldsymbol{x})\right)$.

Bagging

Bagging = **B**ootstrap **agg**regat**ing** (Leo Breiman, 1994).

1. Obtain IID data $\mathcal{S} := ((\boldsymbol{x}_i, y_i))_{i=1}^n$.
2. For $t = 1, 2, \ldots, T$:
   2.1 Resample n points uniformly at random with replacement from S, obtaining "Bootstrap sample" S_t.
   2.2 Train classifier f_t on $\mathcal{S}_t$.
3. Output $\boldsymbol{x} \mapsto \text{MAJ}\left(f_1(\boldsymbol{x}), \ldots, f_T(\boldsymbol{x})\right)$.

- **Good news:** using most of the data for each f_t !
- **Bad news:** errors no longer independent. . . ?

Sampling with replacement?

Question:

Take n samples uniformly at random <u>with replacement</u> from a population of size n. What is the probability that a given individual is <u>not</u> picked?

Sampling with replacement?

Question:

Take n samples uniformly at random <u>with replacement</u> from a population of size n. What is the probability that a given individual is <u>not</u> picked?

Answer: $\left(1 - \frac{1}{n}\right)^n$; for large n: $\lim_{n\to\infty} \left(1 - \frac{1}{n}\right)^n = \frac{1}{e} \approx 0.3679\,.$

Sampling with replacement?

Question:

Take n samples uniformly at random <u>with replacement</u> from a population of size n. What is the probability that a given individual is <u>not</u> picked?

Answer: $\left(1 - \frac{1}{n}\right)^n$; for large n: $\lim_{n\to\infty} \left(1 - \frac{1}{n}\right)^n = \frac{1}{e} \approx 0.3679\,.$

Implications for bagging:

- Each bootstrap sample contains about 63% of the data set.
- Remaining 37% can be used to estimate error rate of classifier trained on the bootstrap sample.
- If we have three classifiers,
  <u>some</u> of their error estimates <u>must</u> share examples!
  Independence is violated!

Random Forests

Random Forests (Leo Breiman, 2001).

1. Obtain IID data $\mathcal{S} := ((\boldsymbol{x}_i, y_i))_{i=1}^n$.
2. For $t = 1, 2, \ldots, T$:
   2.1 Resample n points <u>uniformly at random with replacement</u> from S, obtaining <u>"Bootstrap sample" S_t</u>.
   2.2 Train a decision tree f_T on $\mathcal{S}_t$, considering only $\sqrt{d}$ random features at each split.
3. Output $\boldsymbol{x} \mapsto \text{MAJ}\left(f_1(\boldsymbol{x}), \ldots, f_T(\boldsymbol{x})\right)$.

Random Forests

Random Forests (Leo Breiman, 2001).

1. Obtain IID data $\mathcal{S} := ((\boldsymbol{x}_i, y_i))_{i=1}^n$.
2. For $t = 1, 2, \ldots, T$:
   2.1 Resample n points <u>uniformly at random with replacement</u> from S, obtaining <u>"Bootstrap sample" S_t</u>.
   2.2 Train a decision tree f_T on $\mathcal{S}_t$, considering only $\sqrt{d}$ random features at each split.
3. Output $\boldsymbol{x} \mapsto \text{MAJ}\left(f_1(\boldsymbol{x}), \ldots, f_T(\boldsymbol{x})\right)$.

- Heuristic news: maybe errors are more independent now?

Boosting overview

Boosting overview

- Classifier errors no longer independent.
- Predict with a **reweighted** majority.
- There is a rich theory with many interpretations.

Simplified boosting scheme

1. Start with data $((\boldsymbol{x}_i, y_i)_{i=1}^n$ and classifiers $(h_1, \ldots, h_T)$.
2. Find weights $\boldsymbol{w} \in \mathbb{R}^T$ which approximately minimize empirical risk

$$\frac{1}{n}\sum_{i=1}^n \ell\left(y_i \sum_{j=1}^T w_j h_j(\boldsymbol{x}_i)\right) = \frac{1}{n}\sum_{i=1}^n \ell\left(y_i \boldsymbol{w}^\mathsf{T} \boldsymbol{z}_i\right),$$

   where $\boldsymbol{z}_i = \big(h_1(\boldsymbol{x}_i), \ldots, h_T(\boldsymbol{x}_i)\big) \in \mathbb{R}^T$.
   (We use classifiers to give us features.)
3. Predict with $\boldsymbol{x} \mapsto \sum_{j=1}^T w_j h_j(\boldsymbol{x})$.

Simplified boosting scheme

1. Start with data $((\boldsymbol{x}_i, y_i)_{i=1}^n$ and classifiers $(h_1, \ldots, h_T)$.
2. Find weights $\boldsymbol{w} \in \mathbb{R}^T$ which approximately minimize empirical risk

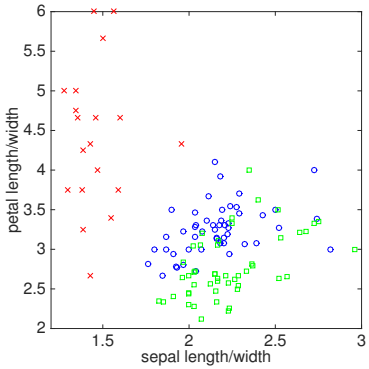
$$\frac{1}{n}\sum_{i=1}^{n} \ell\left(y_i \sum_{j=1}^{T} w_j h_j(\boldsymbol{x}_i)\right) = \frac{1}{n}\sum_{i=1}^{n} \ell\left(y_i \boldsymbol{w}^\intercal \boldsymbol{z}_i\right),$$

   where $\boldsymbol{z}_i = \big(h_1(\boldsymbol{x}_i), \ldots, h_T(\boldsymbol{x}_i)\big) \in \mathbb{R}^T$.
   (We use classifiers to give us features.)
3. Predict with $\boldsymbol{x} \mapsto \sum_{j=1}^{T} w_j h_j(\boldsymbol{x})$.

