Boosting

Jason G. Fleischer, Ph.D.
Asst. Teaching Professor
Department of Cognitive Science, UC San Diego

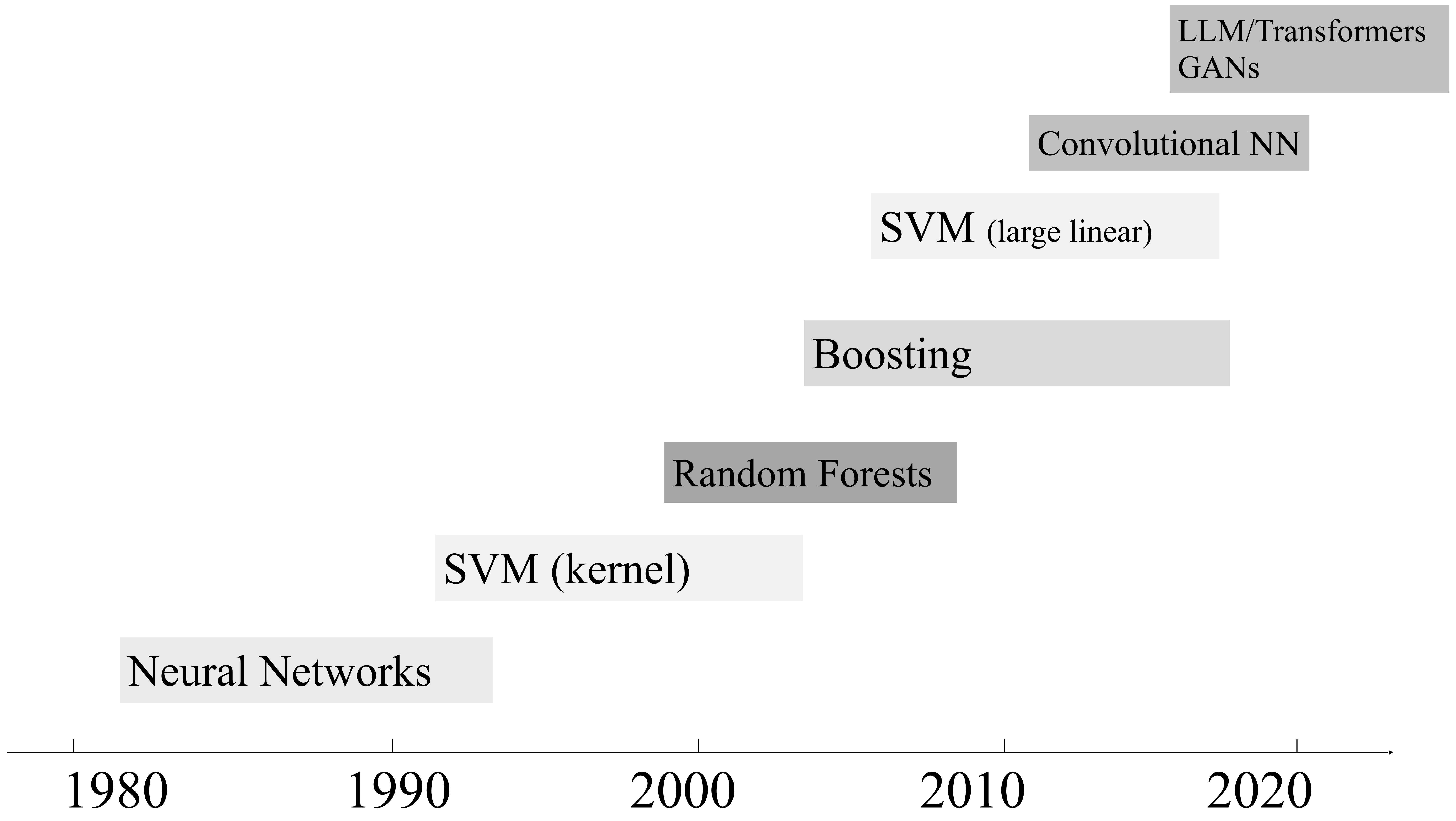
jfleischer@ucsd.edu

@jasongfleischer

https://jgfleischer.com

Slides in this presentation are from material kindly provided by Shannon Ellis and Sebastian Rashka

Bleeding edge ML methods



Boosting

Sequentially constructed ensembles

- Iterative process; each step a classifier is trained that cares more about correcting the mistakes made by the classifier at the last step
- Each classifier is a “weak learner”
- They are boosted into an ensemble that is a “strong learner”
- Two types of boosting, which differ in how weights are updated and classifiers combined
    - Adaptive
    - Gradient

Weak learners

- A decision stump
- k-NN
- Naive Bayes

Adaptive Boosting

e.g., AdaBoost

Freund, Y., & Schapire, R. E. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, 55(1), 119-139.

