Supervised, Unsupervised and Reinforcement Learning in Finance

Week 1: Supervised Learning

Support Vector Machines

Igor Halperin

NYU Tandon School of Engineering, 2017

Support Vector Machines

- SVM was developed as a classification method by Vapnik et al. in 1992 within the framework of statistical learning theory
- SVM was generalized to handle regression problems by Vapnik et al in mid 1990s (Support Vector Regression, or SVR for short). We will collectively refer to both SVM and SVR as SVM
- SVM is based on simple and beautiful geometric ideas (maximum margin hyperplane classifiers). Computationally, SVM amounts to convex optimization unique soln
- SVM handles both linear and non-linear regression. Unlike NN where non-linearity is implicit via the choice of both architecture and parameters, SVM controls non-linearity explicitly via the choice of the kernel
- SVM usually shows better out-of-sample performance than alternative methods including shallow NN for small & mid-size data
- SVM is widely used nowadays for classification and regression analyses such as object identification, text recognition, bioinformatics, speech recognition, etc. Previous financial applications mostly dealt with stock price predictions or bankruptcy predictions

Diff how errors are penalized

Sketch of SVM math

- Start with linear regression: → square loss fn

$$f(x) = \langle w, x \rangle + b \ , \ w \in \mathcal{X}, \ b \in \mathbb{R}$$

where $\mathcal{X}$ is the input pattern space (e.g. $\mathcal{X} = \mathbb{R}^d$), $\langle a, b \rangle$ is a dot product in $\mathcal{X}$. Want to fit a function $f(x)$ admitting at most deviation of ε from the data

- Flatness means a small w - can be formulated as quadratic optimization min $\frac{1}{2}||w||^2$ subject to constraints $y_i - \langle w, x_i \rangle - b \leq \varepsilon$ and $\langle w, x_i \rangle + b - y_i \leq \varepsilon$

- To make the problem feasible, introduce slack variables ξ_i, ξ_i^*, and reformulate the problem as follows:

→ $$\min \ \frac{1}{2}||w||^2 + C \sum_i (\xi_i + \xi_i^*)$$

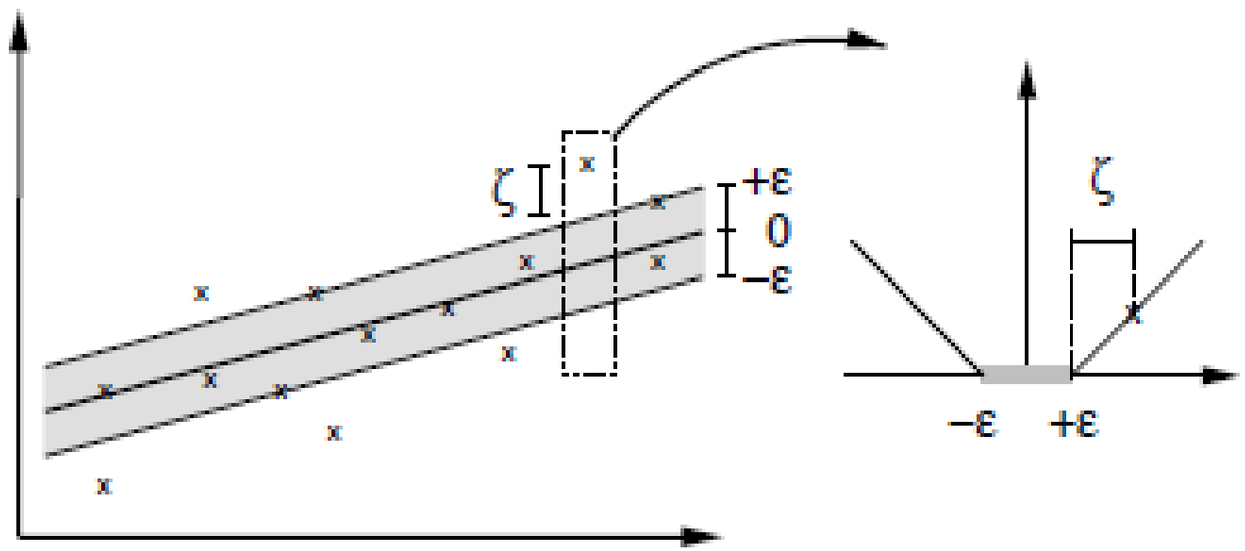
subject to $y_i - \langle w, x_i \rangle - b \leq \varepsilon + \xi_i$ and $\langle w, x_i \rangle + b - y_i \leq \varepsilon + \xi_i^*$, with $\xi_i, \xi_i^* \geq 0$. Parameter $C > 0$ determines the trade-off between flatness and our tolerance for deviations larger than ε.

Small C → many points outside

large C → fitted line varies with data

Sketch of SVM math (cont-ed)

- Geometric interpretation: ε-insensitive loss function. Deviations exceeding ε are penalized in a linear fashion



More noise tolerant within!

- The solution is found in terms of dual variables α_i, α_i^* (Lagrange multipliers) conjugate to the constraints:

$$f(x) = \sum_i (\alpha_i - \alpha_i^*)\langle x_i, x\rangle + b$$

  Inside the ε-tube, α_i, α_i^* vanish (KKT condition), so this is a sparse expansion in *support vectors* - hence the name SVM

- Dependence on x enters only through the dot product

Control question

Select all correct answers

1. Support Vector Machines use vectors of data points that support each other in their probabilistic assignments to one of the predicted classes.
2. An epsilon-insensitive loss function only penalizes small deviations from a model-predicted function.
3. An epsilon-insensitive only penalizes (linearly) large deviations, while assigning zero penalties to points within the ε-tube.
4. Unlike Neural Networks, an objective function in SVM is convex and hence has a unique minimum.

Correct answers: 3, 4

Supervised, Unsupervised and Reinforcement Learning in Finance

Week 1: Supervised Learning

Support Vector Machines part II: the math of SVMs

Igor Halperin

NYU Tandon School of Engineering, 2017

Karush-Kuhn-Tucker (KKT) conditions

Our general setting is of a **constrained optimization** of the form

$$\min\ f(x),\ \ x \in R^n$$

$$s.t.\ \ g(x) \geq 0$$

:

Karush-Kuhn-Tucker (KKT) conditions

Our general setting is of a **constrained optimization** of the form

$$\min\ f(x),\ \ x \in R^n$$

$$s.t.\ \ g(x) \geq 0$$

Use the method of **Lagrange multipliers** with the **KKT conditions**:

$$L(x,\lambda) = f(x) - \lambda g(x)$$

$$s.t.\ \ g(x) \geq 0, \quad \lambda \geq 0, \quad \lambda g(x) = 0$$

:

Karush-Kuhn-Tucker (KKT) conditions

Our general setting is of a **constrained optimization** of the form

$$\min\ f(x),\ \ x \in R^n$$

$$s.t.\ \ g(x) \geq 0$$

Use the method of **Lagrange multipliers** with the **KKT conditions**:

$$L(x,\lambda) = f(x) - \lambda g(x)$$

$$s.t.\ \ g(x) \geq 0, \quad \lambda \geq 0, \quad \lambda g(x) = 0$$

Two scenarios for an optimal point x^*:

1) **Inactive constraint**: $g(x) \geq 0, \quad \lambda = 0, \quad \lambda g(x) = 0$

2) **Active constraint**: $g(x) = 0, \quad \lambda > 0, \quad \lambda g(x) = 0$

When the constraint is inactive for an optimal solution, it means that the constraint plays no role, and the solution to the constrained problem is the same as for the unconstrained one.

Soln lies on the boundary bet areas in input space where g(x) changes from +ve to -ve

Active Constraint

Constrained optimization for SMV

Training of SVM Regression amounts to **constrained optimization**

$$\min_{w,b,\alpha,\eta} \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{N}\left(\xi_i + \xi_i^*\right) - \sum_{i=1}^{N}\left(\eta_i\xi_i + \eta_i^*\xi_i^*\right)$$

ensuring +ve ξ_i, ξ_i^* enforced by η_i, η_i^*

$$-\sum_{i=1}^{N}\alpha_i\left(\varepsilon + \xi_i - y_i + \langle w,x\rangle + b\right) - \sum_{i=1}^{N}\alpha_i^*\left(\varepsilon + \xi_i^* + y_i - \langle w,x\rangle - b\right)$$

$$s.t.\ \ \alpha_i,\ \alpha_i^* \geq 0, \quad \eta_i,\ \eta_i^* \geq 0$$

Primal Lagrange — analytical minimization wrt w, η

plugging them back into primal. the remainder is only dfn of the remaining Lagrange multipliers will be α_i, α_i^*. Dual Lagrange