Remarks.

- If ℓ is convex, this is standard linear prediction: convex in $\boldsymbol{w}$.
- In the classical setting:
  $\ell(r) = \exp(-r)$, optimizer = coordinate descent, $T = \infty$.
- Most commonly, $(h_1, \ldots, h_T)$ are decision stumps.
- Popular software implementations: `xgboost` and `catboost`.

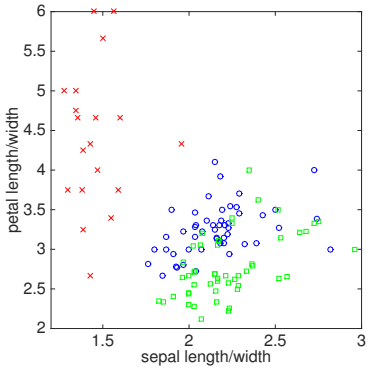
Decision stumps?



Classifying irises by sepal and petal measurements

- $\mathcal{X} = \mathbb{R}^2$, $\mathcal{Y} = \{1, 2, 3\}$
- x_1 = ratio of sepal length to width
- x_2 = ratio of petal length to width

Decision stumps?

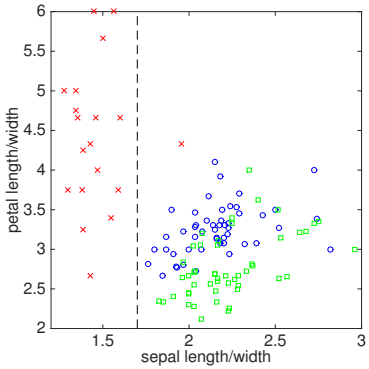


Classifying irises by sepal and petal measurements

- $\mathcal{X} = \mathbb{R}^2$, $\mathcal{Y} = \{1, 2, 3\}$
- x_1 = ratio of sepal length to width
- x_2 = ratio of petal length to width

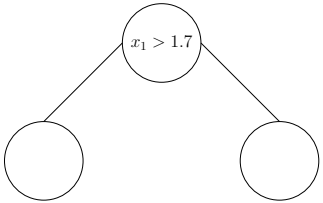
$\hat{y} = 2$

Decision stumps?

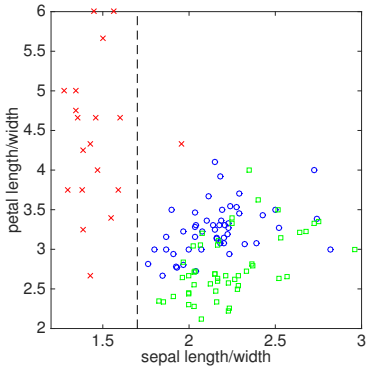


Classifying irises by sepal and petal measurements

- $\mathcal{X} = \mathbb{R}^2$, $\mathcal{Y} = \{1, 2, 3\}$
- x_1 = ratio of sepal length to width
- x_2 = ratio of petal length to width

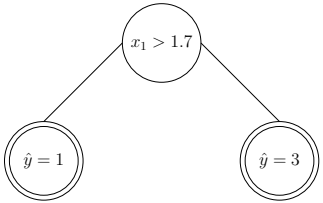


Decision stumps?

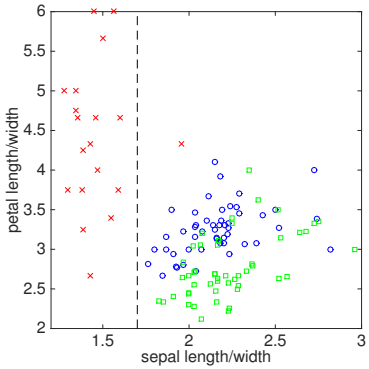


Classifying irises by sepal and petal measurements

- $\mathcal{X} = \mathbb{R}^2$, $\mathcal{Y} = \{1, 2, 3\}$
- x_1 = ratio of sepal length to width
- x_2 = ratio of petal length to width

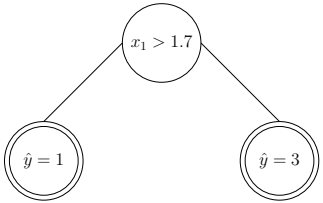


Decision stumps?



Classifying irises by sepal and petal measurements

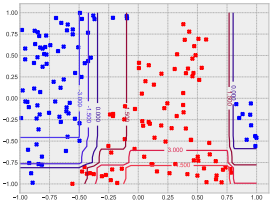
- $\mathcal{X} = \mathbb{R}^2$, $\mathcal{Y} = \{1, 2, 3\}$
- x_1 = ratio of sepal length to width
- x_2 = ratio of petal length to width



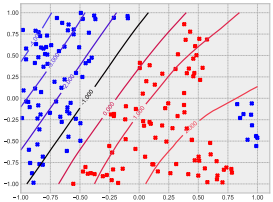
. . . and stop there!

Boosting decision stumps

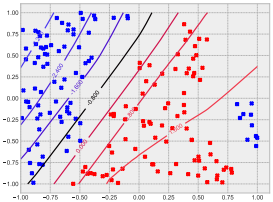
Minimizing $\frac{1}{n}\sum_{i=1}^{n} \ell\left(y_i \sum_{j=1}^{T} w_j h_j(\boldsymbol{x}_i)\right)$ over $\boldsymbol{w} \in \mathbb{R}^T$,
where $(h_1, \ldots, h_T)$ are decision stumps.



Boosted stumps.
($\mathcal{O}(n)$ param.)



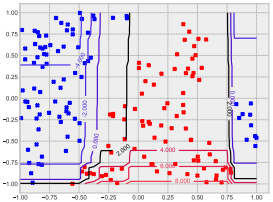
2-layer ReLU.
($\mathcal{O}(n)$ param.)



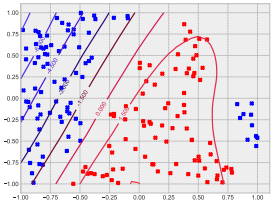
3-layer ReLU.
($\mathcal{O}(n)$ param.)

