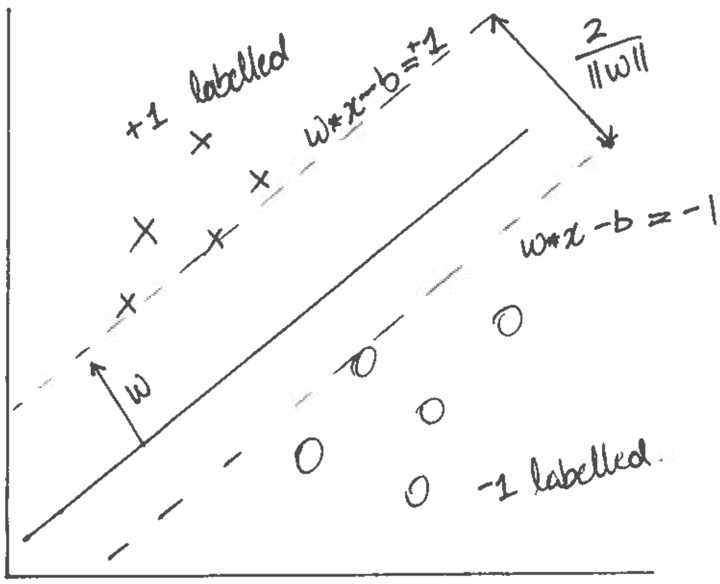
SVM. Training data: $\langle \vec{x}_i, y_i \rangle$; $i = 1 \ldots n$



Linearly separable data

Anything above $\vec{w}^T \vec{x} - b = 1$ is a "x" datapoint

Maximize $\frac{2}{\|\vec{w}\|} \Rightarrow$ Minimize $\|w\|_2$

$\Rightarrow$ If $y_i = 1$, $\vec{w}^T \vec{x}_i - b \geq 1$
$y_i = -1$, $\vec{w}^T \vec{x}_i - b \leq -1$ } Points must be correctly classified

$\Rightarrow y_i (\vec{w}^T \vec{x}_i - b) \geq 1 \quad \forall \; 1 \leq i \leq n$ Constraint

Optimization problem: Quadratic optimization, solve w/ Lagrange multipliers.
↳ Quadratic $\Rightarrow$ single global minima. If we remove support vectors, hyperplane changes!

$$\min_{w,b} \|\vec{w}\|_2^2$$

sub. ~~$y_i \geq$~~ $y_i (\vec{w}^T \vec{x}_i - b) \geq 1 \quad \forall i \in \{1 \ldots n\}$.

↑ hyperplane is determined completely by nearest x_i on either side!

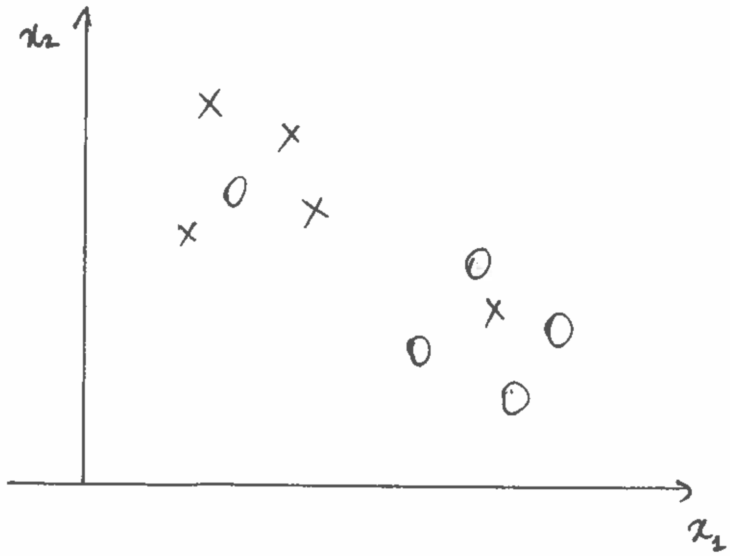
$\Rightarrow$ Most difficult pts to classify

Use the optimization of maximizing margin to reduce number of nonzero weights ~~that~~ $\Rightarrow$ only look at weights that matter $\Rightarrow$ i.e. correspond to the support vectors!