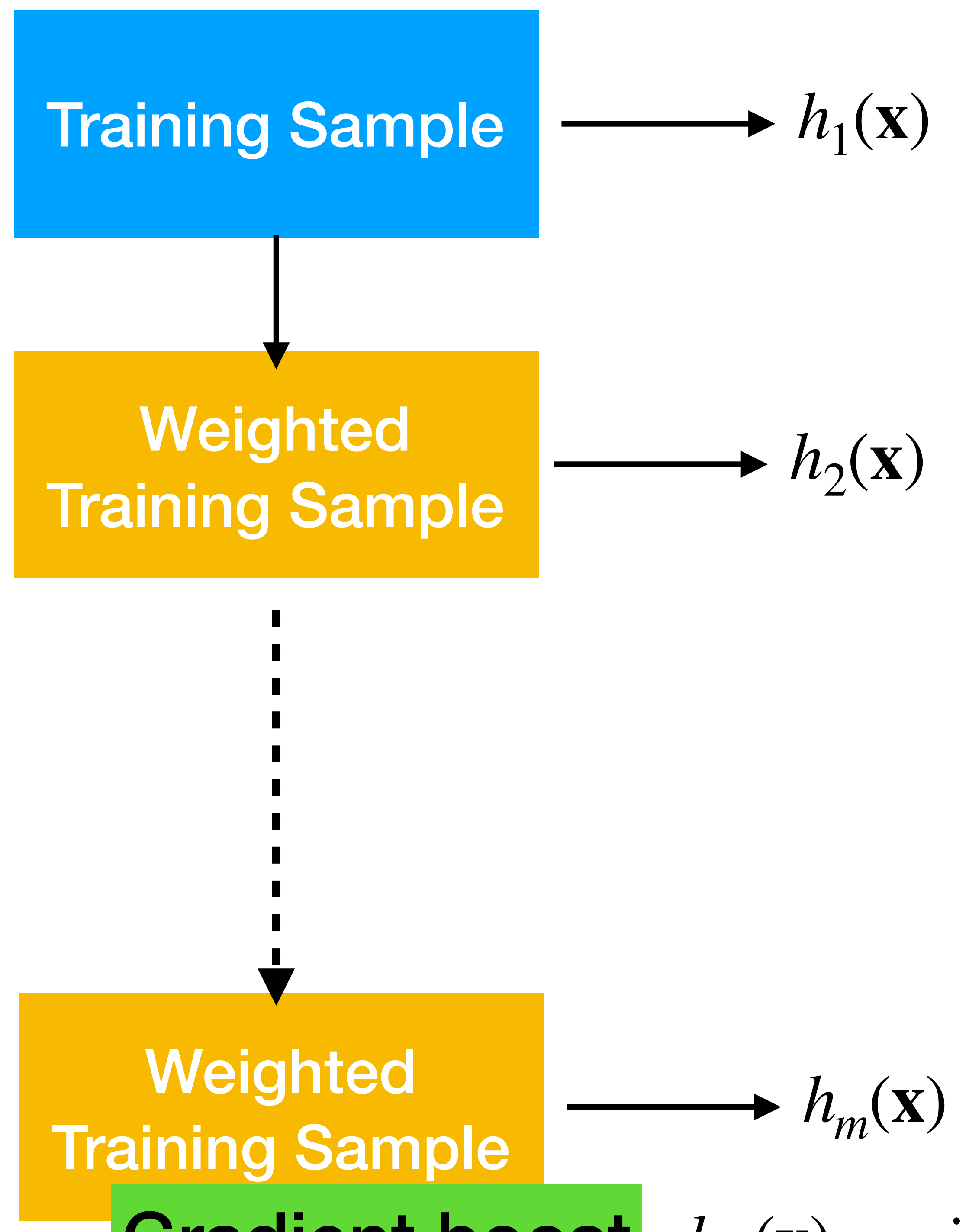
Gradient Boosting

e.g., LightGBM, XGBoost, scikit-learn's GradientBoostingClassifier

Friedman, J. H. (2001). Greedy function approximation: a gradient boosting machine. *Annals of statistics*, 1189-1232.

Chen, T., & Guestrin, C. (2016). Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International conference on knowledge discovery and data mining* (pp. 785-794). ACM.

General Boosting



Gradient boost $h_m(\mathbf{x}) = sign\left(\sum_{j=1}^{m} w_j \, h_j(\mathbf{x})\right)$ for $h(\mathbf{x}) \in \{-1,1\}$

AdaBoost $h_m(\mathbf{x}) = \arg\max_i \left(\sum_{j=1}^{m} w_j \, \mathbf{1}[h_j(\mathbf{x}) = i]\right)$ for $h(\mathbf{x}) = i, \quad i \in \{1,\ldots,n\}$

General boosting idea

- Initialize sample weighting with equal weights
- Loop until we hit boosting end conditions
  - Create a weak learner to learn a dataset where the misclassification error being minimized is sample weighted
  - Calculate classifier weighting based on the total misclassification error
  - Increase sample weighting for misclassified samples as a function of the classifier weighting
- Ensemble predicts a classifier weighted majority vote of the output of each classifier

```
Given: $(x_1, y_1), \ldots, (x_m, y_m)$ where $x_i \in \mathcal{X}$, $y_i \in \{-1, +1\}$.
Initialize: $D_1(i) = 1/m$ for $i = 1, \ldots, m$.
For $t = 1, \ldots, T$:
  • Train weak learner using distribution $D_t$.
  • Get weak hypothesis $h_t : \mathcal{X} \to \{-1, +1\}$.
  • Aim: select $h_t$ with low weighted error:
        $\varepsilon_t = \Pr_{i \sim D_t}\left[h_t(x_i) \neq y_i\right].$
  • Choose $\alpha_t = \frac{1}{2} \ln\left(\frac{1 - \varepsilon_t}{\varepsilon_t}\right)$.
  • Update, for $i = 1$ $m$:
        $D_{t+1}(i) = \frac{D_t(i)}{Z_t} \times \begin{cases} e^{-\alpha_t} & \text{if } y_i = h_t(x_i) \\ e^{\alpha_t} & \text{if } y_i \neq h_t(x_i) \end{cases}$
    where $Z_t$ is a normalization factor (chosen so that $D_{t+1}$ will be a distribution).
Output the final hypothesis:
        $H(x) = \operatorname{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right).$
```

Fig. 1 The boosting algorithm AdaBoost.