Boosting decision stumps

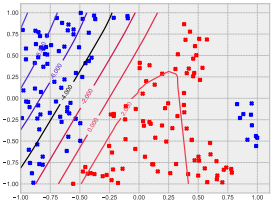
Minimizing $\frac{1}{n}\sum_{i=1}^{n} \ell\left(y_i \sum_{j=1}^{T} w_j h_j(\boldsymbol{x}_i)\right)$ over $\boldsymbol{w} \in \mathbb{R}^T$,
where $(h_1, \ldots, h_T)$ are decision stumps.



Boosted stumps.
($\mathcal{O}(n)$ param.)



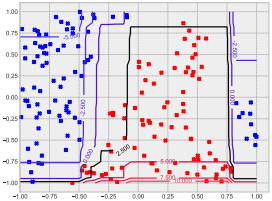
2-layer ReLU.
($\mathcal{O}(n)$ param.)



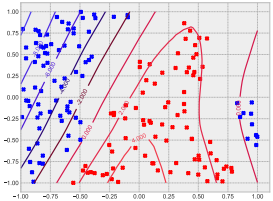
3-layer ReLU.
($\mathcal{O}(n)$ param.)

Boosting decision stumps

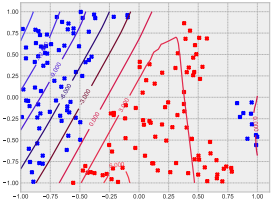
Minimizing $\frac{1}{n}\sum_{i=1}^{n} \ell\left(y_i \sum_{j=1}^{T} w_j h_j(\boldsymbol{x}_i)\right)$ over $\boldsymbol{w} \in \mathbb{R}^T$,
where $(h_1, \ldots, h_T)$ are decision stumps.



Boosted stumps.
($\mathcal{O}(n)$ param.)



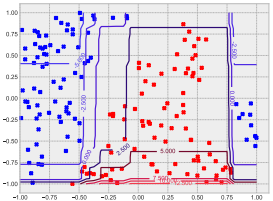
2-layer ReLU.
($\mathcal{O}(n)$ param.)



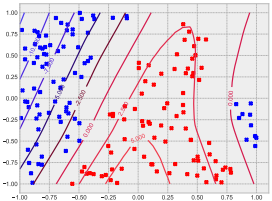
3-layer ReLU.
($\mathcal{O}(n)$ param.)

Boosting decision stumps

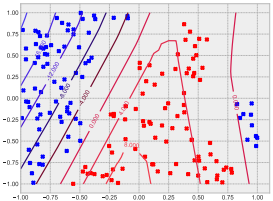
Minimizing $\frac{1}{n}\sum_{i=1}^{n} \ell\left(y_i \sum_{j=1}^{T} w_j h_j(\boldsymbol{x}_i)\right)$ over $\boldsymbol{w} \in \mathbb{R}^T$,
where $(h_1, \ldots, h_T)$ are decision stumps.



Boosted stumps.
($\mathcal{O}(n)$ param.)



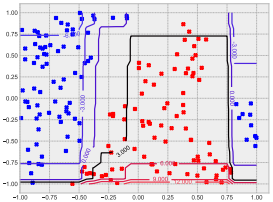
2-layer ReLU.
($\mathcal{O}(n)$ param.)



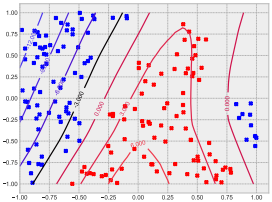
3-layer ReLU.
($\mathcal{O}(n)$ param.)

Boosting decision stumps

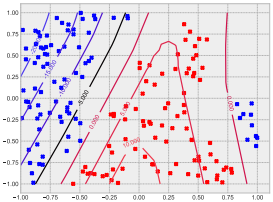
Minimizing $\frac{1}{n}\sum_{i=1}^{n} \ell\left(y_i \sum_{j=1}^{T} w_j h_j(\boldsymbol{x}_i)\right)$ over $\boldsymbol{w} \in \mathbb{R}^T$,
where $(h_1, \ldots, h_T)$ are decision stumps.



Boosted stumps.
($\mathcal{O}(n)$ param.)



2-layer ReLU.
($\mathcal{O}(n)$ param.)



3-layer ReLU.
($\mathcal{O}(n)$ param.)

More realistic boosting algorithm outline

1. Initial predictor $f_0(x) := 0$.
2. For $t \in \{1, 2, \ldots\}$:
   2.1 Define **example weights** $q_i := \exp(-y_i f_t(x_i))$.
   2.2 Selecting **maximally correlated predictor**:

$$(s_t, h_t) := \underset{\substack{s \in \pm 1 \\ h \in \mathcal{H}}}{\arg\max} \sum_{i=1}^{n} q_i y_i s h(x_i).$$

   2.3 Update predictor: $f_t := f_{t-1} - \eta_t s_t h_t$.

More realistic boosting algorithm outline

1. Initial predictor $f_0(x) := 0$.
2. For $t \in \{1, 2, \ldots\}$:
   2.1 Define **example weights** $q_i := \exp(-y_i f_t(x_i))$.
   2.2 Selecting **maximally correlated predictor**:

$$(s_t, h_t) := \underset{\substack{s \in \pm 1 \\ h \in \mathcal{H}}}{\arg\max} \sum_{i=1}^{n} q_i y_i s h(x_i).$$

   2.3 Update predictor: $f_t := f_{t-1} - \eta_t s_t h_t$.

Remarks. (Further details in appendix.)

1. $\mathcal{H}$ typically infinite; the inner maximization is its own ERM problem.
2. This is equivalent to **coordinate descent** with $\ell(y\hat{y}) = \exp(-y\hat{y})$: reweighting is loss derivative term.
3. Step size η_t usually carefully chosen.
4. These remarks follow "AdaBoost" (which has the most theory); `xgboost` has many differences.

