# Adaboost

Aaditya Ramdas

Dept. of Statistics and Data Science
Machine Learning Dept.
Carnegie Mellon University

# Outline

1. Weak learning implies strong learning (1/2 class)

2. ***Adaboost (1/2 class)***

# Recap

A "mixed" strategy is a distribution over actions.

Expected payoff is $\mathbb{E}_{r\sim p, c\sim q}[M(r,c)] = p^T M q = \sum_{r\in[R]} \sum_{c\in C} M(r,c) p_r q_c$

Theorem: $\min_{q\in\Delta_C} \max_{p\in\Delta_R} p^T M q = \max_{p\in\Delta_R} \min_{q\in\Delta_C} p^T M q = v^*$

Value of the game

$e_j$ is the canonical basis vector [0,…,1,…0]

Implications: $\exists p \in \Delta_R \; \forall q \in \Delta_C \;\; p^T M q \geq v^*$

$\forall q \in \Delta_C \exists i \in [R] \;\; e_i^T M q \geq v^*$

$\exists q \in \Delta_C \forall p \in \Delta_R \;\; p^T M q \leq v^*$

$\forall p \in \Delta_R \exists j \in [C] \;\; p^T M e_j \leq v^*$

**For "mixed strategies", order does not matter!**

## Zero-sum games [ edit ]

The minimax theorem was first proven and published in 1928 by John von Neumann,[3] who is quoted as saying "*As far as I can see, there could be no theory of games … without that theorem … I thought there was nothing worth publishing until the Minimax Theorem was proved*".[4]

Formally, von Neumann's minimax theorem states:

Let $X \subset \mathbb{R}^n$ and $Y \subset \mathbb{R}^m$ be compact convex sets. If $f : X \times Y \to \mathbb{R}$ is a continuous function that is concave-convex, i.e.

$f(\cdot, y) : X \to \mathbb{R}$ is concave for fixed $y$, and
$f(x, \cdot) : Y \to \mathbb{R}$ is convex for fixed $x$.

Then we have that

$$\max_{x \in X} \min_{y \in Y} f(x, y) = \min_{y \in Y} \max_{x \in X} f(x, y).$$

# Sion's minimax theorem

From Wikipedia, the free encyclopedia

In mathematics, and in particular game theory, **Sion's minimax theorem** is a generalization of John von Neumann's minimax theorem, named after Maurice Sion.

It states:

Let $X$ be a compact convex subset of a linear topological space and $Y$ a convex subset of a linear topological space. If $f$ is a real-valued function on $X \times Y$ with

$f(x, \cdot)$ upper semicontinuous and quasi-concave on $Y, \forall x \in X$, and
$f(\cdot, y)$ lower semicontinuous and quasi-convex on $X, \forall y \in Y$

then,

$$\min_{x \in X} \sup_{y \in Y} f(x, y) = \sup_{y \in Y} \min_{x \in X} f(x, y).$$