Key diff b/w SVM & $\underbrace{\text{NNs/linear reg}}_{\text{use all data!}}$

Ultimately, classifier: $\text{sign}(\vec{w}^T\vec{x} - b)$ for some new x, using learned / optimized $\vec{w}, b$.

"Fuzzy" data



Some points' labels are "misclassified".

Modify minimization: (using hinge loss)

$$\max\left(0,\ 1 - y_i(\vec{w}^T\vec{x}_i - b)\right)$$

returns 0 if point is correctly classified/labelled
(0,1), proportional to distance from margin if incorrectly classified.

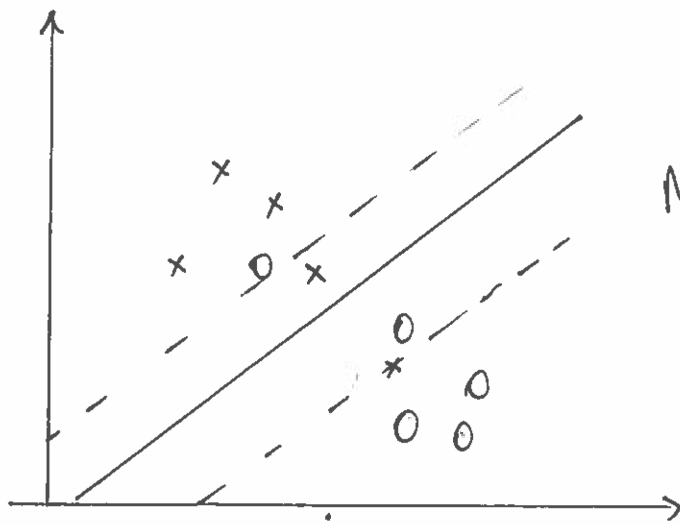
$$\Rightarrow \text{minimize} \quad \lambda\|\vec{w}\|_2^2 + \left[\frac{1}{n}\sum_{i=1}^{n} \max\left(0,\ 1 - y_i(\vec{w}^T\vec{x}_i - b)\right)\right]$$

Rewrite: $$\min_{w,b,\zeta} \|\vec{w}\|_2^2 + C\sum_{i=1}^{n}\zeta_i$$

Riemann zeta function

sub. $y_i(\vec{w}^T\vec{x}_i - b) \geq 1 - \zeta_i, \quad \zeta_i \geq 0 \quad \forall\, i \in \{1 \dots n\}$.

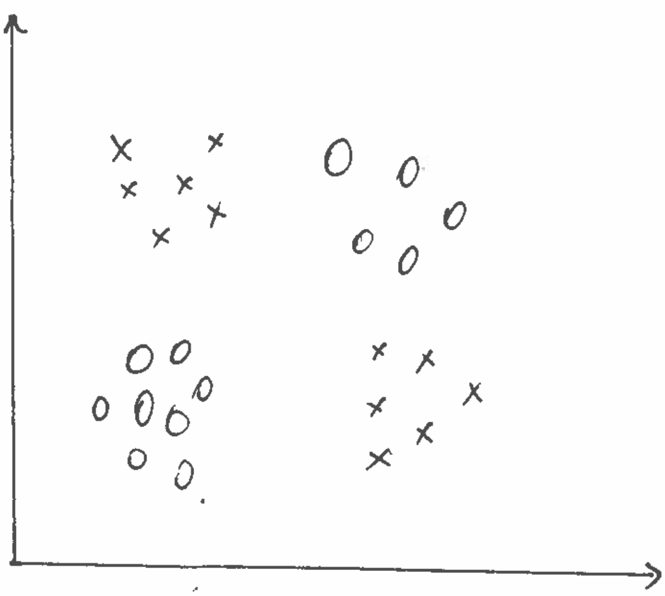
⇒ Our learned classifier now looks like:



Number of support vectors = 4

Very similar to linear discriminant analysis (LDA)

Non-linearly separable Data:



XOR dataset

Gain linear separation by mapping data into a high-dimensional space

We actually solve SVM optimization via the dual formulation (removes dependence on $\vec{w}, b$).