Constrained optimization for SVM

Training of SVM Regression amounts to **constrained optimization**

$$\min_{w,b,\alpha,\eta} \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{N}\left(\xi_i + \xi_i^*\right) - \sum_{i=1}^{N}\left(\eta_i\xi_i + \eta_i^*\xi_i^*\right)$$

$$-\sum_{i=1}^{N}\alpha_i\left(\varepsilon + \xi_i - y_i + \langle w,x\rangle + b\right) - \sum_{i=1}^{N}\alpha_i^*\left(\varepsilon + \xi_i^* + y_i - \langle w,x\rangle - b\right)$$

$$s.t.\ \alpha_i,\ \alpha_i^* \geq 0, \quad \eta_i,\ \eta_i^* \geq 0$$

Dual optimization problem for Lagrange multipliers $\alpha_i,\ \alpha_i^* \geq 0$:

$$\min_{\alpha_i,\alpha_i^*} \frac{1}{2}\sum_{i,j=1}^{N}\left(\alpha_i - \alpha_i^*\right)\left(\alpha_j - \alpha_j^*\right)\langle x_i, x_j\rangle + \varepsilon\sum_{i=1}^{N}\left(\alpha_i + \alpha_i^*\right) - \sum_{i=1}^{N} y_i\left(\alpha_i - \alpha_i^*\right)$$

$$s.t.\ \sum_{i=1}^{N}\left(\alpha_i - \alpha_i^*\right) = 0, \quad 0 \leq \alpha_i,\ \alpha_i^* \leq C$$

the only numerical part of training SVM

Convex opt.

hence → unique soln

NN loss fn → non convex

Constrained optimization for SVM

Training of SVM Regression amounts to **constrained optimization**

$$\min_{w,b,\alpha,\eta} \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{N}\left(\xi_i + \xi_i^*\right) - \sum_{i=1}^{N}\left(\eta_i\xi_i + \eta_i^*\xi_i^*\right)$$

$$-\sum_{i=1}^{N}\alpha_i\left(\varepsilon + \xi_i - y_i + \left\langle w,x\right\rangle + b\right) - \sum_{i=1}^{N}\alpha_i^*\left(\varepsilon + \xi_i^* + y_i - \left\langle w,x\right\rangle - b\right)$$

$$s.t.\ \alpha_i,\ \alpha_i^* \geq 0, \quad \eta_i,\ \eta_i^* \geq 0$$

Dual optimization problem for Lagrange multipliers $\alpha_i,\ \alpha_i^* \geq 0$:

$$\min_{\alpha_i,\alpha_i^*} \frac{1}{2}\sum_{i,j=1}^{N}\left(\alpha_i - \alpha_i^*\right)\left(\alpha_j - \alpha_j^*\right)\left\langle x_i, x_j\right\rangle + \varepsilon\sum_{i=1}^{N}\left(\alpha_i + \alpha_i^*\right) - \sum_{i=1}^{N} y_i\left(\alpha_i - \alpha_i^*\right)$$

$$s.t.\ \sum_{i=1}^{N}\left(\alpha_i - \alpha_i^*\right) = 0, \quad 0 \leq \alpha_i,\ \alpha_i^* \leq C$$

The solution is (see A.Smola and B.Scholkopf, “A tutorial on Support Vector Regression” on how to find b using the KKT conditions):

$$f(x) = \sum_{i=1}^{N}\left(\alpha_i - \alpha_i^*\right)\left\langle x_i, x\right\rangle + b$$

Support vector expansion

Constrained optimization for SVM

Training of SVM Regression amounts to **constrained optimization**

$$\min_{w,b,\alpha,\eta} \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{N}\left(\xi_i + \xi_i^*\right) - \sum_{i=1}^{N}\left(\eta_i\xi_i + \eta_i^*\xi_i^*\right)$$

$$-\sum_{i=1}^{N}\alpha_i\left(\varepsilon + \xi_i - y_i + \left\langle w,x\right\rangle + b\right) - \sum_{i=1}^{N}\alpha_i^*\left(\varepsilon + \xi_i^* + y_i - \left\langle w,x\right\rangle - b\right)$$

$$s.t.\ \alpha_i,\ \alpha_i^* \geq 0, \quad \eta_i,\ \eta_i^* \geq 0$$

Dual optimization problem for Lagrange multipliers $\alpha_i,\ \alpha_i^* \geq 0$:

$$\min_{\alpha_i,\alpha_i^*} \frac{1}{2}\sum_{i,j=1}^{N}\left(\alpha_i - \alpha_i^*\right)\left(\alpha_j - \alpha_j^*\right)\left\langle x_i, x_j\right\rangle + \varepsilon\sum_{i=1}^{N}\left(\alpha_i + \alpha_i^*\right) - \sum_{i=1}^{N} y_i\left(\alpha_i - \alpha_i^*\right)$$

$$s.t.\ \sum_{i=1}^{N}\left(\alpha_i - \alpha_i^*\right) = 0, \quad 0 \leq \alpha_i,\ \alpha_i^* \leq C$$

The solution is (see A.Smola and B.Scholkopf, "A tutorial on Support Vector Regression" on how to find b using the KKT conditions):

$$f(x) = \sum_{i=1}^{N}\left(\alpha_i - \alpha_i^*\right)\left\langle x_i, x\right\rangle + b$$

Points within the ε-tube drop from the sum!

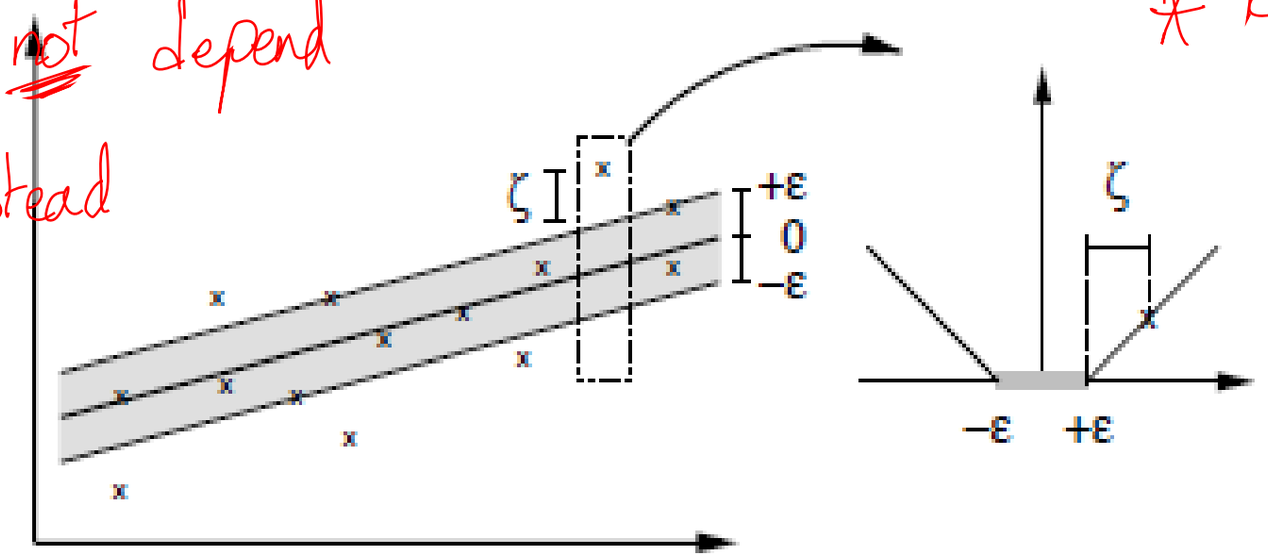
Support vector expansion

only points outside → 8 Support Vector

Support vector expansion for SVM

- Geometric interpretation: ε-insensitive loss function. Deviations exceeding ε are penalized in a linear fashion

* Final soln does not depend on X itself, instead on $\langle x_i, x_j \rangle$



* Dimensionality depends on SV dim, not dim of input space

- The solution is found in terms of dual variables α_i, α_i^* (Lagrange multipliers) conjugate to the constraints:

$$f(x) = \sum_i (\alpha_i - \alpha_i^*) \langle x_i, x \rangle + b$$

Inside the ε-tube, α_i, α_i^* vanish (KKT condition), so this is a sparse expansion in *support vectors* - hence the name SVM

Control question

Select all correct answers

1. When a constraint $g(x) \geq 0$ is active for an optimal solution x^*, it means for that solution, we have $g(x^*) \geq 0,\ \lambda = 0$
2. When a constraint $g(x) \geq 0$ is active for an optimal solution x^*, it means for that solution, we have $g(x^*) = 0,\ \lambda > 0$
3. Optimization in SVM amounts to a convex optimization in Lagrange multipliers of the original optimization problem.
4. As training of SVM amounts to convex optimization, it has the same problems with local minima as Neural Networks.
5. Support Vectors are data points outside of the ε-tube.
6. Coefficients of the Support Vector Expansion vanish inside of the ε-tube due to the KKT condition, therefore the Support Vector Expansion is sparse. The complexity of representation in SVM is determined by the number of Support Vectors, rather than by dimensionality of the inputs.

Correct answers: 2, 3, 5, 6.

