Machine Learning for HFT price movement predictions

James Han, Chun Yu Hong, Nick Sutardja, Sio Fong Wong

Table of Contents

Objectives:

- To use machine learning algorithms to predict price movements based on limit order book data

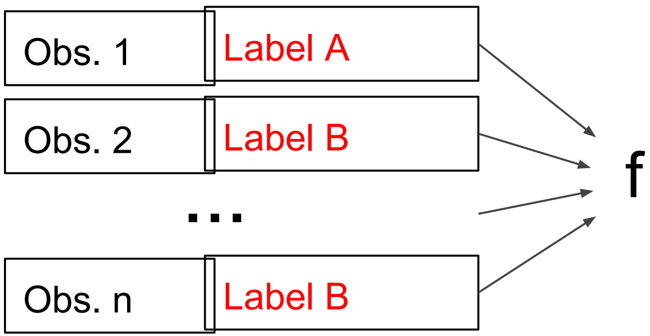
  Ex: Support Vector Machines, Random Forests, etc.

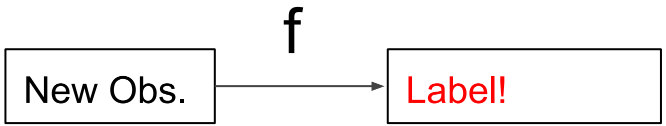
Supervised Learning

-Goal: to infer a function based on labeled data

-Classification: labels are discrete (ex: whether an email is spam or not)

-Regression: labels are continuous (ex: income)

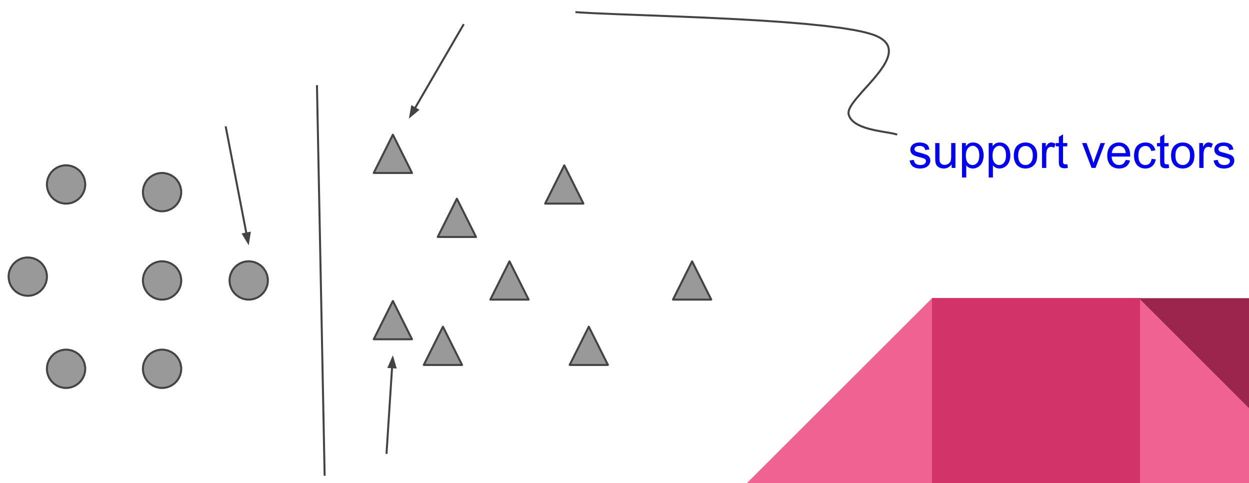




Support Vector Machines (SVMs)

-Basic idea: Find the maximum-margin separating hyperplane that divides the data points into two classes.

-Key property: Only the hard-to-classify data points matter when determining the separating hyperplane.



Support Vector Machines (SVMs)

-Can be formulated as a convex optimization problem:

$$\begin{array}{ll} \min_{w,b} & \frac{1}{2}||w||^2 \\ \text{subject to} & y^{(i)}(w^T z^{(i)} + b) \geq 1, \; i = 1, \ldots, m. \end{array}$$

-Problem: This formulation assumes that the data points are linearly separable!