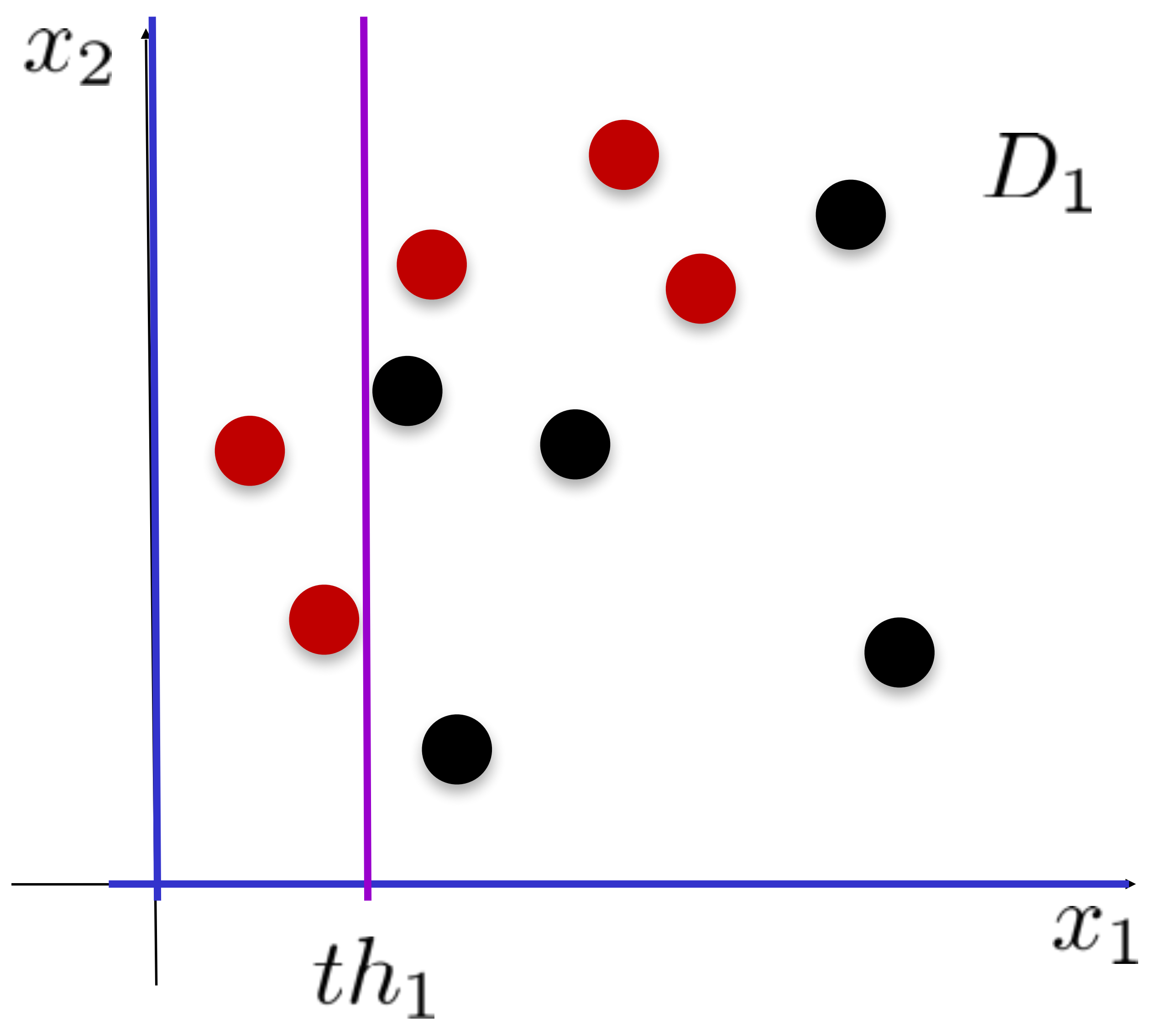
Toy Example

$$S_{training} = \{(\mathbf{x}_i, D_1(i), y_i), i = 1..n\} \qquad \mathbf{x} \in \mathbb{R}^2 \qquad y \in \{-1, +1\}$$