# Notation

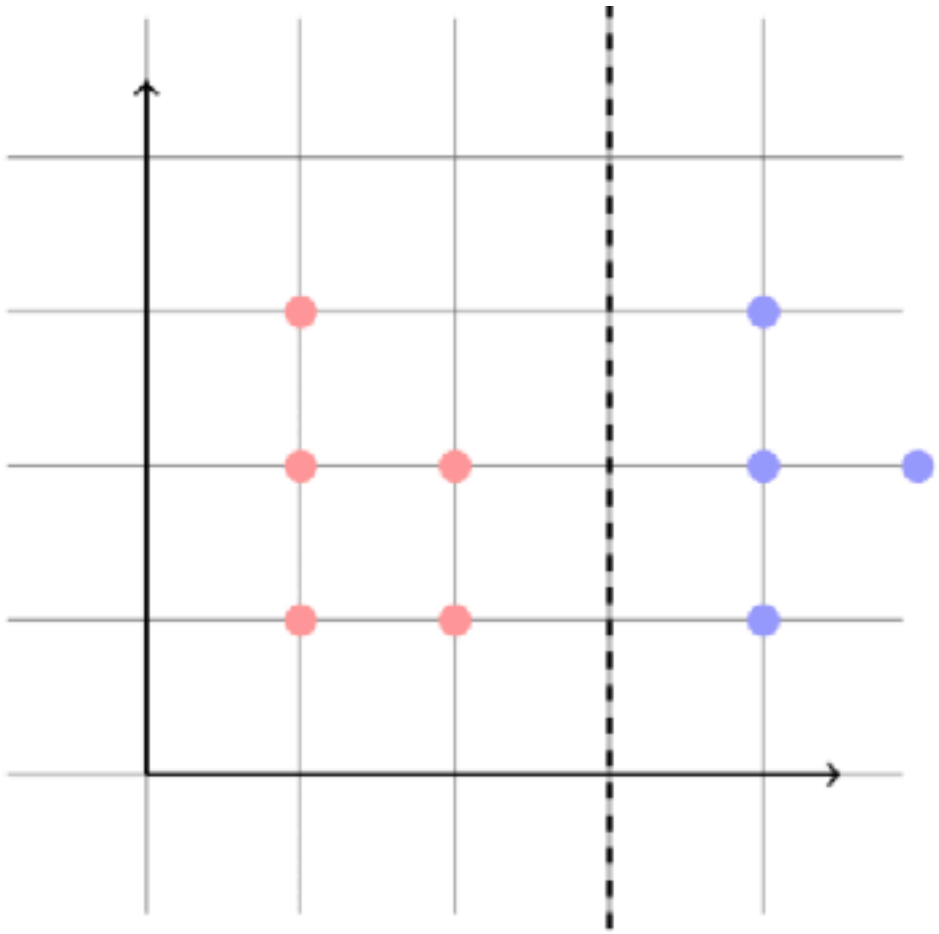
- **Last lectures** (binary) classifier outputs 0/1. In this case the Bayes classifier has form $\mathbb{I}\{\mathbb{E}[Y|X] > 1/2\}$.
- **This lecture** (binary) classifier outputs -1/1. In this case the Bayes classifier has form $\mathbb{I}\{\mathbb{E}[Y|X] > 0\} = \text{sign}(\mathbb{E}[Y|X])$.

Usually classifiers have form $h(x) = \text{sign}(H(x))$. Examples include classification based on logistic regression, $k$-nearest-neighbors, boosting.

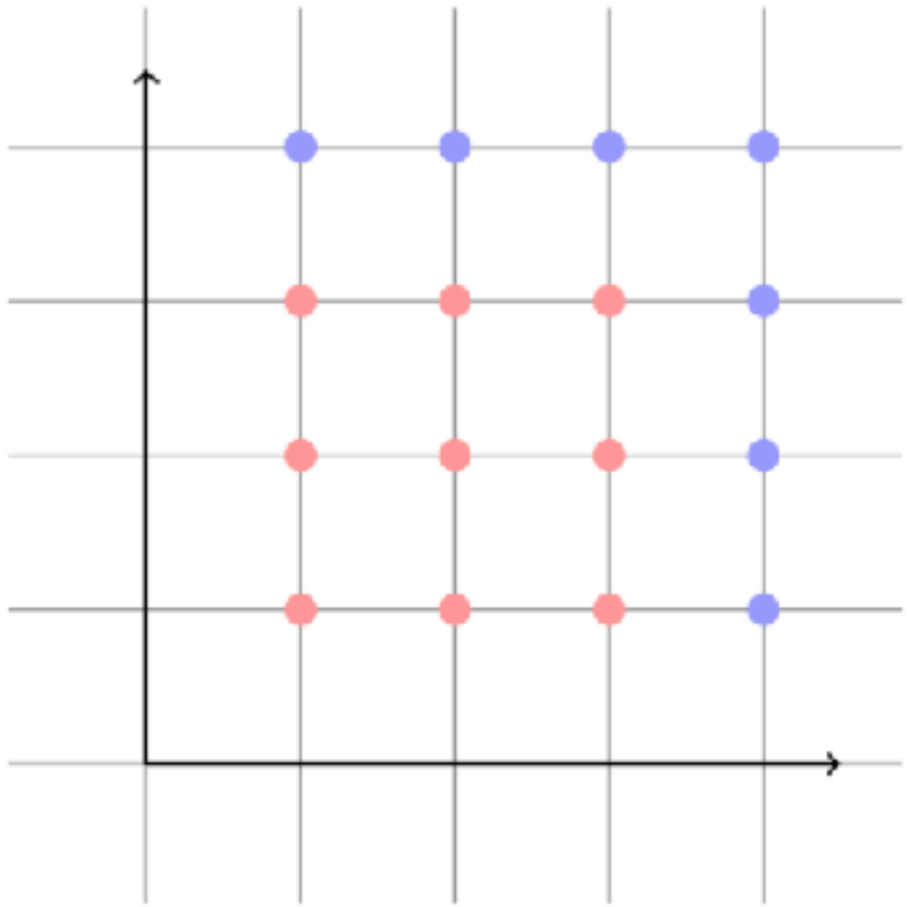
Decision stump: $h(X_i) = 2\mathbb{1}(e_j^T X_i \geq c) - 1$, for some j, c