Dual problem -

Interchange variables & constraints :

$$\begin{aligned} &\min \ c \circ x \\ &\text{sub. } A \circ x = b \\ &\qquad x \geq 0. \end{aligned} \longrightarrow \begin{aligned} &\max \ b \circ y \\ &\text{sub. } A^T \circ y \leq c. \\ &\qquad y \text{ Free. (Unconst.)} \end{aligned}$$

Here :

$$\begin{aligned} &\min \ \|w\|_2^2 \longrightarrow \\ &\text{sub. } y_i.(\vec{w}^T \vec{x}_i - b) .. \end{aligned} \quad \boxed{\begin{aligned} &\min. \ f(\vec{w}) \\ &\text{sub. } g_j(\vec{w}) \leq 0. \\ &\qquad h_k(\vec{w}) = 0. \end{aligned}}$$

General form (Lin sep. / fuzzy).

$\downarrow$ Dual formulation

$$\mathcal{L} \equiv \frac{1}{2} f(\vec{w}) + \sum_j a_j g_j(\vec{w}) + \cancel{\sum_k \beta_k h_k(\vec{w})}$$

0 $\because$ no equality const.

(Method of Lagrange multipliers)

$\Rightarrow$ For a min / max:

$$\frac{\partial \mathcal{L}}{\partial w_i} = 0 \quad ; \quad \frac{\partial \mathcal{L}}{\partial \beta_k} = 0$$

From strong duality & Kuhn - Tucker thms.,

$$\begin{aligned} &\max. \ \mathcal{L}(\alpha, \beta). \\ &\text{sub. } \alpha_j \geq 0 \quad \forall j. \end{aligned} \quad \equiv \quad \text{original primal form.}$$

Generally true $\forall$ convex fns. $f(w)$.

Further, if $\hat{w}$ is optimal soln of primal & $\hat{a}$ & $\hat{\beta}$ are optimal solns of dual:

$$\left.\begin{aligned} f(\hat{w}) &= \mathcal{L}(\hat{a}, \hat{b}) \\ \hat{a}_j g_j(\hat{w}) &= 0 \quad \forall j \end{aligned}\right\} \text{Karush-~~Kuhn~~ Kuhn-Tucker (KKT) complementarity condition.}$$

If we sub. our original constraints for the SVM problem into the Lagrangian,

$$\mathcal{L} = \frac{1}{2}\vec{w}\cdot\vec{w} + \sum_i \alpha_i \left[1 - y_i(\vec{w}\cdot\vec{x}_i + b)\right]$$

For max/min:

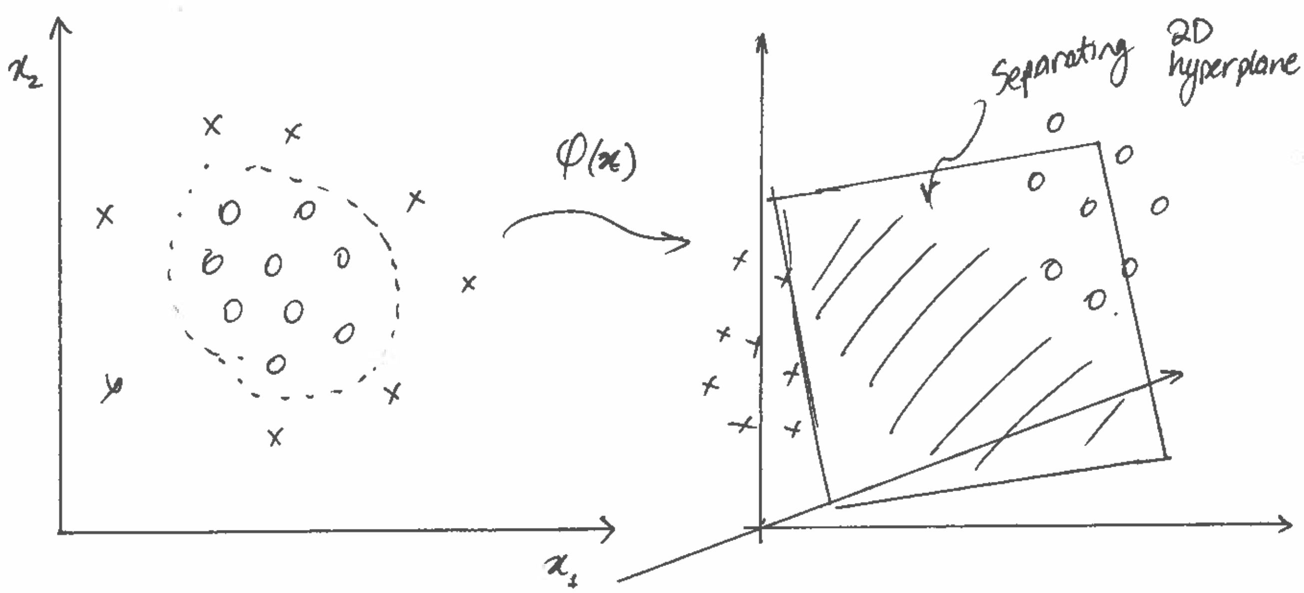
$$0 \overset{set}{=} \frac{\partial \mathcal{L}}{\partial \vec{w}} = \vec{w} - \sum_i \alpha_i y_i \vec{x}_i \qquad \text{and} \qquad 0 = \frac{\partial \mathcal{L}}{\partial b} = \sum_i \alpha_i y_i$$

$$\Rightarrow \hat{w} = \sum_i \hat{\alpha}_i y_i \vec{x}_i$$

$$\Rightarrow \mathcal{L}(\alpha) = \sum_j \alpha_j - \frac{1}{2}\sum_{j,k} \alpha_j y_j (\vec{x}_j \cdot \vec{x}_k) y_k \alpha_k$$

The only place where $\vec{x}$ shows up is here!
This is the only thing that scales w/ number of features. $\Rightarrow$
all other parts scale w/ num. pts. $\Rightarrow$ Primal to dual $\rightarrow$ shift scaling from number (primal) of features to number of pts (dual!)
$\Rightarrow$ Favors data w/ $\downarrow$ numbers of data pts., but huge features

Now, lets talk about non linearity. $\Rightarrow$ "Kernel Trick".



Primal form: $\min. \frac{1}{2}\vec{W}\cdot\vec{W} + \lambda\sum_i \Xi_i$

sub. $y_i(\vec{W}\cdot\varphi(\vec{x}_i) + B) \geq 1 - \Xi_i \qquad i = 1\ldots m.$

But, embedding dim may be $\mathbb{R}^{\text{Billions}}$!

$\Longrightarrow$ Intractable!

Dual problem: ~~min $\frac{1}{2}\alpha \cdot \text{diag}(y)$~~

$$\min. -\sum_j \alpha_j + \frac{1}{2}\sum_{j,k}\alpha_j y_j K_{jk} y_k \alpha_k$$

$$K_{jk} = K(x_j, x_k) = \varphi(x_i) \cdot \varphi(x_k)$$

dot prod.

But, we don't need to explicitly know mapping $\varphi(x)$!

All we need is some way to compute K_{jk} that could have come from some $\varphi(x)$! $\Rightarrow$ $m \times m$ matrix w/ mathematical properties of inner product!

- $K_{ij} = K(x_i, x_j)$ must be symm. in i, j, and nonneg. eigenval.
- K can be composed by addition, mult., scaling by const.

Popular Kernels:

linear: $K(x_i, x_j) = x_i \cdot x_j$.

power: $K(x_i, x_j) = (x_i \cdot x_j)^d \qquad 2 \leq d \leq 20$ (usually).

polynom: $K(x_i, x_j) = (a\, x_i \cdot x_j + b)^d$

sigmoid: $K(x_i, x_j) = \tanh(a x_i \cdot x_j + b)$.