Boosting and classifier complexity

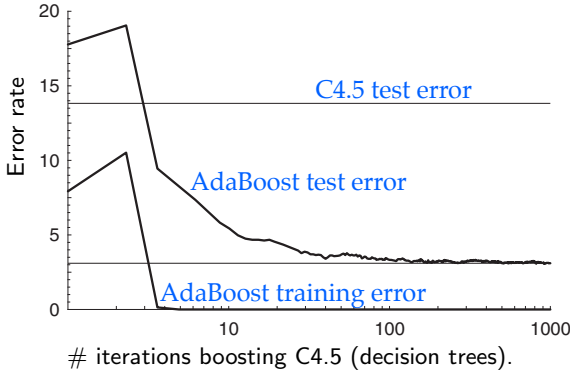


Figure 1.7 from Schapire & Freund textbook.)

Resilient to overfitting.

- # classifiers (and thus "complexity") increase, **but** test error stays low.
- Many consider this similar to deep learning, where test error plateaus or improves with increasing network size.

Margins and generalization?

- Write final classifier as $f_t(\boldsymbol{x}) := \sum_{i=1}^m w_i h_i(\boldsymbol{x})$.
- Define ℓ_1 **margins** $\gamma_i := \frac{y_i f_t(\boldsymbol{x}_i)}{\|\boldsymbol{w}\|_1}$.
  (Recall margin with SVM: $y_i \boldsymbol{x}_i^\mathsf{T} \boldsymbol{w} / \|\boldsymbol{w}\|_2$.)
- Margins and test error seem to improve on "letters" dataset:

	$T = 5$	$T = 100$	$T = 1000$
training error rate	0.0%	0.0%	0.0%
test error rate	8.4%	3.3%	3.1%
fraction $\gamma_i \leq 0.5$	7.7%	0.0%	0.0%
$\min_i \gamma_i$	0.14	0.52	0.55

- There is a lot of theory about margins, generalization properties of margins, and margins in deep learning and boosting.

Summary for today

- Bagging.
- Boosting.

(Appendix.)

The original presentation of dropout called it an ensemble method (you train random subnetworks to do well, and at test time predict with their combination). There's no evidence or math behind this assertion, but still it's an interesting perspective.

Classical perspective on boosting: coordinate descent?

The classical methods used coordinate descent:

- Find the maximum magnitude coordinate of the gradient:

$$\begin{aligned}
&\arg\max_j \left| \frac{\mathrm{d}}{\mathrm{d}w_j} \sum_{i=1}^n \ell\left(\sum_j w_j h_j(\boldsymbol{x}_i) y_i\right) \right| \\
&= \arg\max_j \left| \sum_{i=1}^n \ell'\left(\sum_j w_j h_j(\boldsymbol{x}_i) y_i\right) h_j(\boldsymbol{x}_i) y_i \right| \\
&= \arg\max_j \left| \sum_{i=1}^n q_i h_j(\boldsymbol{x}_i) y_i \right|,
\end{aligned}$$

  where we've defined $q_i := \ell'(\sum_j h_j(\boldsymbol{x}_i) y_i)$.

- Iterate: $\boldsymbol{w}' := \boldsymbol{w} - \eta s \mathbf{e}_j$, where j is the maximum coordinate, $s \in \{-1, +1\}$ is its sign, and η is a step size.

Interpreting coordinate descent

Suppose $h_j : \mathbb{R}^d \to \{-1, +1\}$; then $h_j(\boldsymbol{x})y = 2 \cdot \mathbb{1}[h_j(\boldsymbol{x}) = y] - 1$, and each step solves

$$\arg\max_j \left| \sum_{i=1}^n q_i h_j(\boldsymbol{x}_i) y_i \right| = \arg\max_j \left| \sum_{i=1}^n q_i \left(\mathbb{1}[h_j(\boldsymbol{x}_i) = y] - 1/2 \right) \right|.$$

We are solving a weighted zero-one loss minimization problem.

Interpreting coordinate descent

Suppose $h_j : \mathbb{R}^d \to \{-1, +1\}$; then $h_j(\boldsymbol{x})y = 2 \cdot \mathbb{1}[h_j(\boldsymbol{x}) = y] - 1$, and each step solves

$$\arg\max_j \left| \sum_{i=1}^n q_i h_j(\boldsymbol{x}_i) y_i \right| = \arg\max_j \left| \sum_{i=1}^n q_i \left(\mathbb{1}[h_j(\boldsymbol{x}_i) = y] - 1/2 \right) \right| .$$

We are solving a weighted zero-one loss minimization problem.

Remarks:

- The classical choice of coordinate descent is equivalent to solving a problem akin to weighted zero-one loss minimization.
- We can abstract away a finite set $(h_1, \ldots, h_T)$, and have an arbitrary set of predictors (e.g., all linear classifiers).

Classical boosting setup

There is a Weak Learning Oracle, and a corresponding γ-weak-learnable assumption:

> *A set of points is γ-weak-learnable a weak learning oracle if for any weighting $\boldsymbol{q}$, it returns predictor h so that $\mathbb{E}_{\boldsymbol{q}}\left(h(\boldsymbol{X})Y\right) \geq \gamma$.*

Interpretation: for any reweighting $\boldsymbol{q}$, we get a predictor h which is at least γ-correlated with the target.

Classical boosting setup

There is a Weak Learning Oracle, and a corresponding γ-weak-learnable assumption:

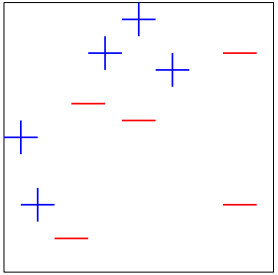
> *A set of points is γ-weak-learnable a weak learning oracle if for any weighting $\boldsymbol{q}$, it returns predictor h so that $\mathbb{E}_{\boldsymbol{q}}\left(h(\boldsymbol{X})Y\right) \geq \gamma$.*

Interpretation: for any reweighting $\boldsymbol{q}$, we get a predictor h which is at least γ-correlated with the target.