Decision list: a sequence of if/else decision stumps

Decision tree: a tree of if/else decision stumps



(a) Decision stump performs well.



(b) Decision stump fails. However, decision lists does well

**Algorithm 1** Decision list example

**if** $e_1^\top x_i > 3.5$ **then**
  Predict $+1$
**else if** $e_2^\top x_i > 3.5$ **then**
  Predict $+1$
**else**
  Predict $-1$
**end if**

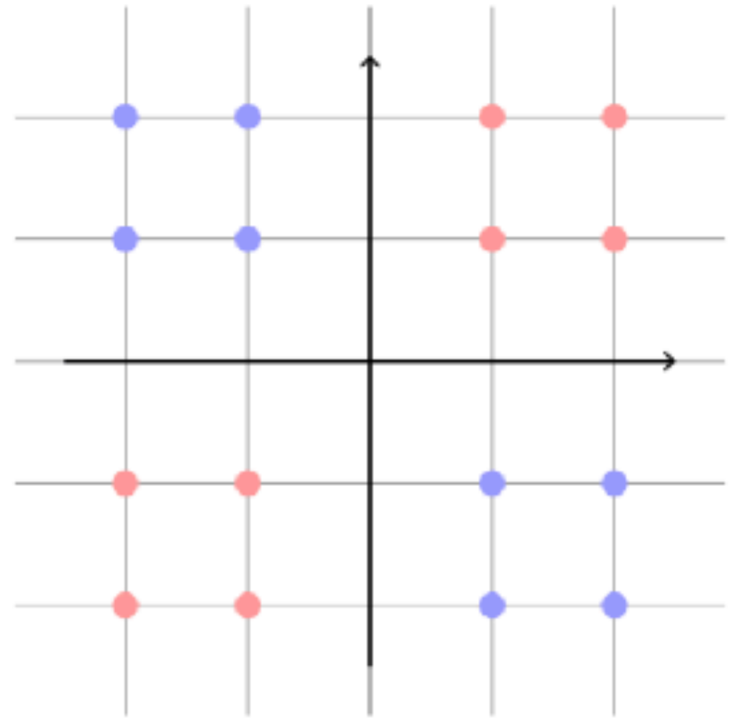


Figure 5.2: Decision list perform poorly. However, decision tree performs well

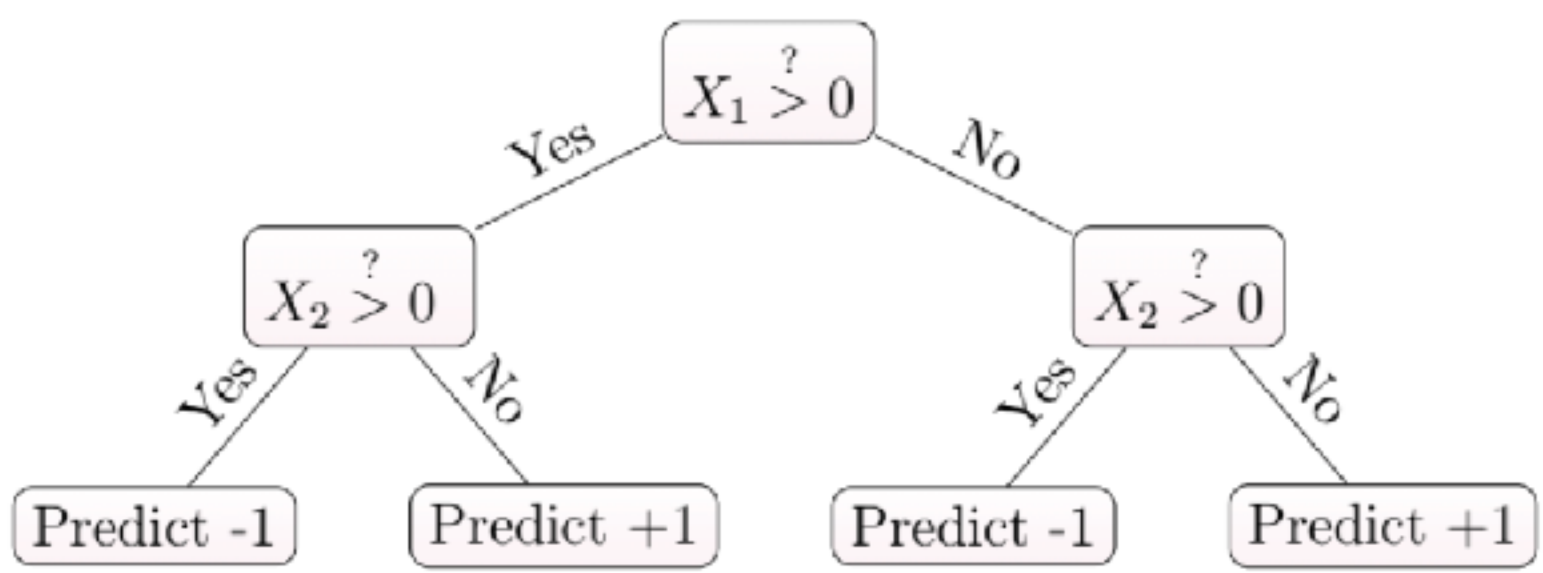



Figure 5.3: Example of a decision tree

# Edge or “margin”, weighted edge

**“Edge” of a classifier** Edge of classifier $h \in \mathcal{H}$ is defined as $\frac{1}{n}\sum_{i=1}^{n} Y_i h(X_i)$. Edge provides a way to describe how much better than chance the classifier is:

- Assume that there is a perfect classifier $h^\star : h^\star(X_i) = Y_i, \forall i \in \{1, \ldots, n\}$. Then its edge is simply equal to 1.
- Consider, on the contrary, a random guess classifier. It is trivial to show that with high probability its edge concentrates around 0.