Support Vector Machines (SVMs)

-Remedy: Introduce **slack** variables.

$$\begin{aligned} \min_{w,b} \quad & \frac{1}{2}||w||^2 + C\sum_{i=1}^{m}\xi^{(i)} \\ \text{subject to} \quad & y^{(i)}(w^T z^{(i)} + b) \geq 1 - \xi^{(i)}, \; i = 1, \ldots, m. \\ & \xi^{(i)} \geq 0, \; i = 1, \ldots, m. \end{aligned}$$

-large C: get as many correctly labeled data points as possible at the expense of having a small margin

-small C: get a large margin at the expense of (potentially) having more incorrectly classified labels

Support Vector Machines (SVMs)

-What if we want to deal with nonlinear decision boundaries? Transformation.

$$\begin{aligned} \min_{w,b} \quad & \frac{1}{2}||w||^2 + C\sum_{i=1}^{m}\xi^{(i)} \\ \text{subject to} \quad & y^{(i)}(w^T z^{(i)} + b) \geq 1 - \xi^{(i)},\ i = 1, \ldots, m. \\ & \xi^{(i)} \geq 0,\ i = 1, \ldots, m. \end{aligned} \longrightarrow \begin{aligned} \min_{w,b} \quad & \frac{1}{2}\langle w, w\rangle + C\sum_{i=1}^{m}\xi^{(i)} \\ \text{subject to} \quad & y^{(i)}(\langle w, \Phi(z^{(i)})\rangle + b) \geq 1 - \xi^{(i)},\ i = 1, \ldots, m. \\ & \xi^{(i)} \geq 0,\ i = 1, \ldots, m. \end{aligned}$$

feature map

Support Vector Machines (SVMs)

-Dual Problem (Karush-Kuhn-Tucker conditions):

$$\begin{aligned} \max_{\alpha} \quad & \sum_{i=1}^{m} \alpha^{(i)} - \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m} \alpha^{(i)}\alpha^{(j)}y^{(i)}y^{(j)}\langle \Phi(z^{(i)}), \Phi(z^{(j)})\rangle \\ \text{subject to} \quad & \sum_{i=1}^{m} \alpha^{(i)}y^{(i)} = 0 \\ & 0 \leq \alpha^{(i)} \leq C, \; i = 1, \ldots, m. \end{aligned}$$

-Key: The optimization problem depends only on $\mathcal{K}(\cdot,\cdot) \triangleq \langle \Phi(\cdot), \Phi(\cdot) \rangle$

kernel / similarity measure

Support Vector Machines (SVMs)

-Commonly used kernels:

Linear kernel: $K(\mathbf{x}, \mathbf{x}') = \mathbf{x}^T \mathbf{x}'$

Polynomial kernel: $K(\mathbf{x}, \mathbf{x}') = (c + \mathbf{x}^T \mathbf{x}')^d$

Gaussian kernel: $K(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{||\mathbf{x} - \mathbf{x}'||^2}{2\sigma^2}\right)$

Gaussian kernel: $K(\mathbf{x}, \mathbf{x}') = \exp(-\gamma ||\mathbf{x} - \mathbf{x}'||^2)$

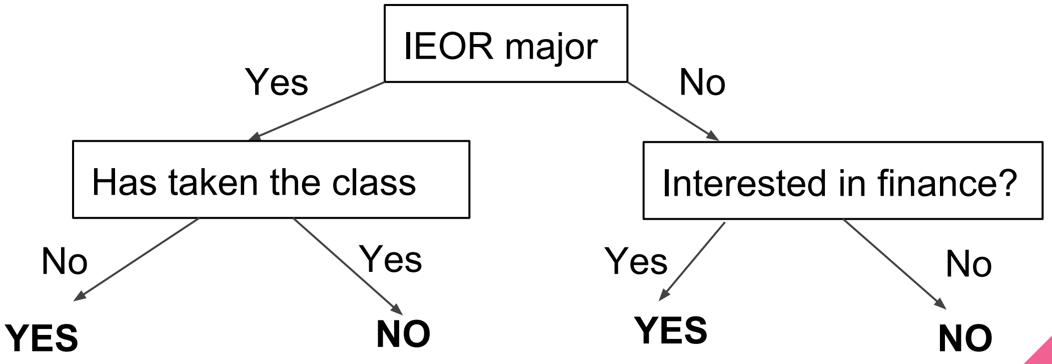
Support Vector Machines (SVMs)

Demo...