Remarks:

- The classical methods iteratively invoke the oracle with different weightings and then output a final aggregated predictor.
- The best-known method, AdaBoost, performs coordinate-descent updates (invoking the oracle) with a specific step size, and needs $\mathcal{O}(\frac{1}{\gamma^2}\ln(\frac{1}{\epsilon}))$ iterations for accuracy $\epsilon > 0$.
- The original description of AdaBoost is in terms of the sequence of weightings $\boldsymbol{q}_1, \boldsymbol{q}_2, \ldots$, and says nothing about coordinate descent.
- **Ada**ptive **Boost**ing: method doesn't need to know γ, and adapts to varying $\gamma_t := \mathbb{E}_{\boldsymbol{q}_t}(h_t(\boldsymbol{X})Y)$.

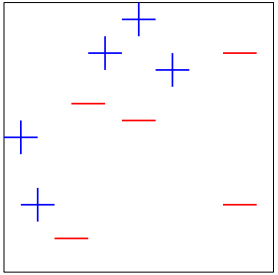
Example: AdaBoost with decision stumps



(This example from Schapire&Freund's book.)

Weak learning oracle (WLO): pick the best decision stump, meaning
$\mathcal{F} := \left\{ \boldsymbol{x} \mapsto \operatorname{sign}(x_i - b) : i \in \{1, \ldots, d\}, b \in \mathbb{R} \right\}$.
(Straightforward to handle weights in ERM.)

Example: AdaBoost with decision stumps



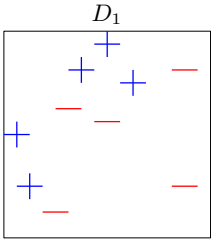
(This example from Schapire&Freund's book.)

Weak learning oracle (WLO): pick the best decision stump, meaning
$\mathcal{F} := \{\boldsymbol{x} \mapsto \operatorname{sign}(x_i - b) : i \in \{1, \ldots, d\}, b \in \mathbb{R}\}$.
(Straightforward to handle weights in ERM.)

Remark:

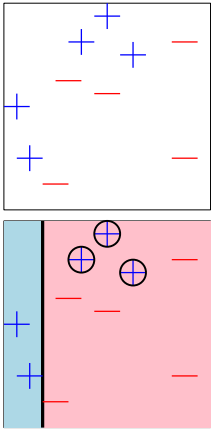
- Only need to consider $\mathcal{O}(n)$ stumps (Why?).