🔴 $y = +1$

⚫ $y = -1$

Minimize: $e_t = \sum_i D_t(i) \times \mathbf{1}(y_i \neq h_t(\mathbf{x}_i))$

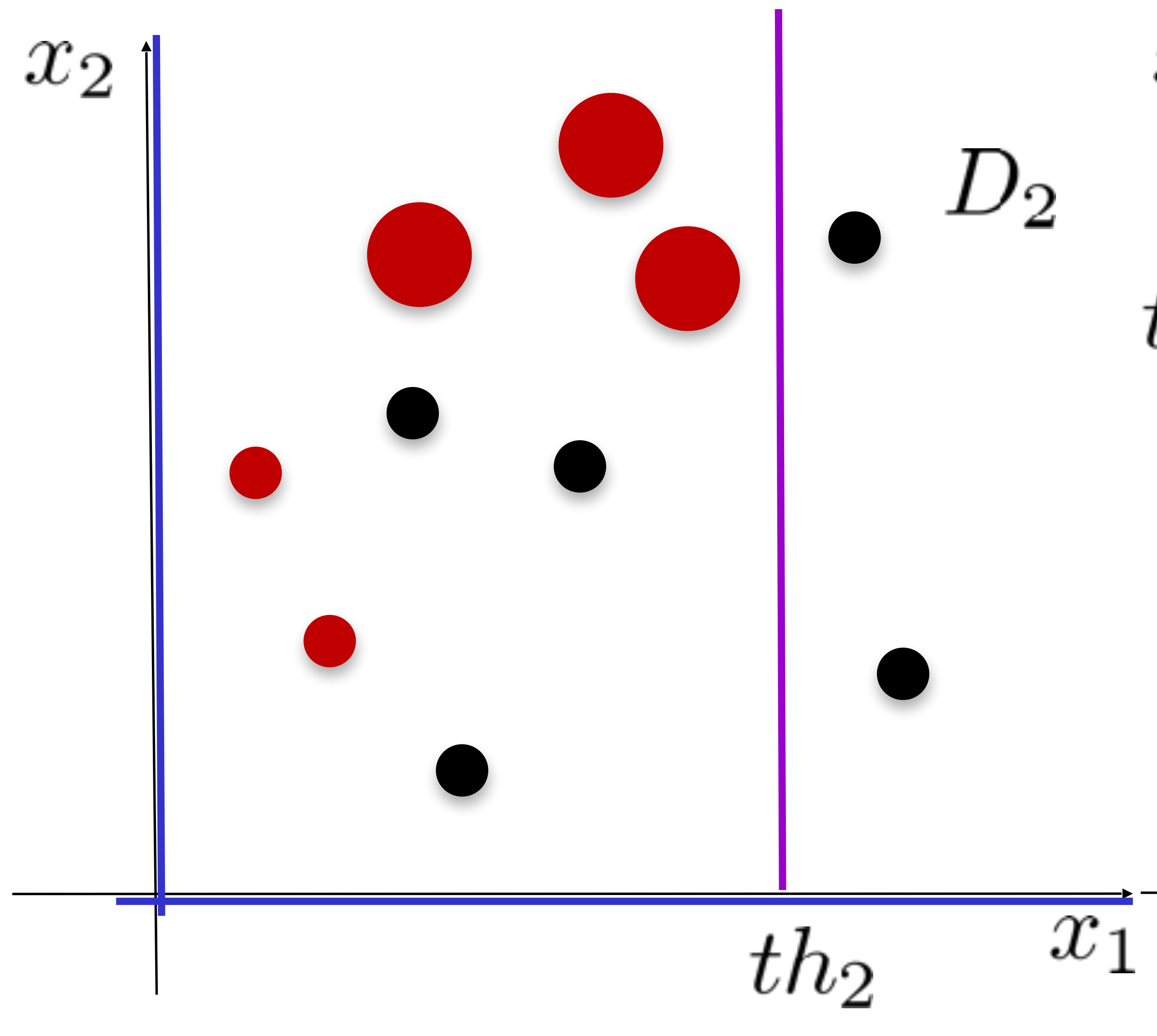


$$h_1(\mathbf{x}) = sign(x_1 \leq th_1)$$

$$\epsilon_1 = 3/9$$

$$\alpha_1 = \frac{1}{2}\ln(\frac{1-e_1}{e_1}) = 0.42$$

$$D_2(i) \propto D_1(i) \times \begin{cases} e^{-\alpha_1} & if\ y_i = h_1(\mathbf{x}_i) \\ e^{\alpha_1} & if\ y_i \neq h_1(\mathbf{x}_i) \end{cases}$$