Error of a classifier in this case can be viewed as: $\frac{1}{2} \cdot (1 - \text{edge})$. For further analysis we define a “weighted” edge as $\sum_{i=1}^{n} w_i Y_i h(X_i)$ where weights satisfy:

$$\sum_{i=1}^{n} w_i = 1, \qquad w_i > 0,\ \forall i \in \{1, \ldots, n\}$$

In the previous definition each data point is equally weighted with weight $1/n$.

# Weak learning hypothesis

Weak learning hypothesis: $\exists \gamma > 0$, such that for any set of weights $w$, there is a classifier $h \in \mathscr{H}$ with weighted edge at least $\gamma$.

Let $M(r,i) = h_r(X_i)Y_i$ $\quad \forall w \in \Delta_n \ \exists h \in [H] \ \ e_h^T M w \geq \gamma \qquad (|\mathscr{H}| = H)$

Strong learning: $\exists$a classifier in span$(\mathscr{H})$ with zero training error

$$\exists p \in \Delta_H \ \forall w \in \Delta_n \ p^T M w \geq \gamma$$

$$\exists p \in \Delta_H \qquad \underbrace{\forall i \in [n] \ p^T M e_i \geq \gamma}_{\text{every element of } p^T M \text{ is positive}}$$

$f(X_i) := \text{sign}(p^T M e_i)$ has zero training error, for some $p \in \Delta_H$

## The breakthrough

Weak learning implies strong learning!

But how do we find this mixture $p \in \Delta_H$ of classifiers?