Example: execution of AdaBoost



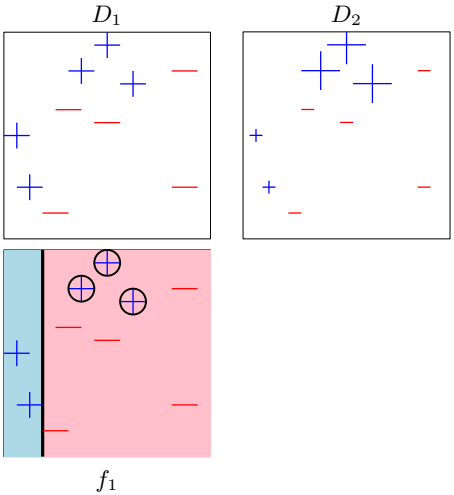
Example: execution of AdaBoost

D_1

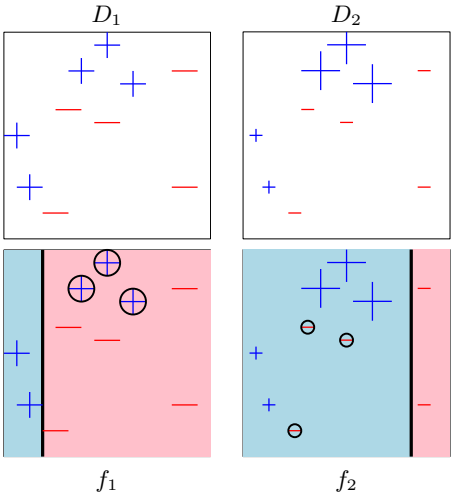


f_1

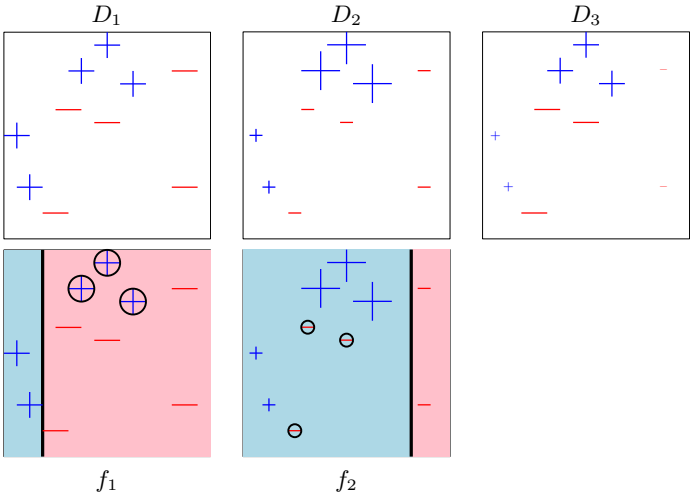
Example: execution of AdaBoost



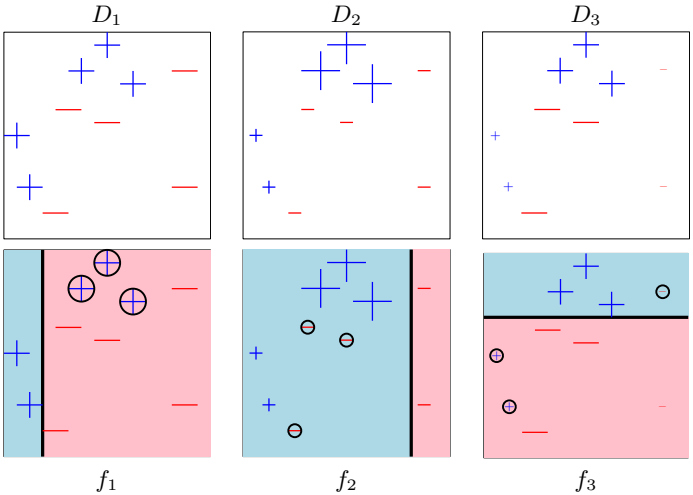
Example: execution of AdaBoost



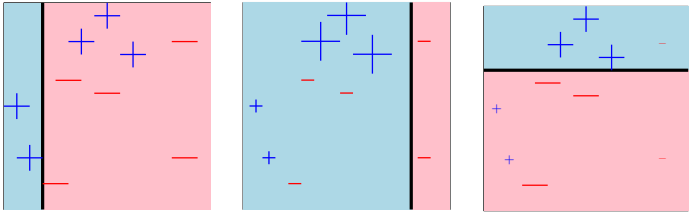
Example: execution of AdaBoost



Example: execution of AdaBoost

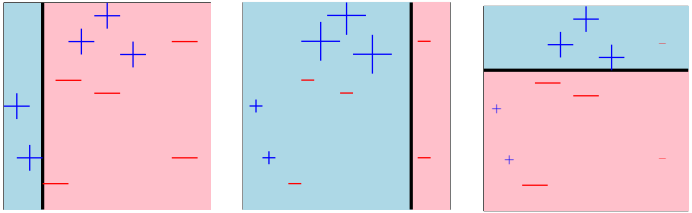


Example: final classifier from AdaBoost



f_1 f_2 f_3

Example: final classifier from AdaBoost

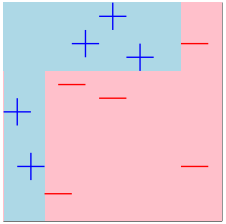


f_1 f_2 f_3

Final classifier

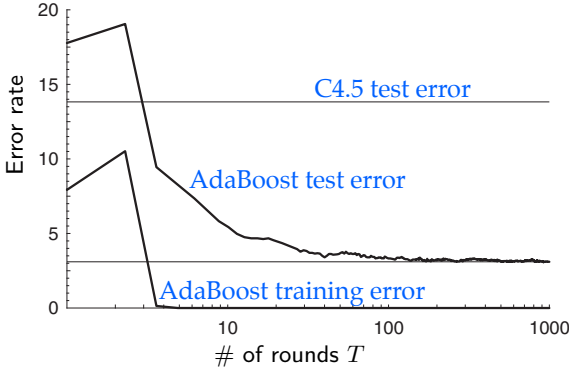
$\hat{f}(\boldsymbol{x}) = \mathrm{sign}(0.42 f_1(\boldsymbol{x}) + 0.65 f_2(\boldsymbol{x}) + 0.92 f_3(\boldsymbol{x}))$

(Zero training error rate!)



A typical run of boosting.

AdaBoost+C4.5 on "letters" dataset.



(# nodes across all decision trees in $\hat{f}$ is $>2 \times 10^6$)

Training error rate is zero after just five rounds,
but test error rate continues to decrease, even up to 1000 rounds!

(Figure 1.7 from Schapire & Freund text)

Boosting the margin.

Final classifier from AdaBoost:

$$\hat{f}(\boldsymbol{x}) \;=\; \operatorname{sign}\underbrace{\left(\frac{\sum_{t=1}^{T}\alpha_t f_t(\boldsymbol{x})}{\sum_{t=1}^{T}|\alpha_t|}\right)}_{g(\boldsymbol{x})\in[-1,+1]}.$$

Call $y \cdot g(\boldsymbol{x}) \in [-1, +1]$ the **margin** achieved on example $(\boldsymbol{x}, y)$.
(Note: ℓ_1 not ℓ_2 normalized.)

Boosting the margin.

Final classifier from AdaBoost:

$$\hat{f}(\boldsymbol{x}) \;=\; \operatorname{sign}\underbrace{\left(\frac{\sum_{t=1}^{T}\alpha_t f_t(\boldsymbol{x})}{\sum_{t=1}^{T}|\alpha_t|}\right)}_{g(\boldsymbol{x})\in[-1,+1]}.$$

Call $y \cdot g(\boldsymbol{x}) \in [-1,+1]$ the **margin** achieved on example $(\boldsymbol{x}, y)$.
(Note: ℓ_1 not ℓ_2 normalized.)

Margin theory [Schapire, Freund, Bartlett, and Lee, 1998]:

- Larger margins $\Rightarrow$ **better generalization**, independent of T.
- AdaBoost tends to increase margins on training examples.

"letters" dataset:

	$T=5$	$T=100$	$T=1000$
training error rate	0.0%	0.0%	0.0%
test error rate	8.4%	3.3%	3.1%
% margins ≤ 0.5	7.7%	0.0%	0.0%
min. margin	0.14	0.52	0.55

- Similar phenomenon in deep networks and gradient descent.

Margin plots

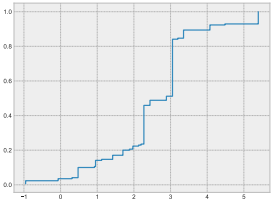
Given $((\boldsymbol{x}_i, y_i))_{i=1}^n$ and f, plot unnormalized margin distribution

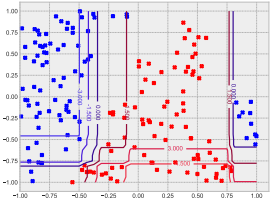
$$f(\boldsymbol{x}_i)_{y_i} - \max_{y \neq y_i} f(\boldsymbol{x}_i)_y.$$

Margin plots

Given $((\boldsymbol{x}_i, y_i))_{i=1}^n$ and f, plot unnormalized margin distribution

$$f(\boldsymbol{x}_i)_{y_i} - \max_{y \neq y_i} f(\boldsymbol{x}_i)_y.$$



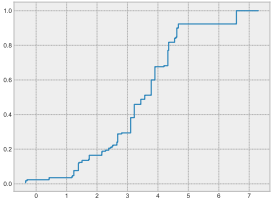


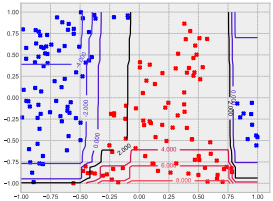
Boosted stumps.
($\mathcal{O}(n)$ param.)