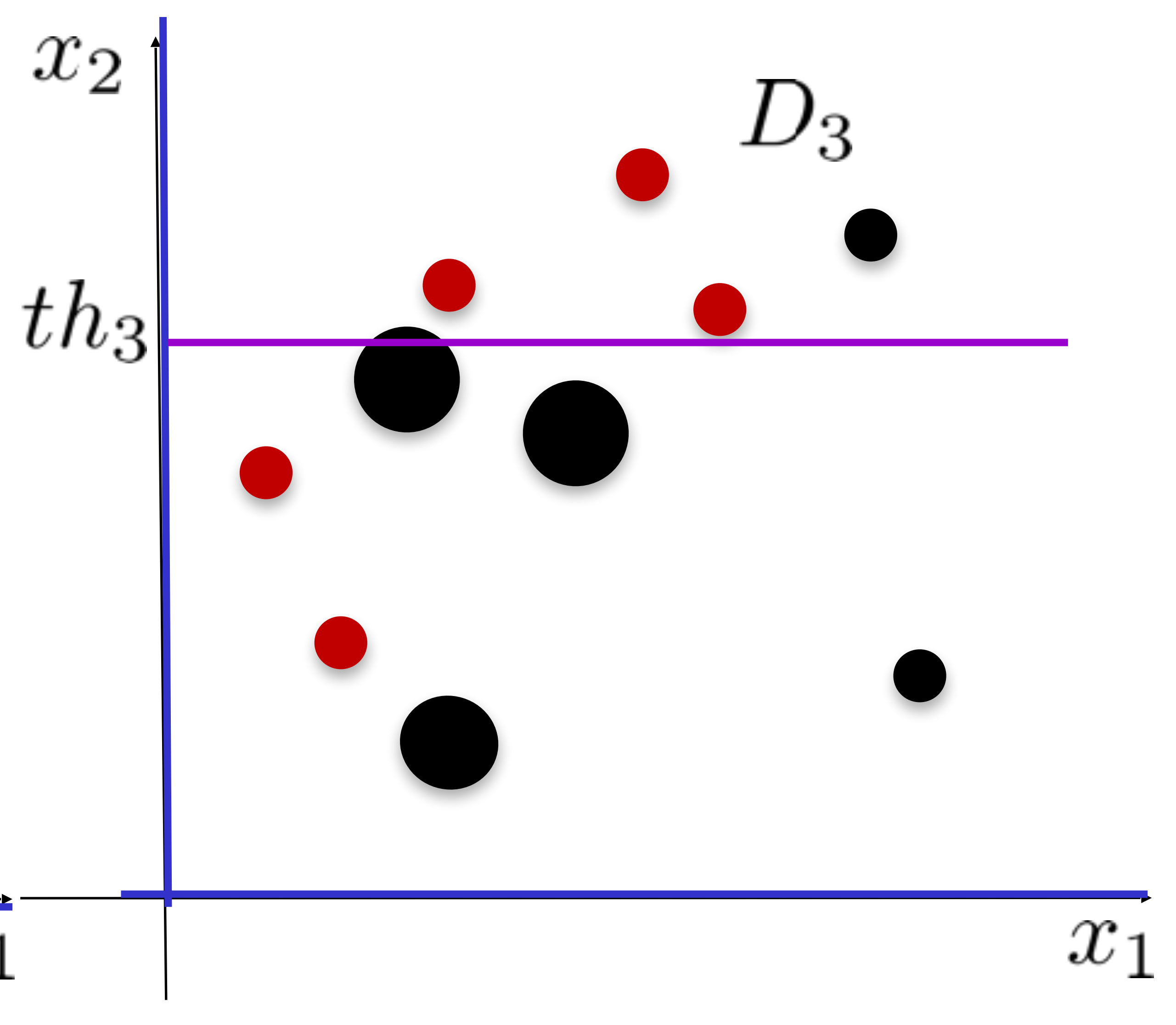


$$h_2(\mathbf{x}) = sign(x_1 \leq th_2)$$

$$\epsilon_2 = 0.215$$

$$\alpha_2 = \frac{1}{2}\ln(\frac{1-e_2}{e_2}) = 0.65$$

$$D_3(i) \propto D_2(i) \times \begin{cases} e^{-\alpha_2} & if\ y_i = h_2(\mathbf{x}_i) \\ e^{\alpha_2} & if\ y_i \neq h_2(\mathbf{x}_i) \end{cases}$$



$$h_3(\mathbf{x}) = sign(x_2 \geq th_3)$$

$$\epsilon_3 = 0.137$$

$$\alpha_3 = \frac{1}{2}\ln(\frac{1-e_3}{e_3}) = 0.92$$

$$D_4(i) \propto D_3(i) \times \begin{cases} e^{-\alpha_3} & if\ y_i = h_3(\mathbf{x}_i) \\ e^{\alpha_3} & if\ y_i \neq h_3(\mathbf{x}_i) \end{cases}$$

The final classifier: $H(\mathbf{x}) = sign(\sum_{t=1}^{T} \alpha_t h_t(\mathbf{x}))$