Supervised, Unsupervised and Reinforcement Learning in Finance

Week 1: Supervised Learning

Support Vector Machines part III: the kernel trick

Igor Halperin

NYU Tandon School of Engineering, 2017

Constrained optimization for SVM

Solution for the SVM Regression:

$$f(x)=\sum_{i=1}^{N}\left(\alpha_i-\alpha_i^*\right)\left\langle x_i,x\right\rangle+b$$

The dual optimization problem for Lagrange multipliers $\alpha_i,\ \alpha_i^*\geq 0$:

$$\min_{\alpha_i,\alpha_i^*}\ \frac{1}{2}\sum_{i,j=1}^{N}\left(\alpha_i-\alpha_i^*\right)\left(\alpha_j-\alpha_j^*\right)\left\langle x_i,x_j\right\rangle+\varepsilon\sum_{i=1}^{N}\left(\alpha_i+\alpha_i^*\right)-\sum_{i=1}^{N}y_i\left(\alpha_i-\alpha_i^*\right)$$

$$s.t.\ \sum_{i=1}^{N}\left(\alpha_i-\alpha_i^*\right)=0,\quad 0\leq\alpha_i,\ \alpha_i^*\leq C$$

The solution depends only on a dot product of x rather than on x itself!

Constrained optimization for SVM

Solution for the SVM Regression:

$$f(x)=\sum_{i=1}^{N}\left(\alpha_i-\alpha_i^*\right)\left\langle x_i,x\right\rangle+b$$

The dual optimization problem for Lagrange multipliers $\alpha_i,\ \alpha_i^*\geq 0$:

$$\min_{\alpha_i,\alpha_i^*}\ \frac{1}{2}\sum_{i,j=1}^{N}\left(\alpha_i-\alpha_i^*\right)\left(\alpha_j-\alpha_j^*\right)\left\langle x_i,x_j\right\rangle+\varepsilon\sum_{i=1}^{N}\left(\alpha_i+\alpha_i^*\right)-\sum_{i=1}^{N}y_i\left(\alpha_i-\alpha_i^*\right)$$

$$s.t.\ \sum_{i=1}^{N}\left(\alpha_i-\alpha_i^*\right)=0,\quad 0\leq\alpha_i,\ \alpha_i^*\leq C$$

The solution depends only on a dot product of x rather than on x itself!

If we used features $\Phi(x)$ instead of raw inputs, the solution is obtained in the same way:

$$f(x)=\sum_{i=1}^{N}\left(\alpha_i-\alpha_i^*\right)\left\langle\Phi(x_i),\Phi(x)\right\rangle+b$$

SVM: introducing non-linearity by preprocessing

How to make features that handles non-linearities by converting a non-linear problem into a linear problem in a feature space

Feature Engineering

Example: we want to capture quadratic features for points on a plane $x \in R^2$

$\Phi(x) = \left(x_1^2, \sqrt{2} x_1 x_2, x_2^2 \right)$ can be viewed as a vector in R^3. The key observation is that (check it!)

$$\left\langle \Phi(x), \Phi(x') \right\rangle = \left\langle \left(x_1^2, \sqrt{2} x_1 x_2, x_2^2 \right), \left(x'^2_1, \sqrt{2} x'_1 x'_2, x'^2_2 \right) \right\rangle = \left\langle x, x' \right\rangle^2$$

Idea

This means that by using features constructed from raw inputs, we can capture quadratic and higher order effects in the SVM, as long as all results depend only on dot products

Questions:

1. How to construct features?
2. Can we produce MANY features, or even infinite number of features? Can it help?

SVM: the kernel trick

If we do linear SVM in a feature space (e.g. R^3, in our example), the results is

$$f(x) = \sum_{i=1}^{N} \left(\alpha_i - \alpha_i^*\right) \left\langle \Phi(x_i), \Phi(x) \right\rangle + b$$

$$= \sum_{i=1}^{N} \left(\alpha_i - \alpha_i^*\right) k(x_i, x) + b$$

fn of both x_i, x rather a data point

Kernel

The kernel trick: replace the feature engineering by a kernel engineering, i.e. model a kernel instead of modeling features!

Why may be a good idea:

1. A kernel should be interpretable as a dot product in some high-dimensional (or even infinite-dimensional) feature space
2. A number of theorems provide conditions for admissible kernels (Mercer's theorem, etc., see Smola and Scholkopf, or Geron)
3. A number of popular kernels is available in both the literature and software

Examples of kernels for SMV

1. A linear kernel $$k(x,x') = \langle x, x' \rangle$$

Does not handle non-linearities, but is faster to work with than using non-linear kernels

2. Polynomial kernel $$k(x,x') = \left(\gamma \langle x, x' \rangle + r\right)^d$$

3. Sigmoid kernel $$k(x,x') = \tanh\left(\gamma \langle x, x' \rangle + r\right)$$

4. Gaussian Radial Basis Function (RBF) kernel

$$k(x,x') = \exp\left(-\gamma \| x - x' \|^2\right)$$

Features corresponding to a Gaussian RBF kernel are actually infinite-dimensional!

Kernel parameters such as γ are model hyper-parameters (alongside C, ε). Can be tuned using a validation dataset or cross-validation. A version called ν-SVR tunes ε

not hand-picked but rather learnt from data

More advanced methods: kernel learning

Control question

Select all correct answers

1. The kernel trick used by the SVM is based on the observation that the final result depends only on dot products of inputs x rather than directly on their values.

2. Instead of constructing features $\Phi(x)$, we can directly model kernels $k(x,x')$.

3.Kernels $k(x,x') = \langle x,x' \rangle$, $k(x,x') = \left(\langle x,x' \rangle + r\right)^d$ are both valid kernels.

4.A Gaussian RBF kernel is defined as $\exp\left(-\gamma k^2(x,x')\right)$

5.A Gaussian RBF kernel is defined as $\exp\left(-\gamma k^2 \left\|x - x'\right\|^2\right)$

Correct answers: 1,2, 3, 5

Supervised, Unsupervised and Reinforcement Learning in Finance

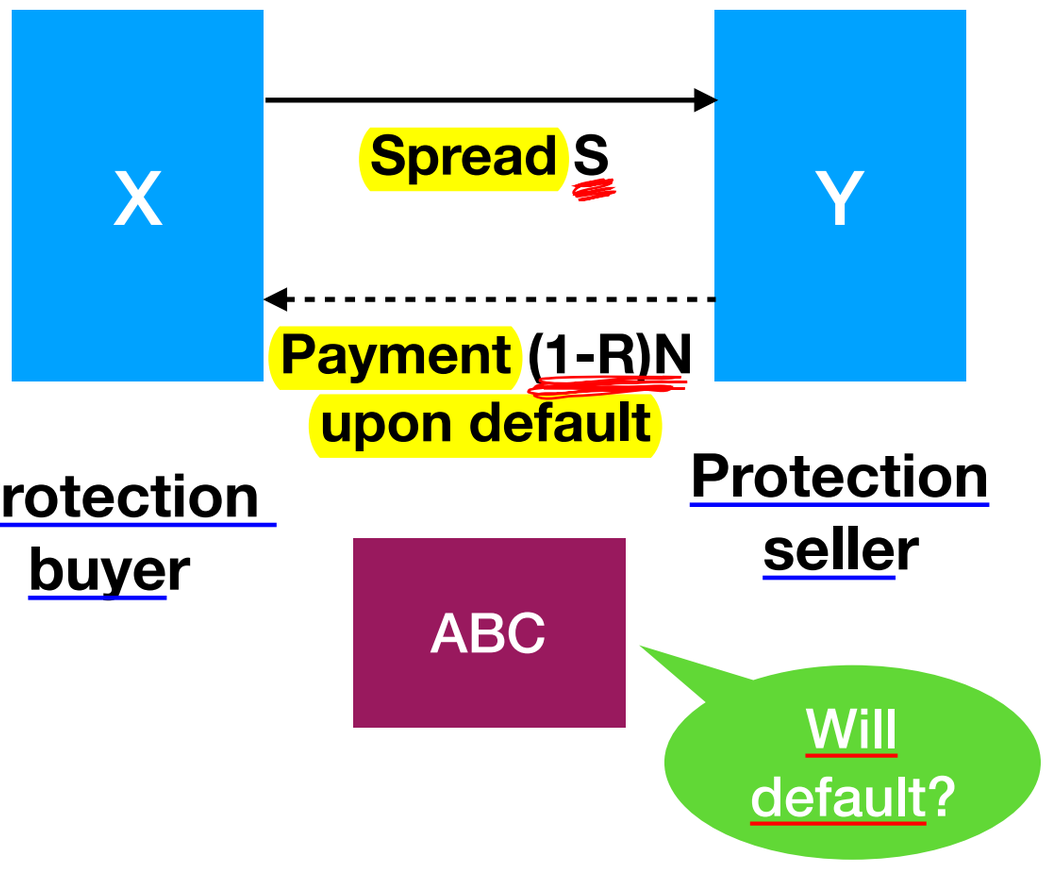
Week 1: Supervised Learning

Support Vector Machines part IV: Example: SVM for prediction of credit spreads

Igor Halperin

NYU Tandon School of Engineering, 2017

Credit Default Swap (CDS)