Margin plots

Given $((\boldsymbol{x}_i, y_i))_{i=1}^n$ and f, plot unnormalized margin distribution

$$f(\boldsymbol{x}_i)_{y_i} - \max_{y \neq y_i} f(\boldsymbol{x}_i)_y.$$



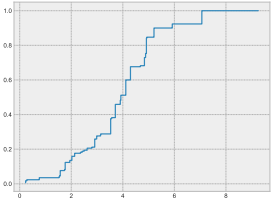


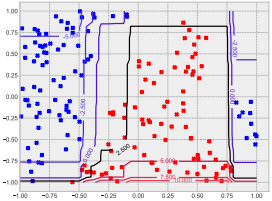
Boosted stumps.
($\mathcal{O}(n)$ param.)

Margin plots

Given $((\boldsymbol{x}_i, y_i))_{i=1}^n$ and f, plot unnormalized margin distribution

$$f(\boldsymbol{x}_i)_{y_i} - \max_{y \neq y_i} f(\boldsymbol{x}_i)_y.$$



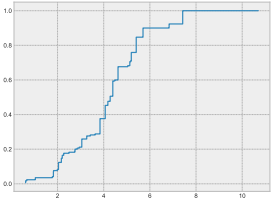


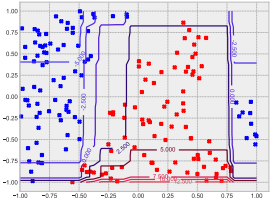
Boosted stumps.
($\mathcal{O}(n)$ param.)

Margin plots

Given $((\boldsymbol{x}_i, y_i))_{i=1}^n$ and f, plot unnormalized margin distribution

$$f(\boldsymbol{x}_i)_{y_i} - \max_{y \neq y_i} f(\boldsymbol{x}_i)_y.$$



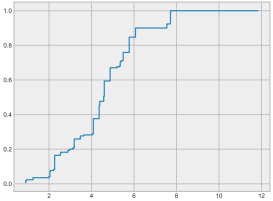


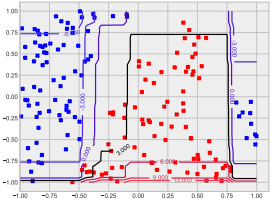
Boosted stumps.
($\mathcal{O}(n)$ param.)

Margin plots

Given $((\boldsymbol{x}_i, y_i))_{i=1}^n$ and f, plot unnormalized margin distribution

$$f(\boldsymbol{x}_i)_{y_i} - \max_{y \neq y_i} f(\boldsymbol{x}_i)_y.$$



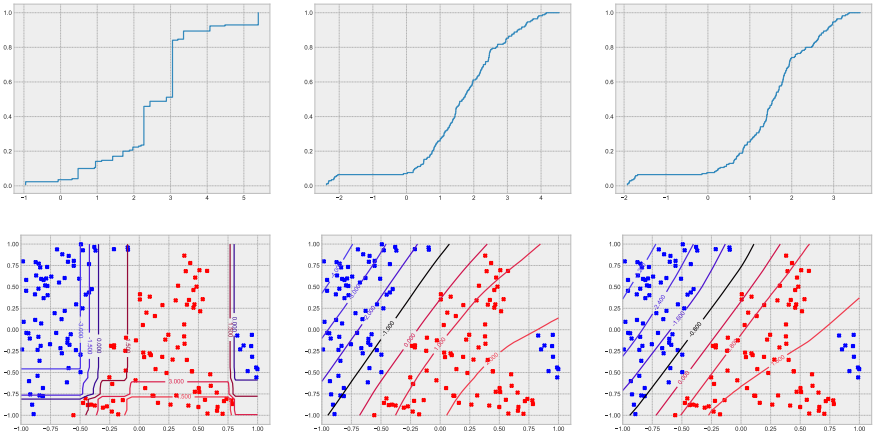


Boosted stumps.
($\mathcal{O}(n)$ param.)

Margin plots

Given $((\boldsymbol{x}_i, y_i))_{i=1}^n$ and f, plot unnormalized margin distribution

$$f(\boldsymbol{x}_i)_{y_i} - \max_{y \neq y_i} f(\boldsymbol{x}_i)_y.$$



Boosted stumps. ($\mathcal{O}(n)$ param.)

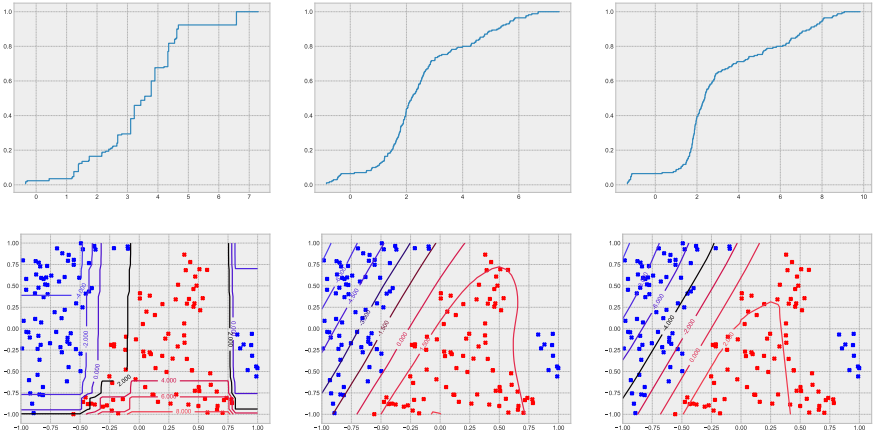
2-layer ReLU. ($\mathcal{O}(n)$ param.)