# Outline

1. Weak learning implies strong learning (1/2 class)

2. ***Adaboost (1/2 class)***

**Algorithm 1** AdaBoost algorithm

**for** $m = 1, \dots M$ **do**

(1) Compute weighted error:

$$\varepsilon(h) = \sum_{i=1}^{n} w_i \mathbb{I}\{Y_i \neq h(X_i)\}$$

Find a classifier $h_m$:

$$h_m = \arg\min_{h \in \mathcal{H}} \varepsilon(h)$$

or pick any $h$ with nontrivial edge

(2) Compute:

$$\alpha_m = \frac{1}{2} \log\left(\frac{1 - \varepsilon_m}{\varepsilon_m}\right)$$

$$\epsilon_m = \epsilon(h_m)$$

(3) Update weights as:

$$w_i \leftarrow \frac{w_i e^{-\alpha_m Y_i h_m(X_i)}}{Z_m}$$

where $Z$ is a normalization

**end for**

Output the classifier:

$$\boxed{\begin{aligned} Z_m &= \sum_{i=1}^{n} w_m(i) \exp(-\alpha_m y_i h(x_i)) \\ &= \sum_{i: y_i h_m(x)=1} w_m(i) \exp(-\alpha_m) + \sum_{i: y_i h_m(x)=-1} w_m(i) \exp(\alpha_m) \\ &= (1 - \varepsilon_m) \exp(-\alpha_m) + \varepsilon_m \exp(\alpha_m) \\ &= 2\sqrt{\varepsilon_m (1 - \varepsilon_m)} \end{aligned}}$$