Random Forests

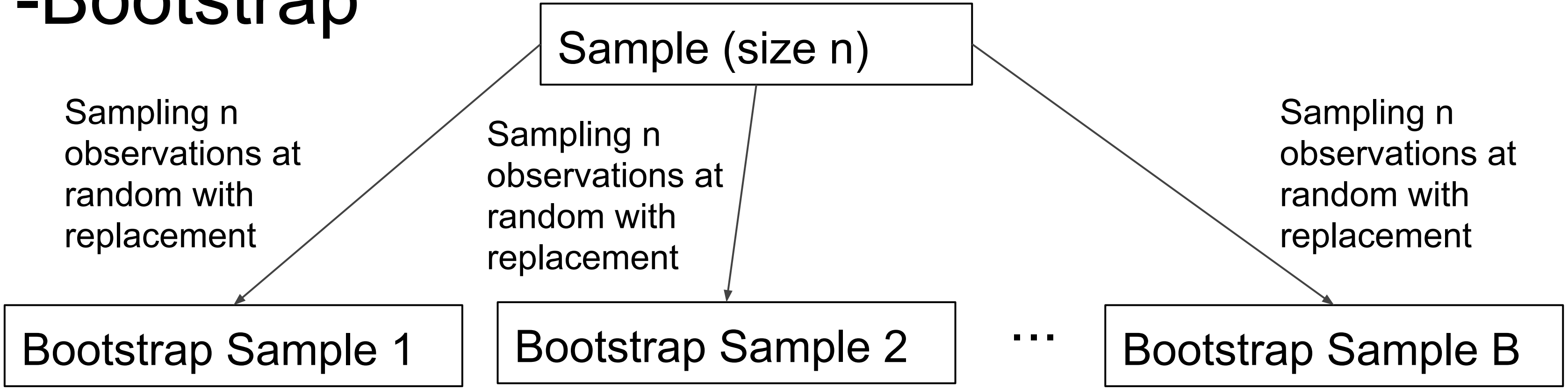
-Three ideas put together: Decision Trees + Random selection of features + Random selection of data (Bootstrapping)

-Example of a decision tree: Predict whether a student is currently taking IEOR 222