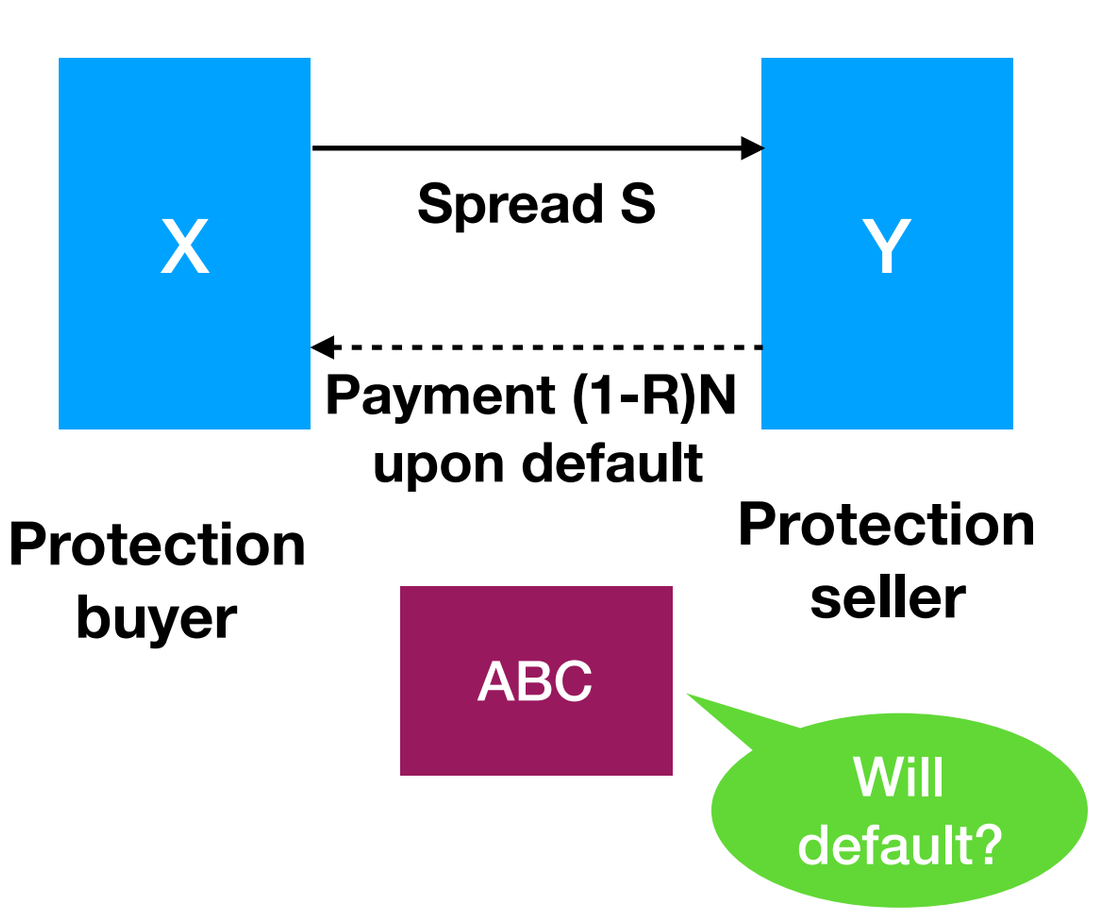


Modelling credit spreads for illiquid names → those companies that issue debt or borrow credit while not having activated bonds or there are no activated CDS referenced in such firms

Spread S is paid on the notional amount N of $10m

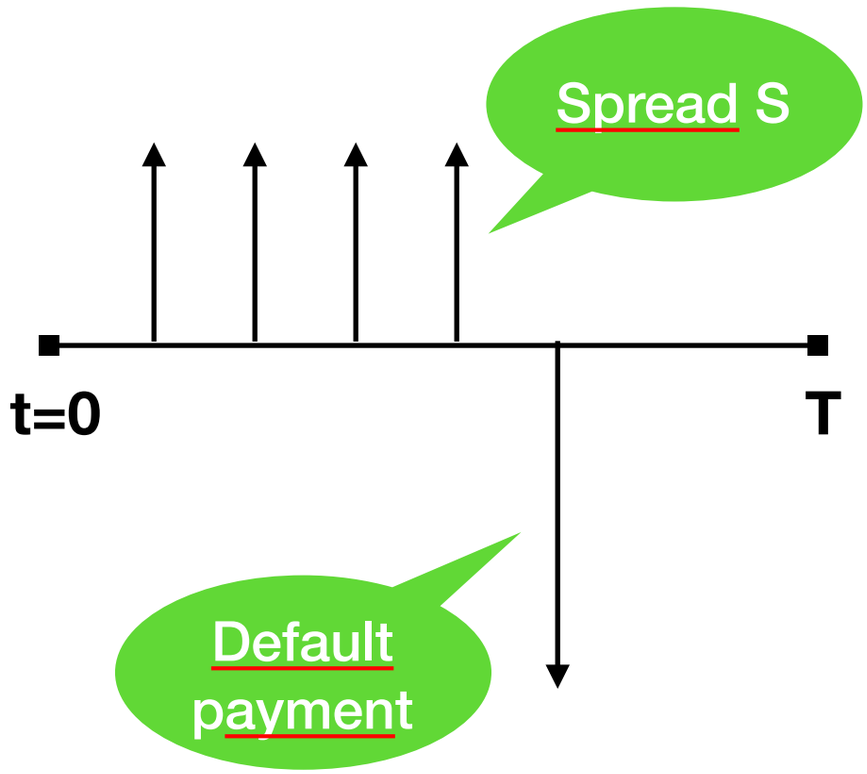
R: recovery rate

Credit Default Swap (CDS)



Spread S is paid on the notional amount of $10m

Cashflows for protection buyer X



↑ prob of default

↑ spread

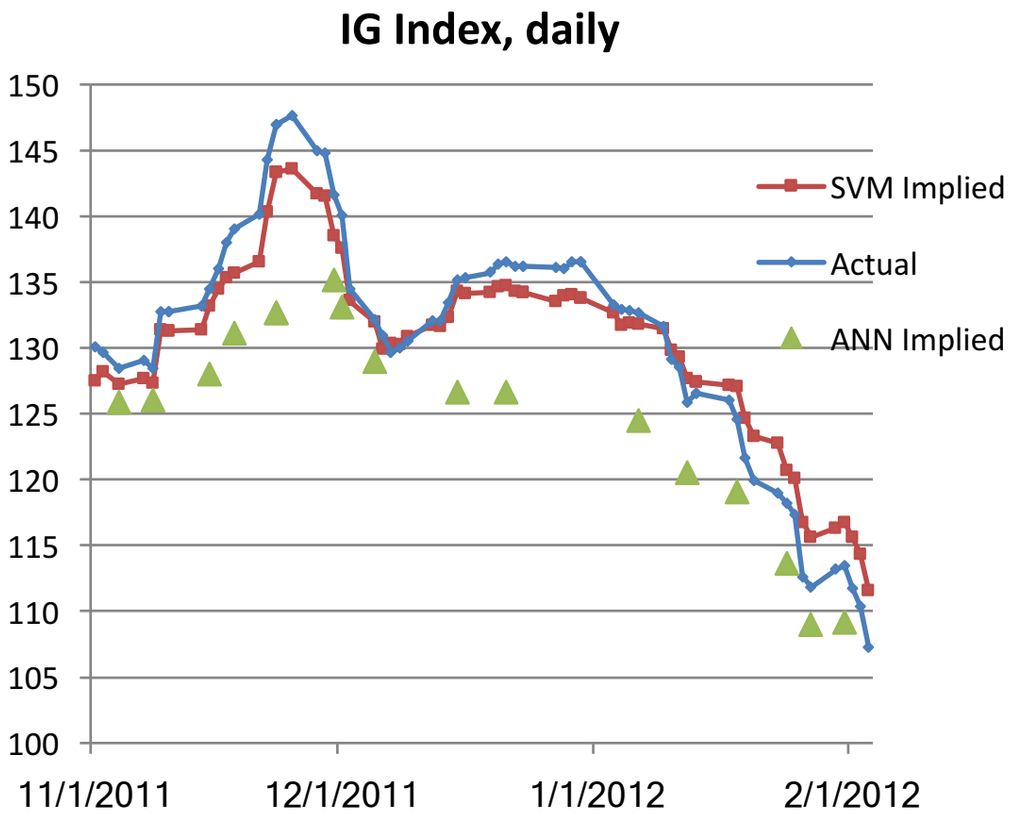
Example: Illiquid CDS spread modeling

- **Task:** Estimate unobserved CDS spreads for companies with illiquid debt based on observed spreads and other characteristics of other, liquid companies.
- **Inputs:** $X_i = (X_i^1, \ldots, X_i^D)$ - company's "features": rating, industry sector, geographic sector, Expected Default Frequency (EDF - supplied by Moody's)
- **Additional optional inputs**: financial indicators, implied volatilities, CreditGrades, etc.
- **Outputs**: Y_i - CDS spread for company i.
- **Training sample**: $\{(X_i, Y_i)\}_{i=1}^N$ for observed universe of liquid CDS (~500 companies in the US market)
- **Usage**: Counterparty credit risk management, capital calculation etc.
- **Models used:** Neural Network (NN) or Support Vector Machine (SVM)