Gaussian rbf: $K(x_i, x_j) = e^{-\frac{1}{2}\frac{|x_i - x_j|^2}{\sigma^2}}$

Tips for using Kernels:

- Gaussian rbf is very popular, $\because$ only 1 hyperparam.

  Guess good initial σ via avg. distance b/w points in feature space.

- Polynomial kernels:

  Start by choosing a, b s.t. $a x_i \cdot x_j + b$ b/w $-1, 1$ $\forall i, j$.

  $d \Rightarrow$ roughly interpret as number of features to be mixed ~~to~~ during partitioning.

  $\Rightarrow d=1 \Rightarrow$ partition space by 1 feature at a time

  $2 \Rightarrow$ 2 features ...

Diff b/w power & polynomial: power considers only d features at once, polynomial considers all combos of d or fewer features.

Decision Tree Models

Start w/ training data, labels

$x_i \in \mathbb{R}^n, i = 1 \ldots N$ data

$y \in \mathbb{R}^N$ labels.

No index, just concat. all labels into one big vector.

Ie., each ~~elt~~ split element of $\mathbb{R}^n$

We want to partition the space of features S.T. the x_i's w/ the same y are grouped together.

Let's represent the data @ some node m w/ Q_m, w/ n_m samples.

made up of a set of x_i, y labels

Then, represent each split as $\theta = (j, t_m)$

Feature (Ie. index of each x_i element)

Threshold value.

$\Rightarrow$ Partition data into left & right subsets:

$Q_m^{left}(\theta) = \{(x, y) \mid x_j \leq t_m\}$. (Below or equal to threshold)

$Q_m^{right}(\theta) = Q_m \setminus Q_m^{left}(\theta)$ (everything else)

Then, define a quality score based on a loss function:

$$G(Q_m, \theta) = \frac{n_m^{left}}{n_m} H(Q_m^{left}(\theta)) + \frac{n_m^{right}}{n_m} H(Q_m^{right}(\theta))$$

Then, ~~sel~~ select θ^* that min. G: $\theta^* = \operatorname{argmin}_\theta G(Q_m, \theta)$ ~~$\operatorname{argmin}_\theta G(Q_m(\theta))$~~

To define H, define a measure for the proportion of class k observations in node m:

$$p_{mk} = \frac{1}{n_m} \sum_{y \in Q_m} I(y=k)$$

$\Rightarrow$ Gini loss: $H(Q_m) = \sum_k p_{mk}(1-p_{mk})$

Log loss: $H(Q_m) = -\sum_k p_{mk} \log(p_{mk})$

Random Forest: & XG Boost.

General idea: combine influence of several different decision tree models.

RF :

Build a forest where each tree uses 1) a random subset of features -> low correlation b/w trees.

"Random subspace method"

Sklearn implementation - use all features or random subset of features.

2) Each tree is built from ~~f~~ a sample drawn w/ replacement from the training set.

Net effect: decrease variance of ensemble, reduce overfitting.
Combine diverse trees, at cost of increasing model bias.

If training data is $[x_1, x_2, x_3, x_4, x_5, x_6]$,
one tree gets $[x_1, x_1, x_3, x_4, x_4, x_4, \cancel{x_6}]$. same size

Sampled with replacement

"Bagging" / "Bootstrapping"

Form a prediction w/ a consensus of trees.