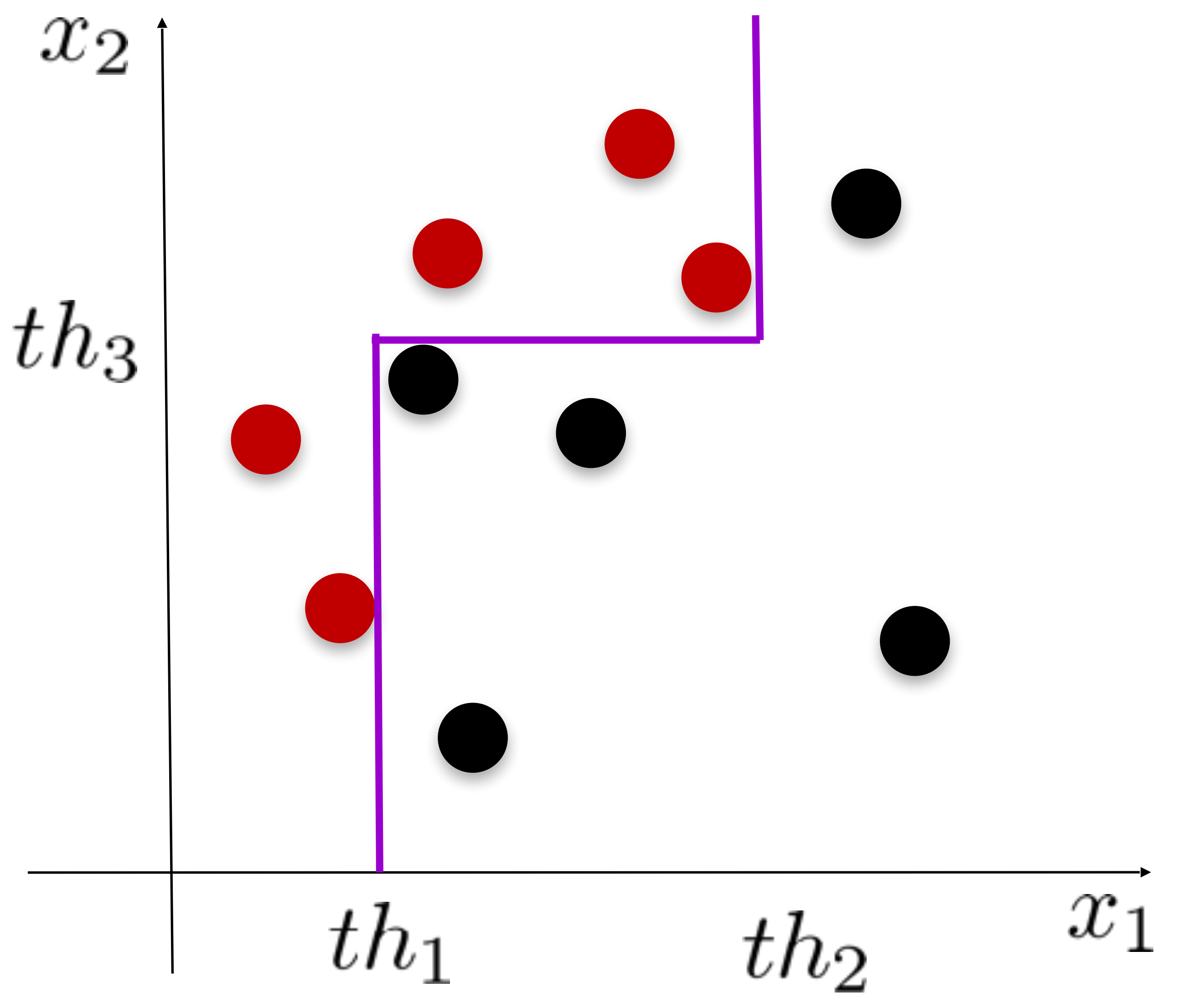
AdaBoost Example

$$S_{training} = \{(\mathbf{x}_i, y_i), i = 1..n\} \qquad \mathbf{x} \in \mathbb{R}^2 \qquad y \in \{-1, +1\}$$

● $y = +1$

● $y = -1$

Weak classifier: h decision stump.



$$H(\mathbf{x}) = 0.42 \times h_1(\mathbf{x}) + 0.65 \times h_2(\mathbf{x}) + 0.92 \times h_3(\mathbf{x})$$

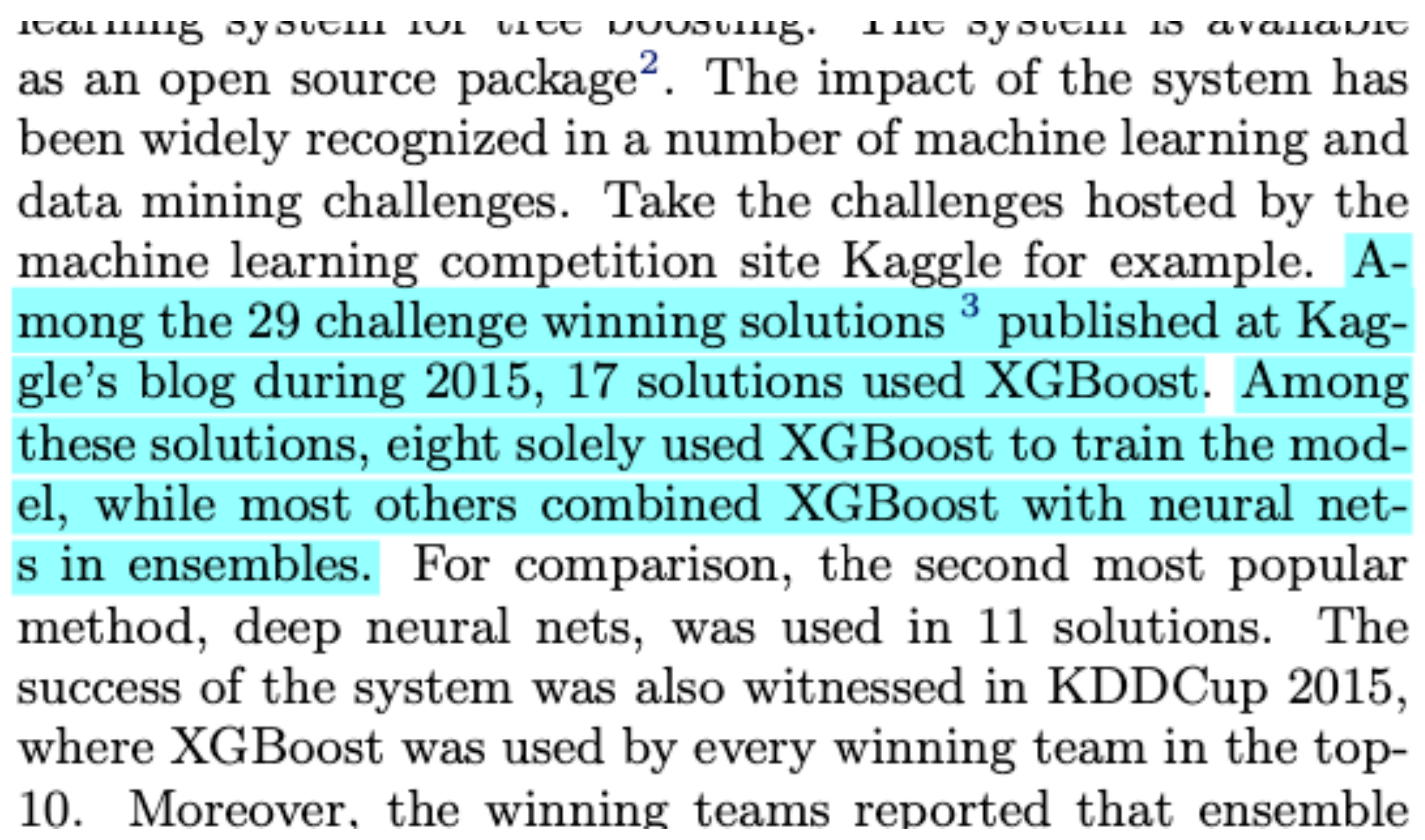
$$= 0.42 \times sign(x_1 \leq th_1) + 0.65 \times sign(x_1 \leq th_2) + 0.92 \times sign(x_2 \geq th_3)$$