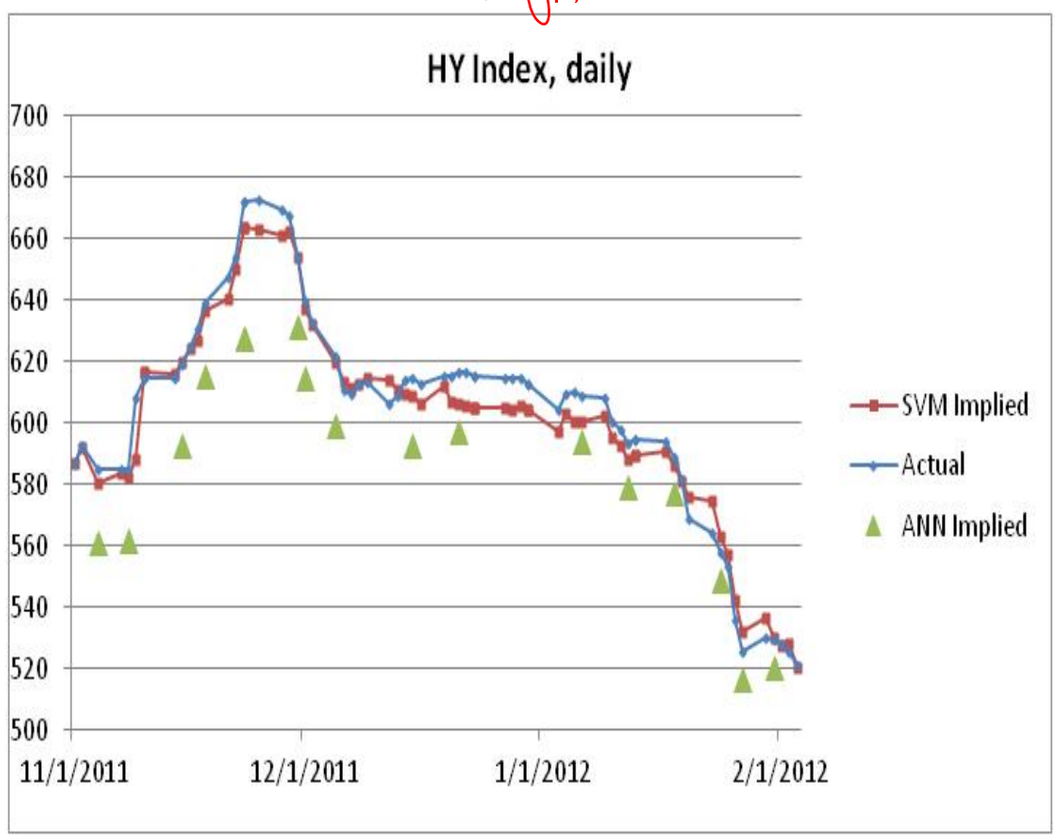
Illustration of performance of SVM for spread prediction

As one of the metrics for tracking model performance, one can compute time series for a theoretical index spread mimicking (in terms of generic names for given sector/rating) CDX or iTraxx indices, and compare them with the actual time series of the index

Index Rate



High Yield



Control question

Select all correct answers

1. A Credit Default Swap (CDS) is an insurance-like contract between a protection buyer (X) and protection seller (Y), whose payoff is contingent on a default event by a reference entity ABC.
2. As upon default by reference entity, protection buyer (X) benefits from a payment from a CDS contract, it shares its revenues with ABC, so the latter is partially compensated for its default.
3. A one-time payment to the protection buyer X by the protection seller Y happens only in the case of default by the reference entity ABC.

Correct answers: 1, 3

Supervised, Unsupervised and Reinforcement Learning in Finance

Week 1: Supervised Learning

Tree methods: CART Trees

Igor Halperin

NYU Tandon School of Engineering, 2017

CART for bank analysis

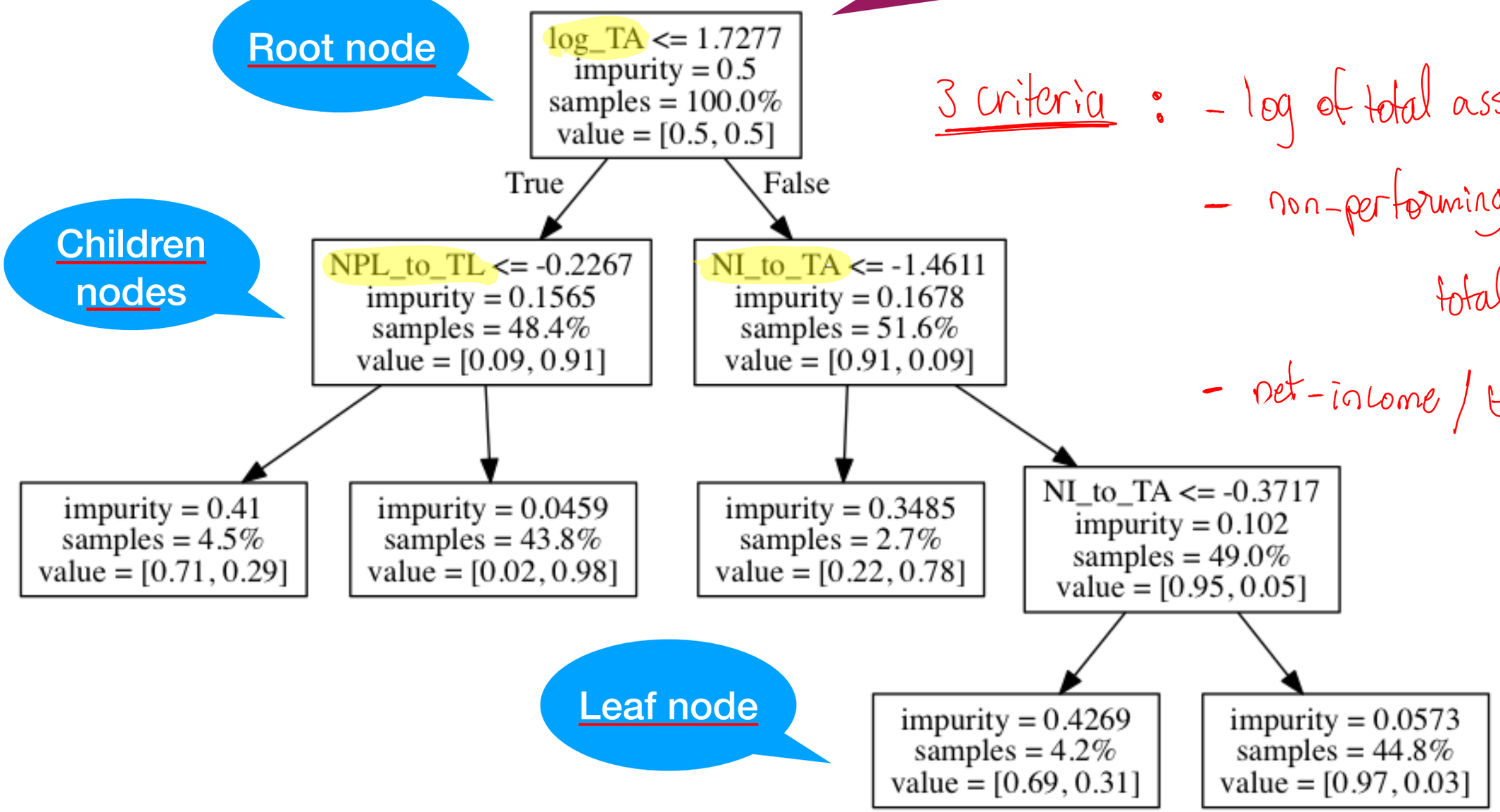
CART = Classification and Regression Tree