XGBoost

Additive trees: classifier & regressor trees

XG CART:

Group Members

Louis

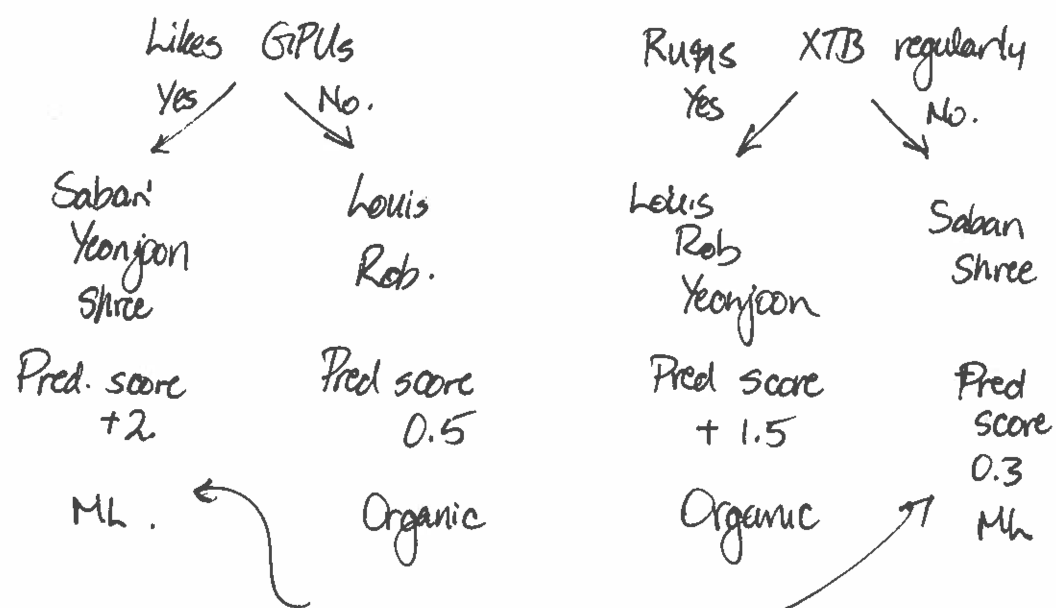
Rob.

Sabari

Yeonjoon

Shree.

Interests



$f(\text{Sabari}) = 2 + 0.3 = 2.3.$

Mathematically, $\hat{y}_i = \sum_{k=1}^{K} f_k(x_i)$, $f_k \in \mathcal{F} \Rightarrow \mathcal{F}$ is set of all possible CARTs.

$K \Rightarrow$ total number of trees.

$$\Rightarrow \min \cancel{obj(\theta)}\ obj = \underbrace{\sum_{i=1}^{N} l(y_i, \hat{y}_i)}_{\text{Loss fn. over data.}} + \underbrace{\sum_{i=1}^{K} \omega(f_k)}_{\text{Regularization over trees}}$$

$\rightsquigarrow$ Controls tree complexity

③.

What are parameters of trees $\Rightarrow$ what's "f"?

↳ Structure of tree + leaf scores!

This isn't a gradient optimization problem:

Learning on space of all possible trees is totally intractable!

Instead, additive predictions: Fix what we have learned, add one tree at a time.

$$\cancel{\hat{y}_i(} \quad \hat{y}_i^{0} = 0$$

$$\hat{y}_i^{1} = f_1(x_i) := \hat{y}_i^{0} + f_1(x_i)$$

First CART tree.

$$\hat{y}_i^{2} = \overset{f_1(x_i)+f_2(x_i)}{\cancel{f_2(x_i)}} := \hat{y}_i^{(1)} + f_2(x_i)$$

$$\vdots$$

$$y_i^{(t)} = \cancel{f} \sum_{k=1}^{t} f_k(x_i) := \hat{y}_i^{(t-1)} + f_t(x_i)$$

Into objective fn:

$$obj^{(t)} = \sum_{i=1}^{n} l(y_i, \hat{y}_i^{(t)}) + \sum_{i=1}^{t} \omega(f_i)$$

$$= \sum_{i=1}^{n} l(y_i, \hat{y}_i^{(t-1)} + f_t(x_i)) + \omega(f_t) + c$$

we can only optimize structure of t^{th} tree! Roll other ω's into const.