XGBoost

Summary and Main Points:

- scalable implementation of gradient boosting
- Improvements include: regularized loss, sparsity-aware algorithm, weighted quantile sketch for approximate tree learning, caching of access patterns, data compression, sharding
- Decision trees based on CART
- Regularization term for penalizing model (tree) complexity
- Uses second order approximation for optimizing the objective
- Options for column-based and row-based subsampling
- Single-machine version of XGBoost supports the exact greedy algorithm

Chen, T., & Guestrin, C. (2016, August). Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining* (pp. 785-794). ACM.



learning system for tree boosting. The system is available as an open source package[2]. The impact of the system has been widely recognized in a number of machine learning and data mining challenges. Take the challenges hosted by the machine learning competition site Kaggle for example. Among the 29 challenge winning solutions [3] published at Kaggle's blog during 2015, 17 solutions used XGBoost. Among these solutions, eight solely used XGBoost to train the model, while most others combined XGBoost with neural nets in ensembles. For comparison, the second most popular method, deep neural nets, was used in 11 solutions. The success of the system was also witnessed in KDDCup 2015, where XGBoost was used by every winning team in the top-10. Moreover, the winning teams reported that ensemble

Chen, T., & Guestrin, C. (2016, August). Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining* (pp. 785-794). ACM.

More GBM Implementations

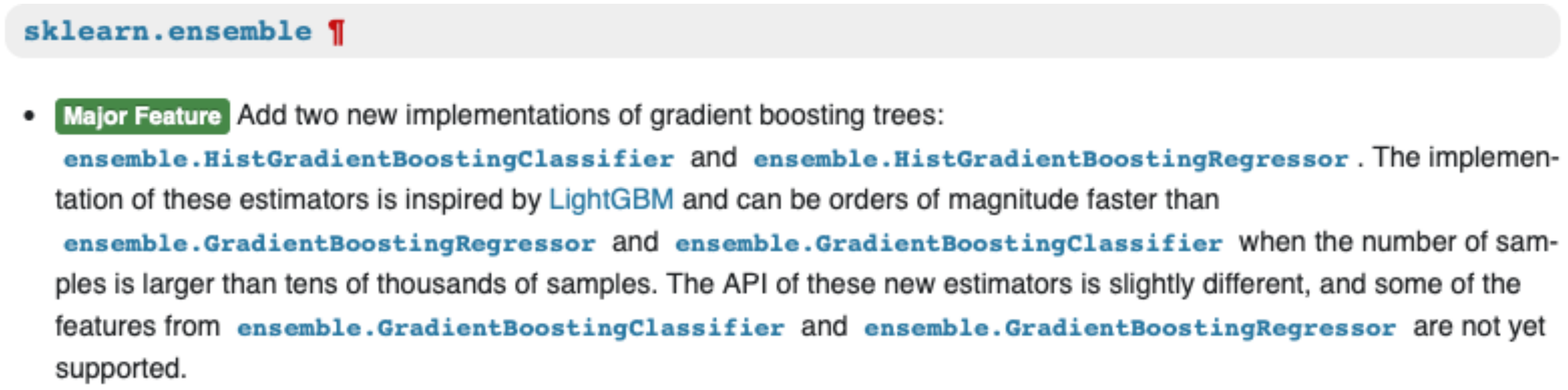
LightGBM, Light Gradient Boosting Machine

From https://github.com/Microsoft/LightGBM:

- Faster training speed and higher efficiency
- Lower memory usage
- Better accuracy
- Support of parallel and GPU learning
- Capable of handling large-scale data

Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., ... & Liu, T. Y. (2017). Lightgbm: A highly efficient gradient boosting decision tree. In *Advances in Neural Information Processing Systems* (pp. 3146-3154).