Recursive partitioning of the input space

Root node

Children nodes

Leaf node



3 criteria : - log of total assets

- non-performing loans / total loans

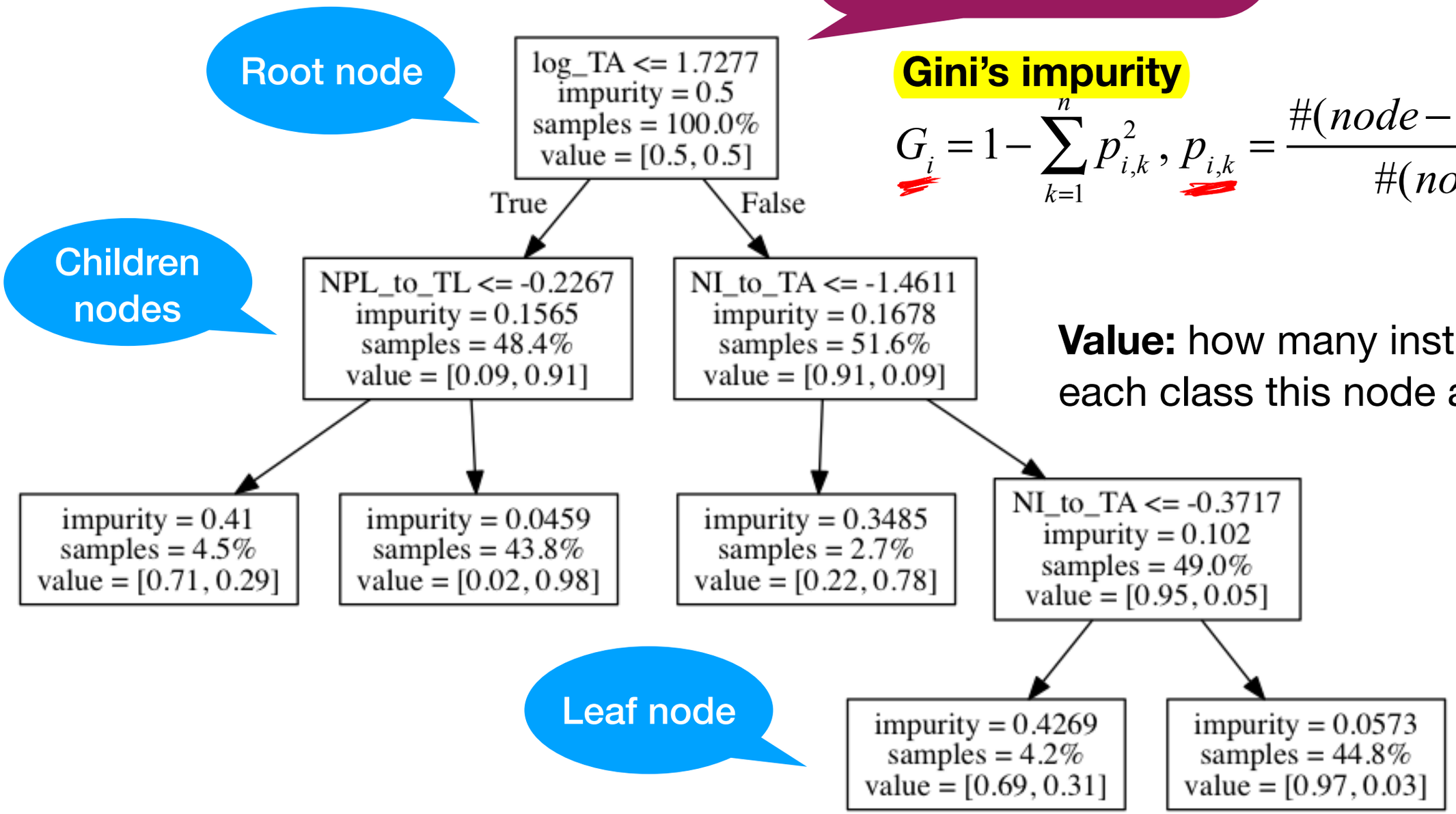
- net-income / total assets

CART for bank analysis

CART = Classification and Regression Tree

Recursive partitioning of the input space

Root node



Children nodes

Leaf node

Gini's impurity

$$G_i = 1 - \sum_{k=1}^{n} p_{i,k}^2 \,,\; p_{i,k} = \frac{\#(node - i, class - k)}{\#(node - i)}$$

Value: how many instances of each class this node applies to

The CART training algorithm

Greedy algorithm to grow a tree starting from a root note:
The cost function

$$J(k,t_k) = \frac{m_{left}}{m} G_{left} + \frac{m_{right}}{m} G_{right}$$

$$where\ G_{left,right} = impurity\ of\ the\ left\ /\ right\ subset$$

At each step, choose feature k and threshold t_k to minimize the cost function

Can use other measures instead of Gini impurities $G_{left,right}$:

Entropy: $H_i = -\sum_{k=1}^{n} p_{i,k} \log p_{i,k}$ (Entropy is zero if there is only one class in a node, so behaves similar to Gini)

Entropy and Gini often, but not always, produce similar trees.

Complexity control for trees

Trees can easily overfit, need regularization:

- Constrain the maximum depth of the tree (max_depth)
- Min_samples_split (minimal number of samples in a node to be considered for a split)
- Min_samples_leaf (min number of samples to be in each leaf node)
- Min_impurity_split (a node will split if its impurity is above the threshold, otherwise it is a leaf)

2. These hyper-parameters can be tuned using a validation set, or by cross-validation.

Trees: pros and cons

Pros:

- Simple to interpret
- Require almost no pre-processing
- Can handle missing data
- Insensitive to monotone transformations of inputs
- Perform automatic variable selection → feature picked at root is the most important
- Scale up well to large datasets

Cons:

- Lower accuracy than other model (due to the greedy tree construction)
- Potential instability under small variation of input data (the same origin)
  (Trees are high variance estimators)

Control question

Select all correct answers

1. Gini impurity is defined as $G_i = 1 - \sum_{n}^{} p_{i,k}^2$
   where $p_{i,k}$ is a fraction of instances of class k among all instances at node i.

2. Gini impurity is defined as $G_i = -\sum_{n}^{} p_{i,k} \log p_{i,k}$

3. Hyper-parameters of a CART tree are optimized by minimizing the CART cost function on the training set.

Incorrect !!

4. Trees are simple to interpret, and they perform an automatic feature selection

5. Trees are typically high variance estimations

Correct answers: 1, 4, 5.

Supervised, Unsupervised and Reinforcement Learning in Finance

Week 1: Supervised Learning

Tree methods: Random Forests

Igor Halperin

NYU Tandon School of Engineering, 2017

Ensemble methods for trees

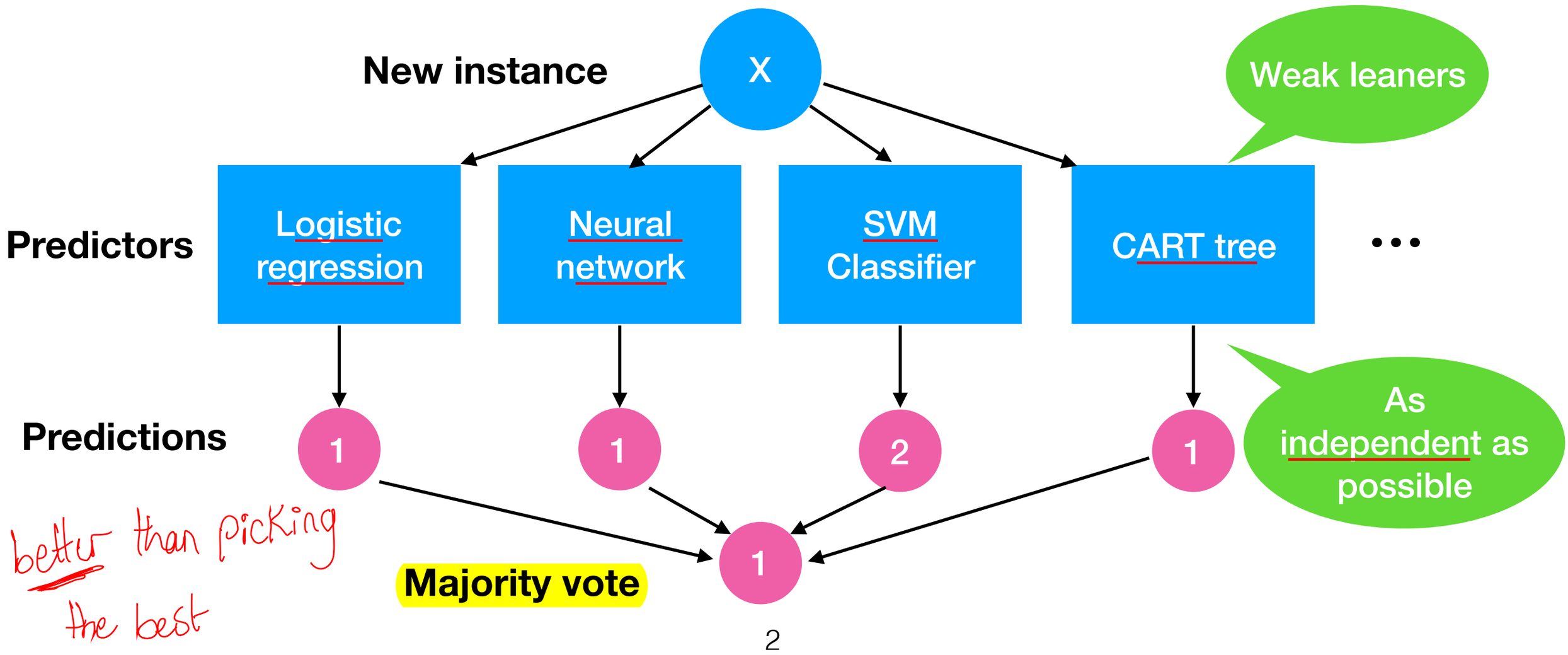
We can combine (aggregate) predictions made by many classifiers, each trained on the same input data.