If we apply MSE as the loss fn:

$$obj^{(t)} = \sum_{i=1}^{n} (y_i - (\hat{y}_i^{(t-1)} + f_t(x_i)))^2 + \omega(f_t) + c$$

$$\cancel{= y_i^2 \, 2}$$

$$= \sum_{i=1}^{n} \Big(\underset{\text{const}}{\cancel{y_i^2}} - 2y_i(\underset{\text{const}}{\cancel{\hat{y}_i^{t-1}}} + f_t(x_i)) + \underset{\text{const}}{\cancel{(\hat{y}_i^{t-1})^2}} + 2\hat{y}_i^{(t-1)} f_t(x_i) + (f_t(x_i))^2\Big)$$

$$+ \omega(f_t) + c$$

$$= \sum^{n} \left[2(\hat{y}_i^{(t-1)} - y_i) f_t(x_i) + (f_t(x_i))^2\right] + \omega(f_t) + c$$

④.

$$\text{obj}^{(t)} = \sum_{t=1}^{n} \left[2(\hat{y}^{(t-1)} - y_i) f_t(x_i) + (f_t(x_i))^2 \right] + \omega(f_t) + c.$$

1st order term (residual)

2nd order term

We got a nice form in the case of MSE; but for other loss fns this isn't the case! So, in general, use Taylor expansion:

$$\text{obj}^{(t)} = \sum_{t=1}^{n} \left[l(y_i, \hat{y}_i^{(t-1)}) + g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i) \right] + \omega(f_t) + c$$

where $g_i = \dfrac{\partial l(y_i, \hat{y}_i^{(t-1)})}{\partial \hat{y}_i^{(t-1)}}$; $h_i = \dfrac{\partial l(y_i, \hat{y}_i^{(t-1)})}{\partial y_i^{(t-1)}}$ "gradient boosting"

Constants don't affect minimization procedure:

$$\Rightarrow \text{obj}^{(t)} = \sum_{t=1}^{n} \left[g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i) \right] + \omega(f_t)$$

XGBoost defines $\omega(f)$ as:

$$\omega(f) = \gamma T + \frac{1}{2} \lambda \sum_{j=1}^{T} \omega_j^2$$

Total num of trees

Vector of scores on each leaf

Gamma fn

hyperparam

Rewrite & compress: $\text{obj.}^{(t)} = \sum_{j=1}^{T} \left[G_j \omega_j + \frac{1}{2}(H_j + \lambda)\omega_j^2 \right] + \gamma T.$

SHAP Analysis

SHAP ⇒ "Shapley Additive Explanations"

Comes from cooperative game theory:

A number of players cooperate to achieve an objective, that leads to some overall gain.

But, some players may contribute more

One might have more bargaining power

Another might threaten to destroy everything

⇒ Question: what amount of the overall gain should be assigned to each player?

· Ie. how important is each player to the overall gain, and how much payoff should each player expect

/feature

John: Imagine the group being formed one person at a time. Each person demands their contribution as "fair compensation".

Then, for each person, average contribution over the different ways we can form the group.

For ML: Assume only some features are present, while others aren't

>

Shapley values are only attribution method that satisfies:

1) Efficiency: Feature contributions sum to the diff of a prediction x and the average:

$$\sum_{j=1}^{p} \phi_j = \hat{f}(x) - E_X(\hat{f}(x))$$

Expectation value.

2) Symmetry: Contributions of features j & k are the same if they contribute equally to all possible groups

If $val(S \cup \{j\}) = val(S \cup \{k\})$.

$$\forall \; S \subseteq \{1, \dots p\} \setminus \{j, k\}$$

Group w/o j, k.

Then $\phi_j = \phi_k$.

3) Dummy: A feature j that doesn't change predicted value, regardless of group, gets a value of 0.

If $val(S \cup \{j\}) = val(S) \;\; \forall \; S \subseteq \{1, \dots p\}$.

$$\phi_j = 0$$

Additivity: For a objective w/ combined payouts, ~~the~~ SHAP values are additive:

$$\phi_j + \phi_j^+ \Rightarrow \cancel{\text{the}}\ \text{combined value.}$$

This is important, since for RF/ensemble models, additivity guarantees that if we calc. SHAP val. for each indiv. tree & average → SHAP for forest