3-layer ReLU. ($\mathcal{O}(n)$ param.)

Margin plots

Given $((\boldsymbol{x}_i, y_i))_{i=1}^n$ and f, plot unnormalized margin distribution

$$f(\boldsymbol{x}_i)_{y_i} - \max_{y \neq y_i} f(\boldsymbol{x}_i)_y.$$



Boosted stumps. ($\mathcal{O}(n)$ param.)

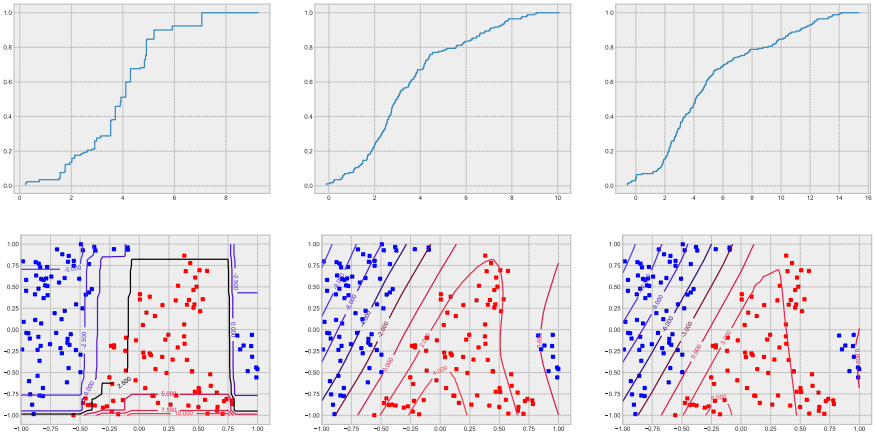
2-layer ReLU. ($\mathcal{O}(n)$ param.)

3-layer ReLU. ($\mathcal{O}(n)$ param.)

Margin plots

Given $((\boldsymbol{x}_i, y_i))_{i=1}^n$ and f, plot unnormalized margin distribution

$$f(\boldsymbol{x}_i)_{y_i} - \max_{y \neq y_i} f(\boldsymbol{x}_i)_y.$$



Boosted stumps. ($\mathcal{O}(n)$ param.)

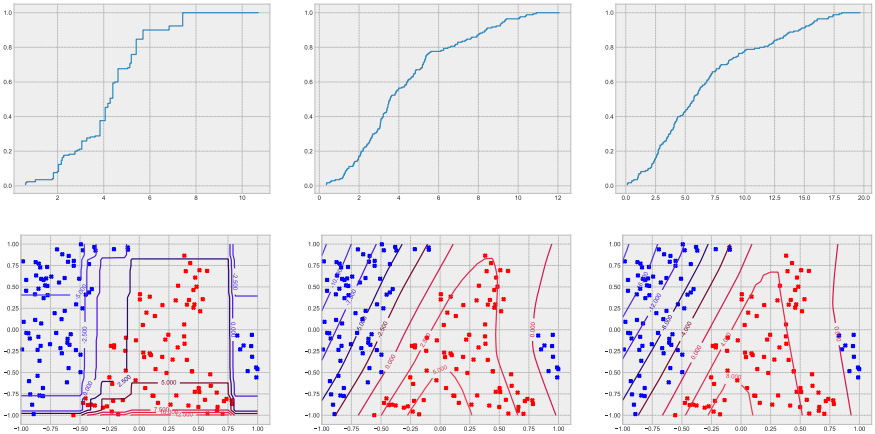
2-layer ReLU. ($\mathcal{O}(n)$ param.)

3-layer ReLU. ($\mathcal{O}(n)$ param.)

Margin plots

Given $((\boldsymbol{x}_i, y_i))_{i=1}^n$ and f, plot unnormalized margin distribution

$$f(\boldsymbol{x}_i)_{y_i} - \max_{y \neq y_i} f(\boldsymbol{x}_i)_y.$$



Boosted stumps. ($\mathcal{O}(n)$ param.)

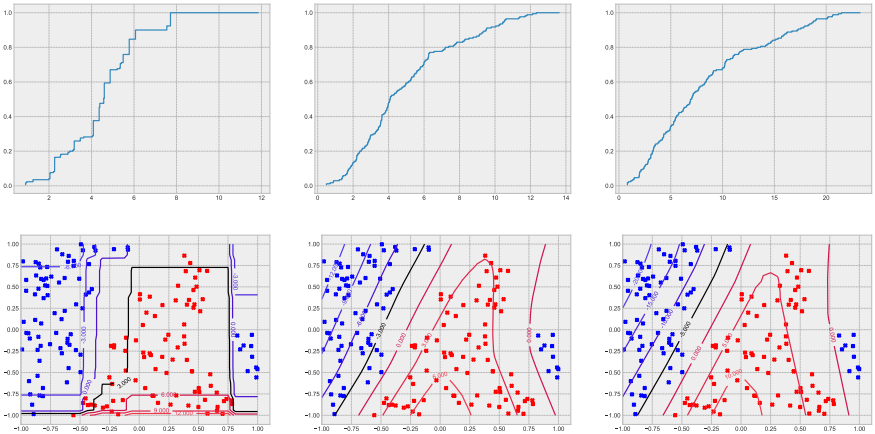
2-layer ReLU. ($\mathcal{O}(n)$ param.)

3-layer ReLU. ($\mathcal{O}(n)$ param.)

Margin plots

Given $((\boldsymbol{x}_i, y_i))_{i=1}^n$ and f, plot unnormalized margin distribution

$$f(\boldsymbol{x}_i)_{y_i} - \max_{y \neq y_i} f(\boldsymbol{x}_i)_y.$$



Boosted stumps. ($\mathcal{O}(n)$ param.)

2-layer ReLU. ($\mathcal{O}(n)$ param.)

3-layer ReLU. ($\mathcal{O}(n)$ param.)