Wisdom of the crowd

This is called **Ensemble Learning.**

How it can work: train many classifiers, then use a majority vote to define the final class for an instance.

weak learners



Ensemble learning: a coin example

Assume we have a biased coin with the head probability of 51%.

Toss the coin $N = 1000$ times.

1. Each trial produces a **weak learner** with the probability of $p_H = 0.51$ to get the right answer.

2. Construct a **strong learner** by taking a majority vote among weak learner

$$P\left(N_H > N_T\right) = P\left(N_H > \frac{N}{2}\right) = 1 - P\left(N_H \le \frac{N}{2}\right) \simeq 73\%$$

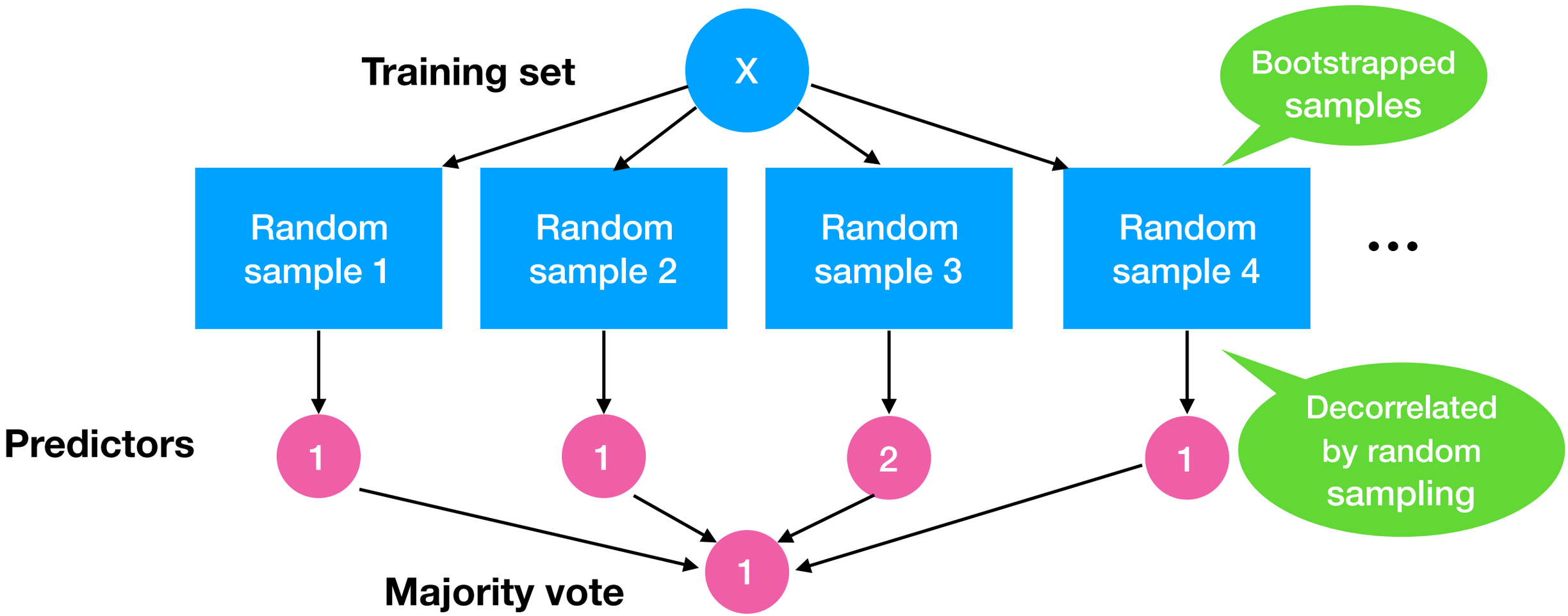
3. The strong learner uses the **law of large numbers** to make a prediction that is better than the best predictor in its set of weak learners.

Bagging

Bagging = bootstrap aggregation (Breiman, 1996)

Bagging = Ensemble Learning with weak learners obtained from the same algorithm

How it works: train many trees on random sub-sets of the input data, then use a majority vote to define the final class for an instance.



Random Forests

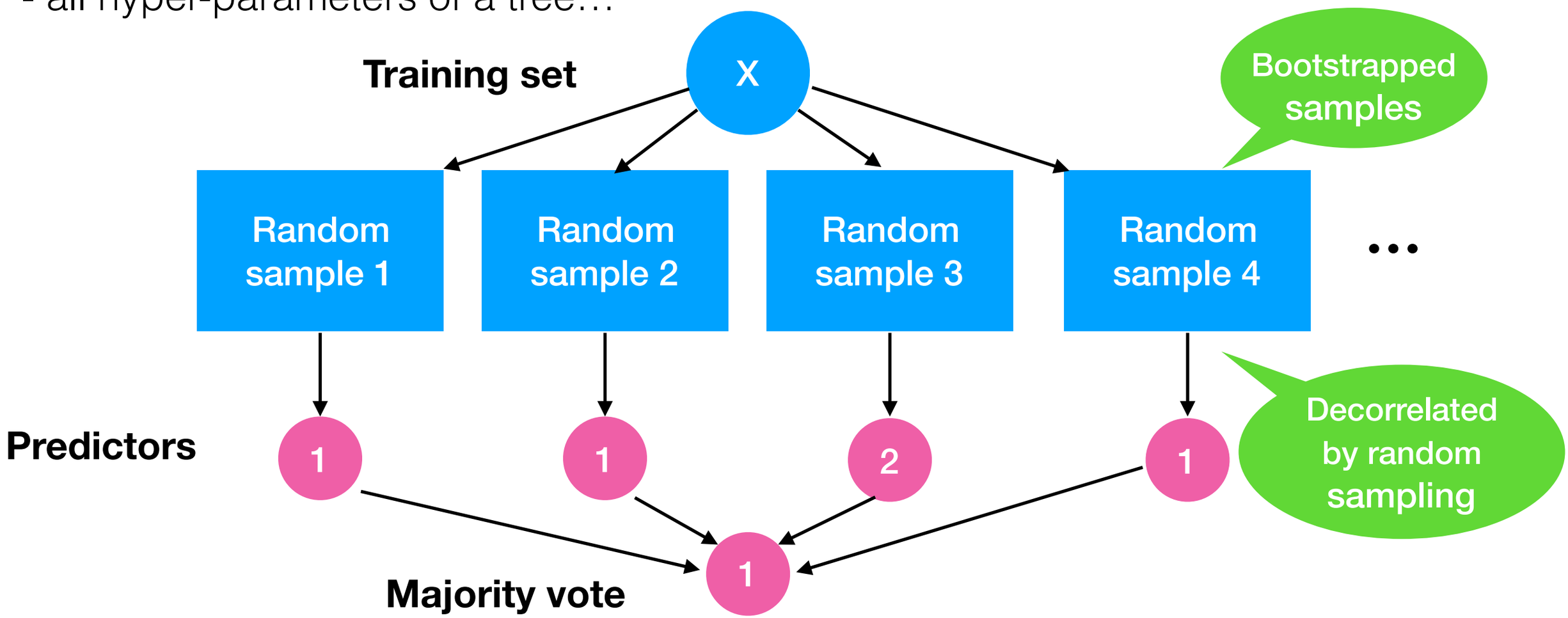
Random Forest (Breiman, 2001): de-correlate base (weak) learners by randomly sampling features and subsets of a train dataset, use bagging for predictions.