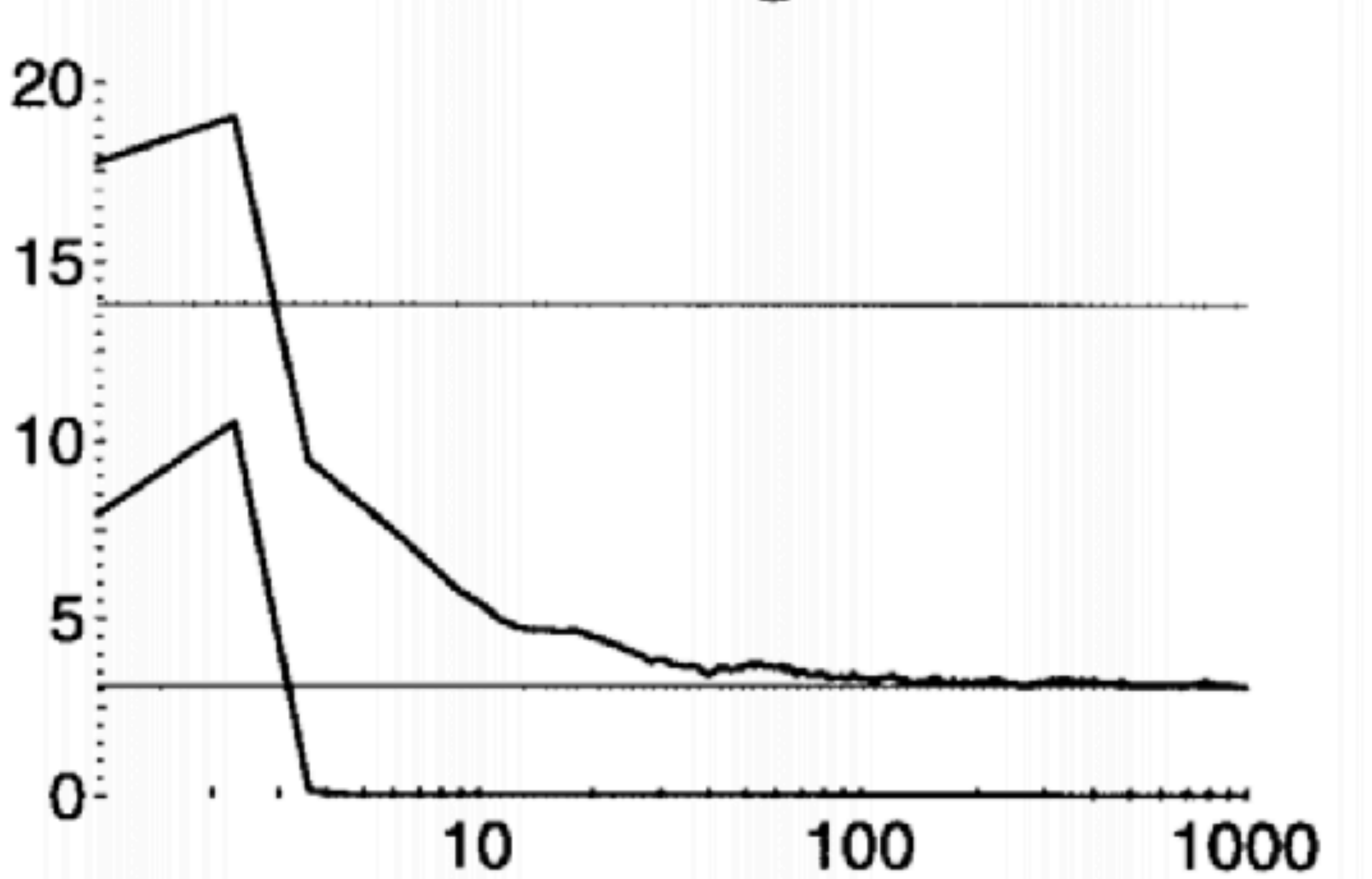



Figure 5.4: This fugure is taken from [2]: each learning curve shows the training and test error curves (lower and upper curves, respectively) of the combined classifier as a function of the number of classifiers combined.

**Theorem 2 (Convergence of empirical risk of Adaboost)** *The empirical risk of the output of Adaboost algorithm 1 $\hat{R}(f)$ satisfies:*

$$\hat{R}(f) \leq \exp(-2\sum_{m=1}^{M}(\frac{1}{2} - \varepsilon_m)^2) \tag{6.3}$$

$\leq \exp(-2M\gamma^2)$ if weak learning hypothesis is true

If $M > \frac{\log n}{2\gamma^2}$, then $\hat{R}(f) < 1/n$, and hence $\hat{R}(f) = 0$

**Theorem 2 (Convergence of empirical risk of Adaboost)** *The empirical risk of the output of Adaboost algorithm 1 $\hat{R}(f)$ satisfies:*

$$\hat{R}(f) \leq \exp(-2\sum_{m=1}^{M}(\frac{1}{2}-\varepsilon_m)^2) \tag{6.3}$$

$$\begin{aligned}\hat{R}(f) =& \frac{1}{n}\sum_{i=1}^{n}\mathbb{I}(y_i \neq f(x_i)) \\ \leq& \frac{1}{n}\sum_{i=1}^{n}\exp(-y_i f(x_i)) \\ =& \frac{1}{n}\sum_{i=1}^{n}[n\prod_{m=1}^{M} Z_m] w_{M+1}(i) \\ =& \prod_{m=1}^{M} Z_m\end{aligned}$$

Surrogate loss

$$w_{M+1}(i) = \frac{w_M(i)e^{-\alpha_m Y_i h_m(X_i)}}{Z_M} = \frac{e^{-Y_i\sum_m \alpha_m h_m(X_i)}}{n\prod_{m=1}^{M} Z_m}$$