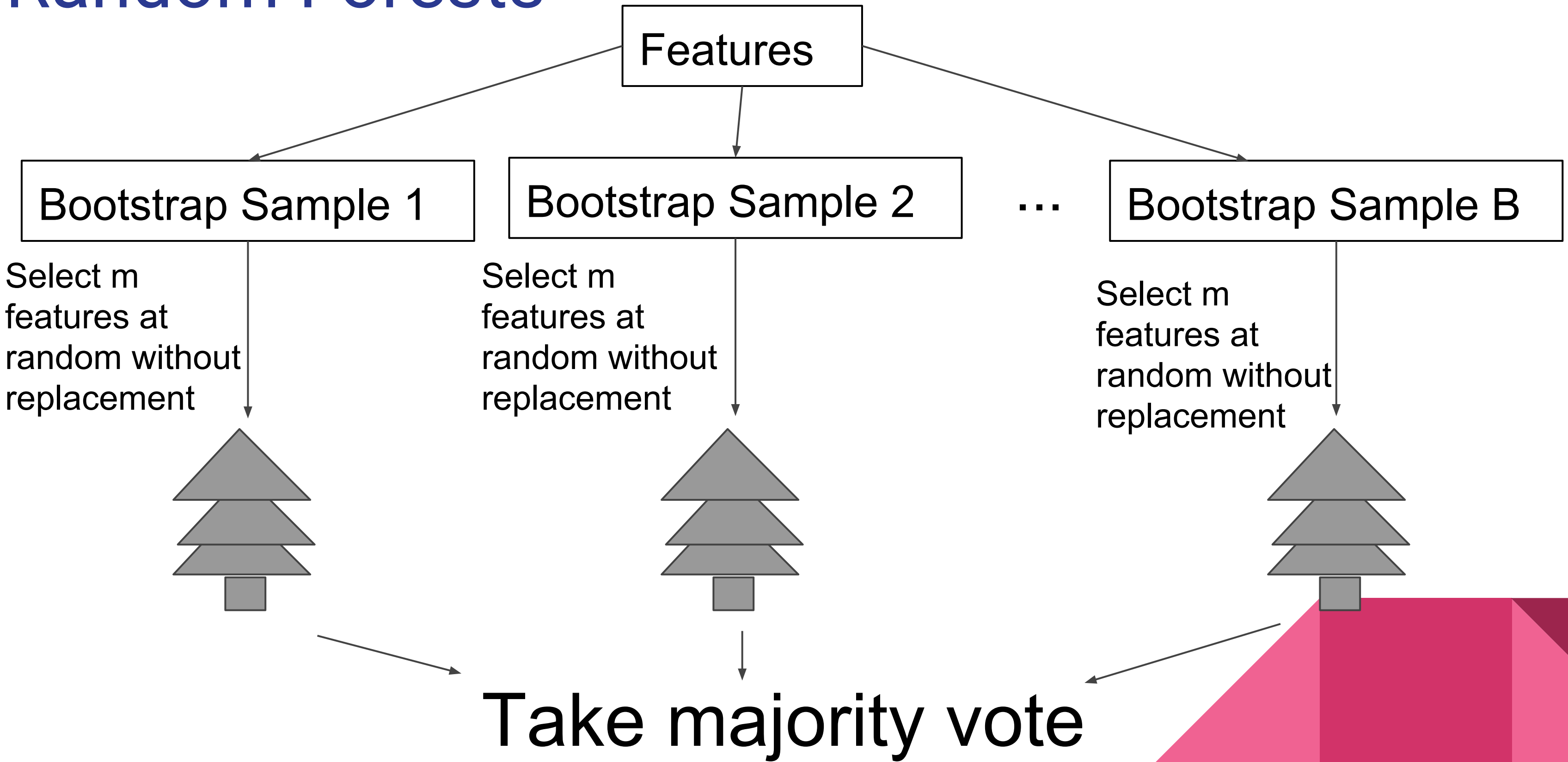


Random Forests

-Bootstrap



Random Forests



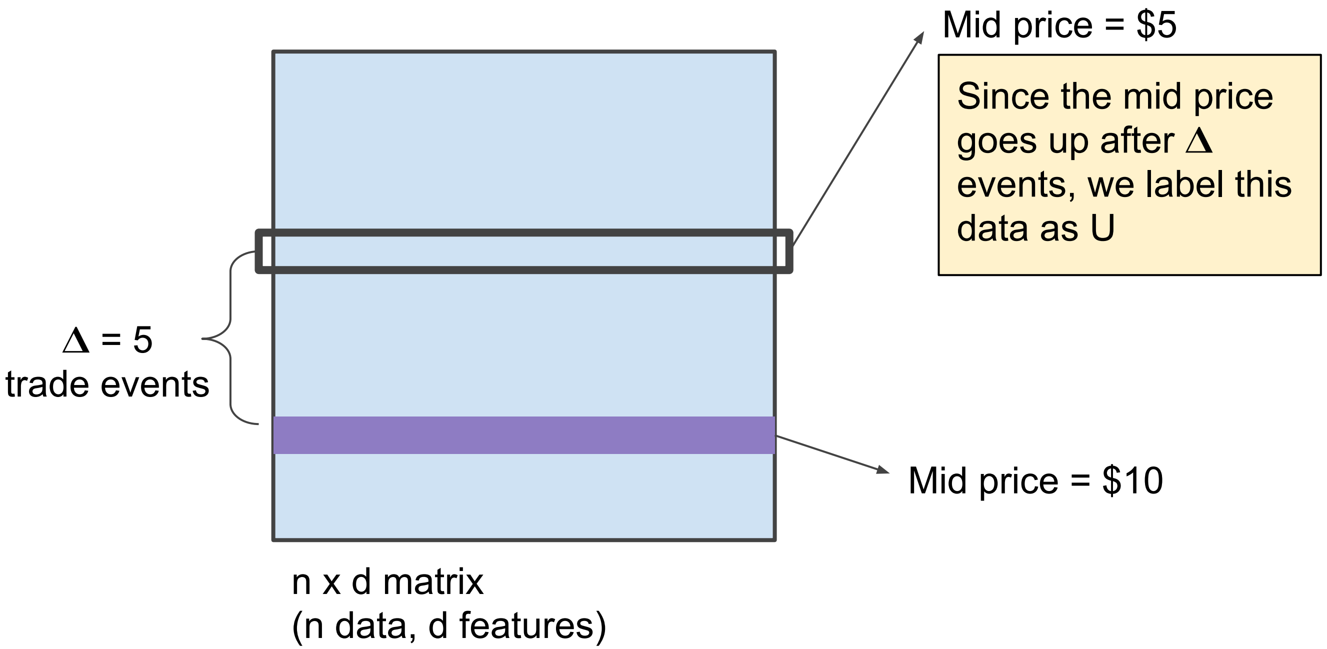
Connections?