- Produces lower variance estimators than a tree

good generalization

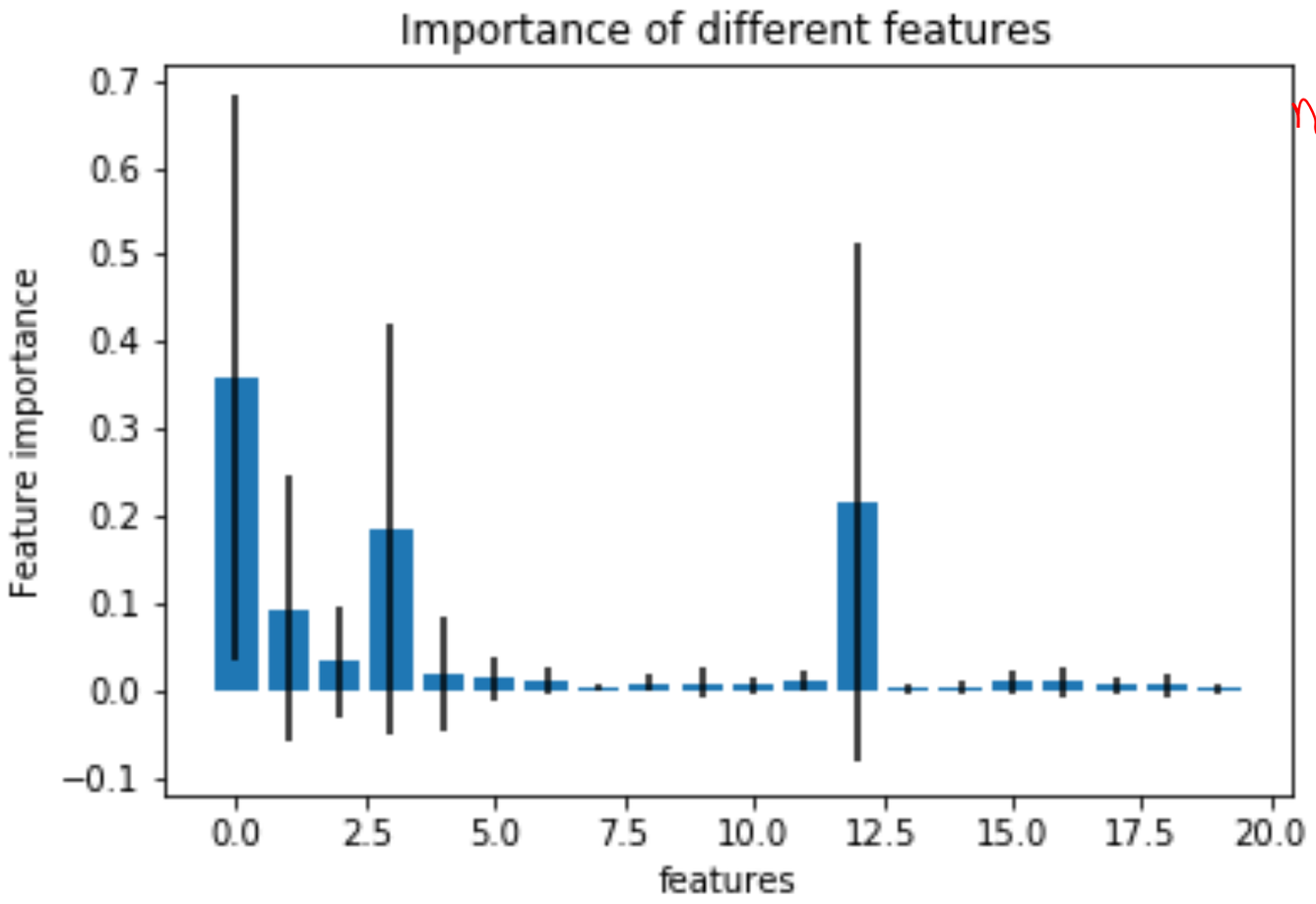
Hyper-Parameters of a Random Forest:

- num_estimators
- all hyper-parameters of a tree…



Feature importance in Random Forest

Important of feature is given by an average depth at which at appears across all trees in a Random Forest, plus a standard deviation of this depth:



most important →

	Importance	Std
log_TA	0.350043	0.329071
NI_to_TA	0.094549	0.156222
Equity_to_TA	0.030421	0.035744
NPL_to_TL	0.133095	0.192915
REO_to_TA	0.025687	0.077842
ALLL_to_TL	0.011211	0.034182
core_deposits_to_TA	0.010999	0.013458
brokered_deposits_to_TA	0.001025	0.003567
liquid_assets_to_TA	0.009313	0.011609
loss_provision_to_TL	0.011886	0.021393
NIM	0.011005	0.013408
assets_growth	0.003713	0.006769
FDIC_assessment_base_n	0.271504	0.320079

Control question

Select all correct answers

1. Weak learners in ensemble methods should be as independent of each other as possible.

2. A strong learner in ensemble learning is the best learner among weak learners.

3. Performance of the strong performer is better than performance of the best weak learning, as ensemble learning relies on the law of large numbers to reduce variance of the final estimator.

4. Random forest algorithm creates de-correlated weak learning by random sampling of subsets of features and subsets of data points.

5. As it is based on randomization, a Random Forest estimator typically has higher variance than a single tree estimator.

Correct answers: 1, 3, 4.

Supervised, Unsupervised and Reinforcement Learning in Finance

Week 1: Supervised Learning

Tree methods: Boosting

Igor Halperin

NYU Tandon School of Engineering, 2017

What is boosting

more powerful than Random Forests

Boosting is any Ensemble method that produces a strong learner out of week learners

Two elements:

- how to pick weak learners (typically shallow CART trees)
- how to construct a strong learners

Two most popular boosting approaches:

- **AdaBoost** (Adaptive Boosting)
- **Gradient Boosting**

Hyper-parameters of Boosting algorithms:

- number of base classifiers
- hyper-parameters of base classifiers
- learning rate

Boosting as gradient descent

Boosting = gradient descent in function space (Breiman 1998)

Boosting solves the following optimization problem:

$$\min_{f} \sum_{i=1}^{N} L\left(y_i, f(x_i)\right) \quad \Leftarrow \quad f(x) = w_0 + \sum_{m=1}^{M} w_m \phi_m(x, \gamma)$$

Examples of Loss functions:

Regression: the squared loss $L\left(y_i, f(x_i)\right) = \frac{1}{2}\left(y_i - f(x_i)\right)^2$ ==> L2Boosting

Regression: L1-loss $L\left(y_i, f(x_i)\right) = \left|y_i - f(x_i)\right|$ ==> Gradient Boosting

Classification: Exponential loss $L\left(y_i, f(x_i)\right) = \exp\left(-y_i f(x_i)\right)$ ==> AdaBoost

* Our optimization too general, and not practical since it is an infinite-dim problem. As a workaround we shall restrict $f(x)$ to be an expansion in some set of features or basis fn ϕ_m

Boosting as gradient descent

Boosting = gradient descent in function space (Breiman 1998)
Boosting solves the following optimization problem:

$$\min_f \sum_{i=1}^{N} L\left(y_i, f(x_i)\right) \quad \Leftarrow \quad f(x) = w_0 + \sum_{m=1}^{M} w_m \phi_m(x, \gamma)$$

Solve iteratively:

- initialize $f_0(x) = \arg\min_w \sum_{i=1}^{N} L\left(y_i, f(x_i, \gamma)\right)$

(e.g. for a squared error $L\left(y_i, f(x_i)\right) = \frac{1}{2}\left(y_i - f(x_i)\right)^2$, we set $f_0(x) = \bar{y}$)

mean of outputs

Boosting as gradient descent

Boosting = gradient descent in function space (Breiman 1998)
Boosting solves the following optimization problem:

$$\min_f \sum_{i=1}^{N} L\left(y_i, f(x_i)\right) \quad \Leftarrow \quad f(x) = w_0 + \sum_{m=1}^{M} w_m \phi_m(x, \gamma)$$

Solve iteratively:

- initialize $f_0(x) = \arg\min_w \sum_{i=1}^{N} L\left(y_i, f(x_i, \gamma)\right)$

(e.g. for a squared error $L\left(y_i, f(x_i)\right) = \frac{1}{2}\left(y_i - f(x_i)\right)^2$, we set $f_0(x) = \bar{y}$)

- At iteration m, compute:

$$\left(\beta_m, \gamma_m\right) = \arg\min_{\beta,\gamma} \sum_{i=1}^{N} L\left(y_i, f_{m-1}(x_i) + \beta\phi(x_i, \gamma)\right)$$

$$f_m(x) = f_{m-1}(x) + \beta_m \phi(x, \gamma_m)$$ updating

This is called forward stage-wise additive modeling (no going back and updating earlier parameters). For more on boosting methods, see. Chap. 16.4 in Murphy)

Example: L2Boosting as gradient descent

Boosting = gradient descent in function space (Breiman 1998)
Boosting solves the following optimization problem:

$$\min_f \sum_{i=1}^{N} L\left(y_i, f(x_i)\right) \quad \leftarrow \quad f(x) = w_0 + \sum_{m=1}^{M} w_m \phi_m(x, \gamma)$$

Solve iteratively for the squared error $L\left(y_i, f(x_i)\right) = \frac{1}{2}\left(y_i - f(x_i)\right)^2$:

- initialize $f_0(x) = \bar{y}$

- At iteration m, compute:

$$\left(\beta_m, \gamma_m\right) = \arg\min_{\beta,\gamma} \sum_{i=1}^{N} \left(y_i - f_{m-1}(x_i) - \beta\phi(x_i, \gamma)\right)^2$$

$$f_m(x) = f_{m-1}(x) + \beta_m \phi(x, \gamma_m)$$

Each new basis function is optimized to fit the current residual $r_{im} = y_i - f_{m-1}(x_i)$
This is called L2Boosting, or Least Squares Boosting (Buhlmann and Yu, 2003)

Control question

Select all correct answers

1. Boosting amounts to inflating the weight of a best weak learner among all weak learners.
2. If your boosted tree overfit, you should increase the number of weak learners, which adds more noise to the problem, and hence reduces the generalization error.
3. Boosting methods typically use shallow CART trees as weak learners.
4. Boosting can be understood as optimization in a functional space. Depending on the specification of loss function, such procedure gives rise to algorithms such as L2Boosting, AdaBoost, or Gradient Boosting.

Correct answers: 3,4.