$\alpha_m$ minimizes weighted loss

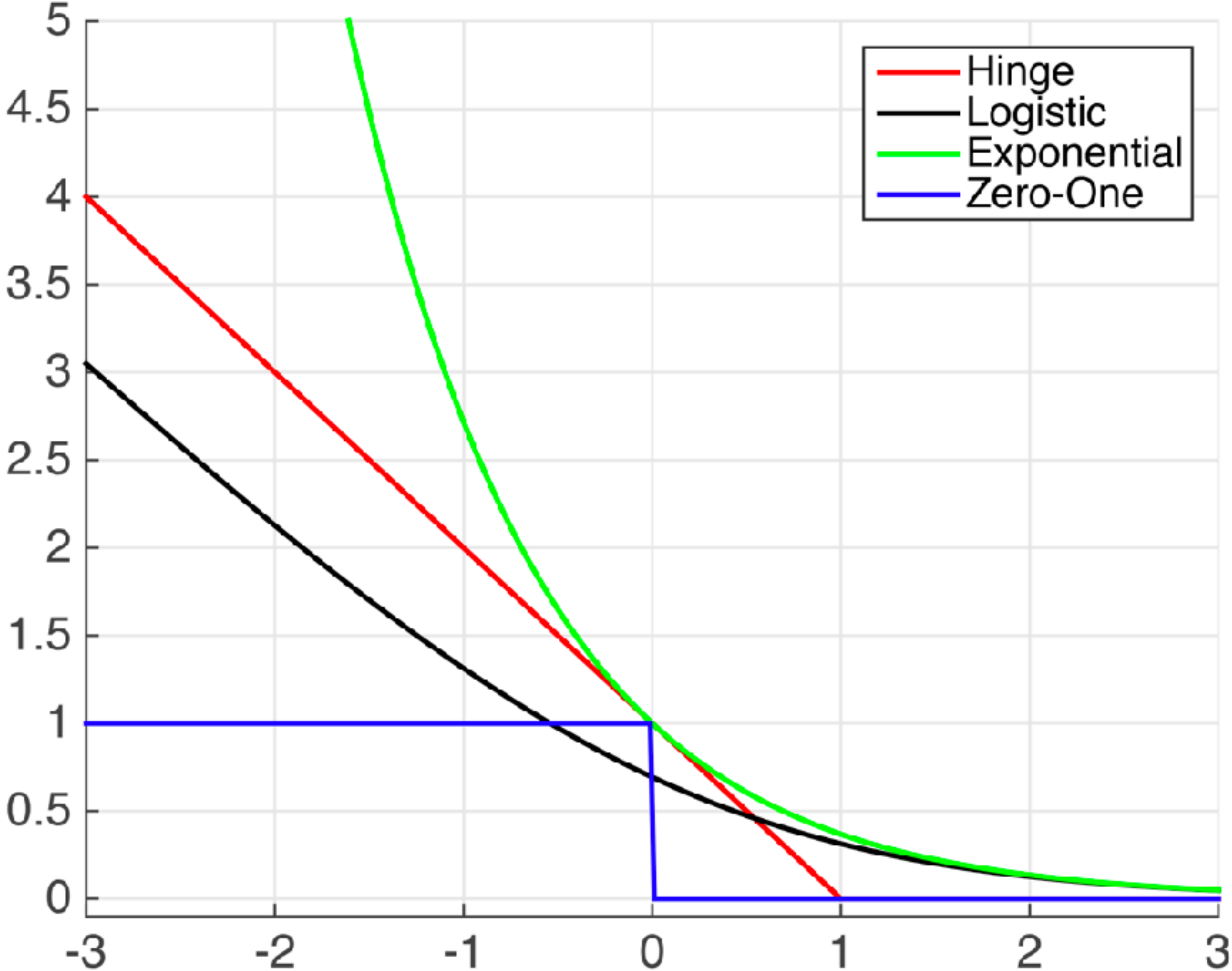
$$\begin{aligned}Z_m =& \sum_{i=1}^{n} w_m(i)\exp(-\alpha_m y_i h(x_i)) \\ =& \sum_{i:y_ih_m(x)=1} w_m(i)\exp(-\alpha_m) + \sum_{i:y_ih_m(x)=-1} w_m(i)\exp(\alpha_m) \\ =& (1-\varepsilon_m)\exp(-\alpha_m) + \varepsilon_m\exp(\alpha_m) \\ =& 2\sqrt{\varepsilon_m(1-\varepsilon_m)} \quad \boxed{\leq \exp(-2(\frac{1}{2}-\varepsilon_m)^2)}\end{aligned}$$

# Convex surrogate loss minimization by coordinate descent

$$\hat{R}(\beta) = \frac{1}{n}\sum_{i=1}^{n} \exp\left(-y_i \sum_{j=1}^{H} \beta_j h_j(x_i)\right)$$

Adaboost solves $\min_{\beta \in \mathbb{R}_+^H} \hat{R}(\beta) = \min_{f \in \mathrm{span}(\mathscr{H})} \hat{R}(f)$ by "coordinate descent".

1. Begin at $\beta^{(0)} = [0,0,\ldots,0]$
2. At step $t$, pick direction $e_t \in \{e_j\}_{j \in J}$ and stepsize $\alpha_t \geq 0$ to minimize $\hat{R}(\beta^{(t-1)} + \alpha_t e_t)$
3. Gradient with respect to coordinate $j$ is
   $\hat{R}'(\beta^{t-1})_j \propto (2\epsilon_{t,j} - 1)\prod_{s=1}^{t-1} Z_s$, where $\epsilon_{t,j}$ is weighted error of $h_j$
4. $h_t$ is chosen to minimize the weighted error,
   optimal stepsize happens to equal $\log(\frac{1-\epsilon_t}{\epsilon_t})$


Hinge
Logistic
Exponential
Zero-One
5
4.5
4
3.5
3
2.5
2
1.5
1
0.5
0
-3
-2
-1
0
1
2
3