SVM
Random forest

Price Movement Prediction

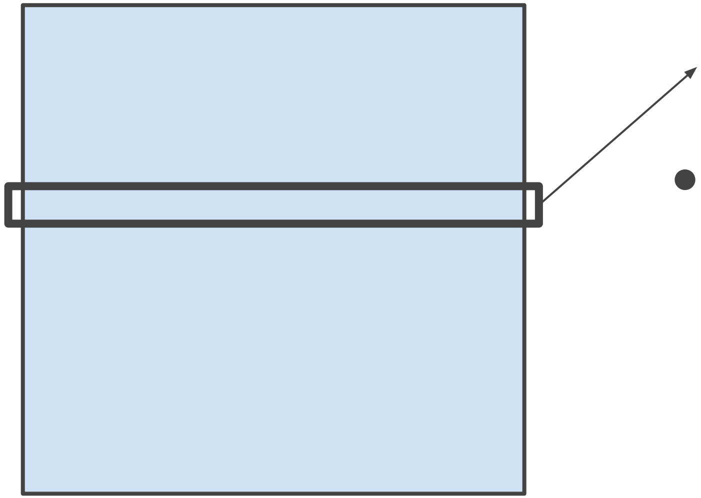
Classification Problem

Class Label Definition: Mid-price movement



Label can be
1) Upward
2) Downward
3) Stationary

Feature Extraction



n x d matrix
(n data, d features)

LOB Snapshot (10 Level Bid, 10 Level Ask)

- Feature vector set

- Each LOB Snapshot will have its own associated features

Feature Vector Set

Basic Set	Description(i = *level index*, $n = 10$)
$v_1 = \{P_i^{ask}, V_i^{ask}, P_i^{bid}, V_i^{bid}\}_{i=1}^n,$	*price and volume (n levels)*

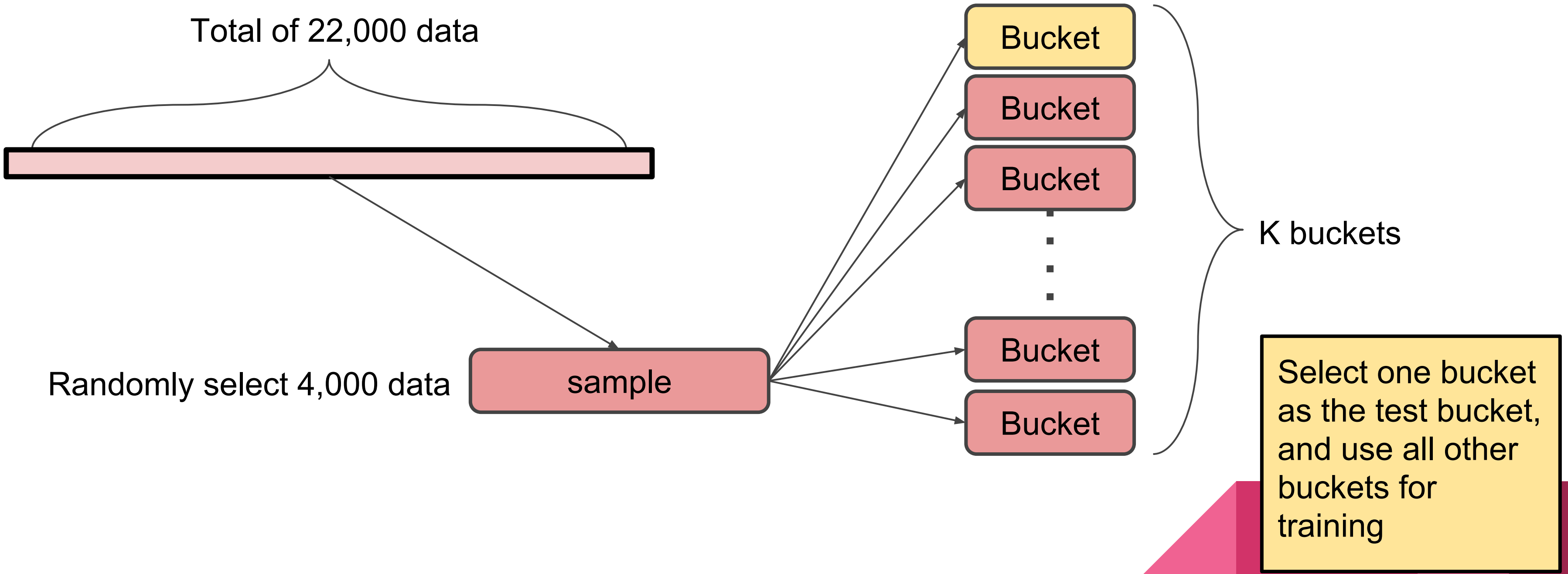
Time-insensitive Set	Description(i = *level index*)
$v_2 = \{(P_i^{ask} - P_i^{bid}), (P_i^{ask} + P_i^{bid})/2\}_{i=1}^n,$	*bid-ask spreads and mid-prices*
$v_3 = \{P_n^{ask} - P_1^{ask}, P_1^{bid} - P_n^{bid}, \lvert P_{i+1}^{ask} - P_i^{ask}\rvert, \lvert P_{i+1}^{bid} - P_i^{bid}\rvert\}_{i=1}^n,$	*price differences*
$v_4 = \{\frac{1}{n}\sum_{i=1}^n P_i^{ask}, \frac{1}{n}\sum_{i=1}^n P_i^{bid}, \frac{1}{n}\sum_{i=1}^n V_i^{ask}, \frac{1}{n}\sum_{i=1}^n V_i^{bid}\},$	*mean prices and volumes*
$v_5 = \{\sum_{i=1}^n (P_i^{ask} - P_i^{bid}), \sum_{i=1}^n (V_i^{ask} - V_i^{bid})\},$	*accumulated differences*