https://scikit-learn.org/stable/whats_new.html#version-0-21-0



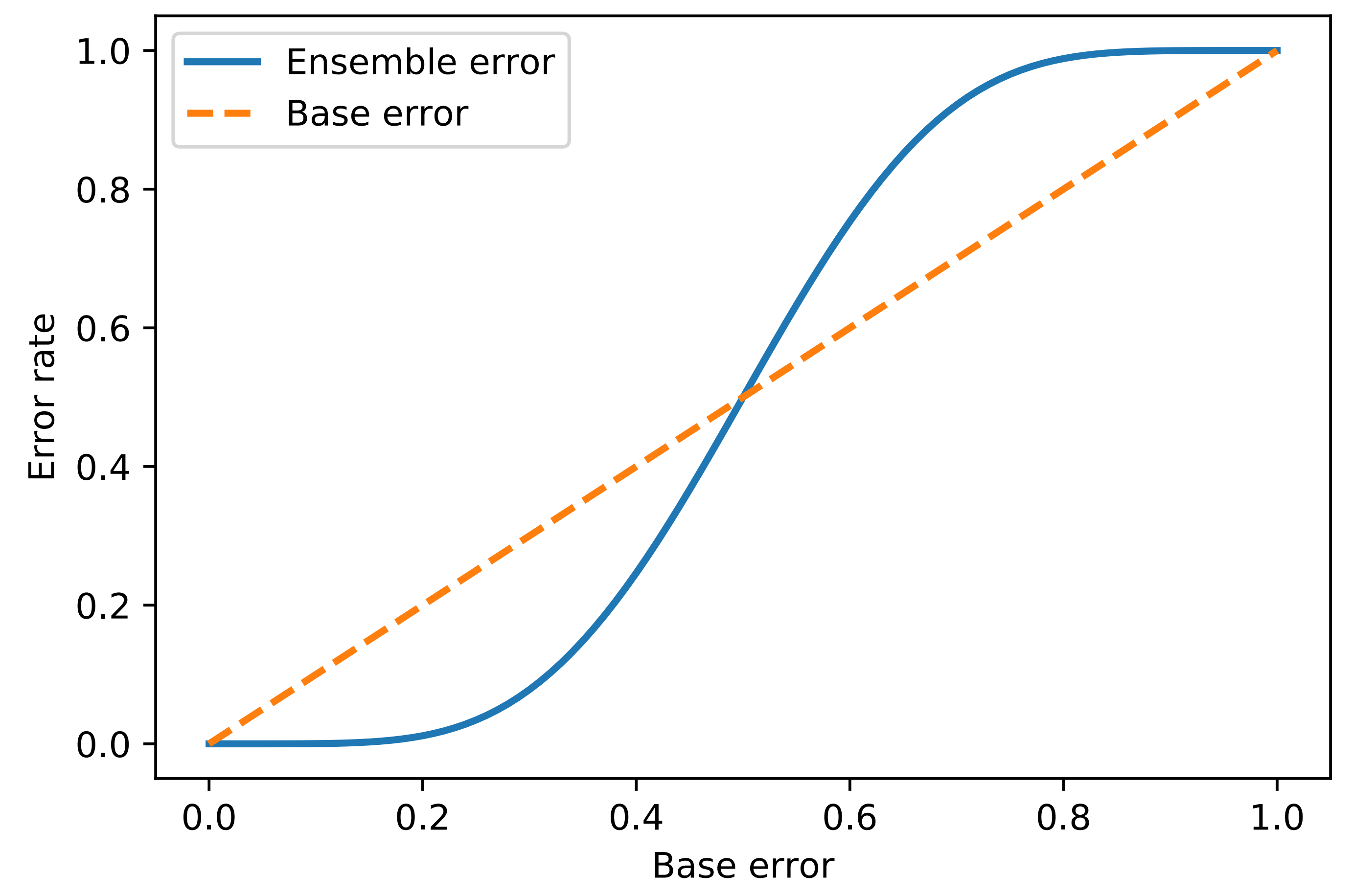
`sklearn.ensemble` ¶

- Major Feature Add two new implementations of gradient boosting trees: `ensemble.HistGradientBoostingClassifier` and `ensemble.HistGradientBoostingRegressor`. The implementation of these estimators is inspired by LightGBM and can be orders of magnitude faster than `ensemble.GradientBoostingRegressor` and `ensemble.GradientBoostingClassifier` when the number of samples is larger than tens of thousands of samples. The API of these new estimators is slightly different, and some of the features from `ensemble.GradientBoostingClassifier` and `ensemble.GradientBoostingRegressor` are not yet supported.

Ensemble summaries

- A necessary and sufficient condition for an ensemble to be more accurate than any of its individual classifiers is that those classifiers are accurate and diverse [Hansen & Salamon, *IEEE Trans Pattern Anal Mach Intell* 1990]
- An accurate classifier has a lower error rate than uniform random guessing
- Diverse classifiers make different errors when generalizing to new data.

$$\epsilon_{ens} = \sum_{k}^{n} \binom{n}{k} \epsilon^k (1-\epsilon)^{n-k}$$



Ensemble summaries

Classifier diversity can be achieved by injecting randomness via:

- Subsetting the training examples
- Subsetting the input features
- Non-deterministic algorithms

Ensemble summaries

- Ensembles often provide a boost in accuracy over base learner
- Increase runtime over base learner, but compute cycles are usually much cheaper than training instances
- Some ensemble approaches (e.g. bagging, random forests) are easily parallelized
- Prediction contests (e.g. Kaggle, Netflix Prize) usually won by ensemble solutions
- Ensemble models are usually low on the comprehensibility scale, although see work by [Craven & Shavlik, NIPS 1996] [Domingos, Intelligent Data Analysis 1998] [Van Assche & Blockeel, ECML 2007]