Time-sensitive Set	Description(i = *level index*)
$v_6 = \{dP_i^{ask}/dt, dP_i^{bid}/dt, dV_i^{ask}/dt, dV_i^{bid}/dt\}_{i=1}^n,$	*price and volume derivatives*
$v_7 = \{\lambda_{\Delta t}^{la}, \lambda_{\Delta t}^{lb}, \lambda_{\Delta t}^{ma}, \lambda_{\Delta t}^{mb}, \lambda_{\Delta t}^{ca}, \lambda_{\Delta t}^{cb}\}$	*average intensity of each type*
$v_8 = \{1_{\{\lambda_{\Delta t}^{la} > \lambda_{\Delta T}^{la}\}}, 1_{\{\lambda_{\Delta t}^{lb} > \lambda_{\Delta T}^{lb}\}}, 1_{\{\lambda_{\Delta t}^{ma} > \lambda_{\Delta T}^{ma}\}}, 1_{\{\lambda_{\Delta t}^{mb} > \lambda_{\Delta T}^{mb}\}}\},$	*relative intensity indicators*
$v_9 = \{d\lambda^{ma}/dt, d\lambda^{lb}/dt, d\lambda^{mb}/dt, d\lambda^{la}/dt\},$	*accelerations(market/limit)*

[Kercheval 2014]

- Adding Additional Feature Improves Performance
- Certain Features may be more significant than others (Econ. Set)

K-fold Cross Validation and Performance



Data and Definitions

- SPY and AAPL Data - 05/10/2012 (30minutes only)
- Δ := Horizon of predicting time
- Distribution (U,D,S) := number of labels seen
- Economical set := V1 to V6 (Basic Set + Time Insensitive + Time Sensitive)

compare various ML algorithms ...

Performance Measurement

For each label/class y:

- Precision: P = #(correctly labeled y)/#(y in the predictions)
- Recall: R = #(correctly labeled y)/#(y in the sample)
- F1-measure: F1 = 2PR/(P + R)

Experimental Results - SVM w/ Linear kernel

SPY	Precision	Recall	F1 Measure
UP	'NA'	18.8%	'NA'
DOWN	28.6%	42.7%	34.3%
STATIONARY	64.1%	45.0%	52.9%

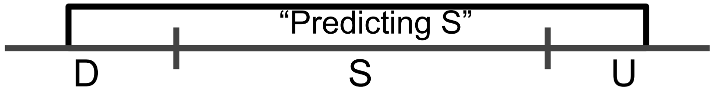
- Data is inseparable with Linear Kernel
- Try different Kernel
- NA = Does not predict U/D/S

- Δ = 30
- (U,D,S) SPY = (4761, 5390, 11385)

Experimental Results - SVM w/ RBF kernel

SPY	Precision	Recall	F1 Measure
UP	93.2%	27.4%	41.2%
DOWN	95.3%	23.3%	37.3%
STATIONARY	60.1%	98.7%	74.7%

- For RBF: Precision is good but Recall is bad
- Predicting too many stationary



- Δ = 30
- (U,D,S) SPY = (4761, 5390, 11385)

Experimental Results - Random Forest

SPY	Precision	Recall	F1 Measure
UP	85.2%	73.1%	78.6%
DOWN	85.4%	81.9%	83.6%
STATIONARY	84.2%	90.8%	87.4%

AAPL	Precision	Recall	F1 Measure
UP	84.3%	80.2%	82.1%
DOWN	81.9%	89.5%	85.5%
STATIONARY	72.7%	44.9%	55.1%

- Δ = 30
- (U,D,S) SPY = (4761, 5390, 11385)
- (U,D,S) AAPL = (4972, 5807, 777)

- Random Forest performs better in terms of overall performance (especially when the distribution is around (1,1,2))
- AAPL has worse stationary performance because Δ = 30 does not have enough stationary samples in the training set
  - can remedy this by having different Δ optimized for unique assets

Feature Vector Set Insights

Basic Set	Description(i = *level index*, $n = 10$)
$v_1 = \{P_i^{ask}, V_i^{ask}, P_i^{bid}, V_i^{bid}\}_{i=1}^n,$	*price and volume (n levels)*

Time-insensitive Set	Description(i = *level index*)
$v_2 = \{(P_i^{ask} - P_i^{bid}), (P_i^{ask} + P_i^{bid})/2\}_{i=1}^n,$	*bid-ask spreads and mid-prices*
$v_3 = \{P_n^{ask} - P_1^{ask}, P_1^{bid} - P_n^{bid}, \lvert P_{i+1}^{ask} - P_i^{ask}\rvert, \lvert P_{i+1}^{bid} - P_i^{bid}\rvert\}_{i=1}^n,$	*price differences*
$v_4 = \{\frac{1}{n}\sum_{i=1}^n P_i^{ask}, \frac{1}{n}\sum_{i=1}^n P_i^{bid}, \frac{1}{n}\sum_{i=1}^n V_i^{ask}, \frac{1}{n}\sum_{i=1}^n V_i^{bid}\},$	*mean prices and volumes*
$v_5 = \{\sum_{i=1}^n (P_i^{ask} - P_i^{bid}), \sum_{i=1}^n (V_i^{ask} - V_i^{bid})\},$	*accumulated differences*

Time-sensitive Set	Description(i = *level index*)
$v_6 = \{dP_i^{ask}/dt, dP_i^{bid}/dt, dV_i^{ask}/dt, dV_i^{bid}/dt\}_{i=1}^n,$	*price and volume derivatives*
$v_7 = \{\lambda_{\Delta t}^{la}, \lambda_{\Delta t}^{lb}, \lambda_{\Delta t}^{ma}, \lambda_{\Delta t}^{mb}, \lambda_{\Delta t}^{ca}, \lambda_{\Delta t}^{cb}\}$	*average intensity of each type*
$v_8 = \{1_{\{\lambda_{\Delta t}^{la} > \lambda_{\Delta T}^{la}\}}, 1_{\{\lambda_{\Delta t}^{lb} > \lambda_{\Delta T}^{lb}\}}, 1_{\{\lambda_{\Delta t}^{ma} > \lambda_{\Delta T}^{ma}\}}, 1_{\{\lambda_{\Delta t}^{mb} > \lambda_{\Delta T}^{mb}\}}\},$	*relative intensity indicators*
$v_9 = \{d\lambda^{ma}/dt, d\lambda^{lb}/dt, d\lambda^{mb}/dt, d\lambda^{la}/dt\},$	*accelerations(market/limit)*

- Biggest Effect on improving our measures was v6, the price and volume derivatives taken from the message book data

Real Implementation Considerations and Future Work

- More training data (aka: window size) will give better results but takes longer
  - currently running on 2.3GHz intel core i-7 (2012) mac -> 30 minute data
  - solution: get a supercomputer and use a window size (aka dataset) that optimally trades off compute time with sufficient update frequency -> several hours of data
- Optimize Information gain from each feature to determine optimal economical feature vector set
- Use random forest to determine feature importance