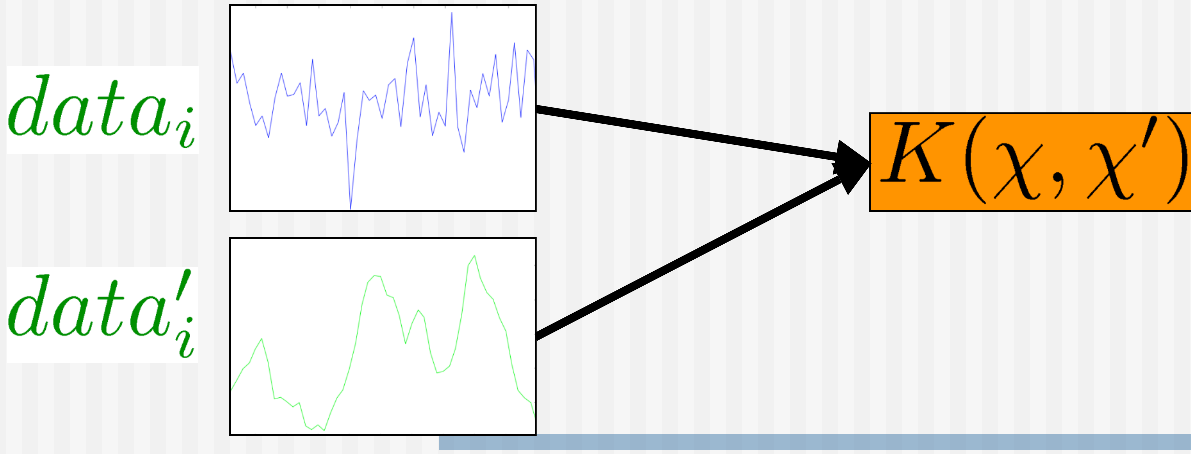
Point Set Kernel Applications to Time Series

- The Bhattacharyya Kernel in Hilbert Space
- Time Series Classification
- Simulated Data
- Real Data
- Conclusion



- The series is an unordered set of vectors:

$$\{x_i\}_1^k, \{x'_i\}_1^{k'} = \{(i, data_i)\}_1^k, \{(i, data'_i)\}_1^{k'}$$

- Map each set of vectors into Hilbert Space via rbf kernel

$$\kappa(\chi_1, \chi_2) = exp(-||\chi_1 - \chi_2||/(2\sigma^2))$$

- Fit Gaussian to data in Hilbert Space

$$\{\Phi_\kappa(x_i)\} \sim \mathcal{N}(\mu, \sigma^2), \{\Phi_\kappa(x'_i)\} \sim \mathcal{N}(\mu', \sigma'^2)$$

- K is Bhattacharyya between restricted Gaussians

Time Series Classification

- Include variance for financial time series

$$return_i = 100 \log \left(\frac{price_i}{price_{i-1}} \right)$$

$$X = \{x_i\} = \{(i, return_i, Var(return_i))\}$$

- Classify based on fundamental characteristics

$$X \in Financials, X' \in ConsumerGoods, ..., etc$$

- May use any kernel based classification or clustering technique
- SVM one-versus-all strategy

Simulated Data

- Simulate multiple financial time series with ARMA-GARCH model.

ARMA(p,q)-GARCH(m,s):

$$r_t = \phi_0 + \sum_{i=1}^{p} \phi_i r_{t-i} + a_t - \sum_{i=1}^{q} \theta_i a_{t-i},$$

$$a_t = \sigma_t \epsilon_t,$$

$$\sigma_t^2 = \alpha_0 + \sum_{i=1}^{m} \alpha_1 a_{t-i}^2 + \sum_{j=1}^{s} \beta_j \sigma_{t-j}^2$$

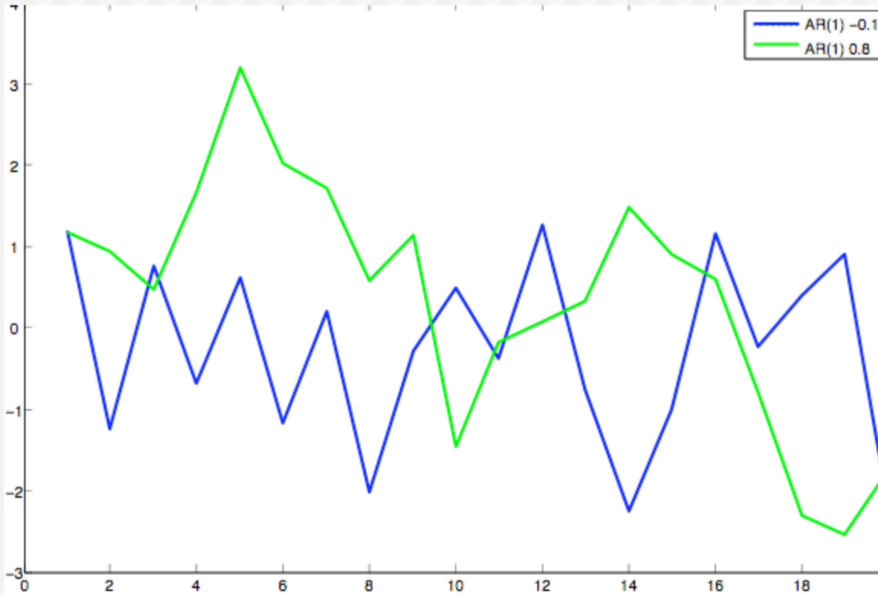
where ϵ_t i.i.d. R.V.s.

Simulated Data

- Choose two distinct sets of model parameters: Θ_1, Θ_2
- Simulate ~50 time series from each model
- Compute pairwise Bhattacharyya kernels
- Classify with an SVM.
- ~7-20% classification error using average performance on multiple splits.
- Error increases as model params converge

Simulated Data

- 2 sample AR(1) series with different params



- Note: good performance with only ~15-20 time points
- Interesting to see which ARMA-GARCH params cause the best decision boundaries.

Real Data

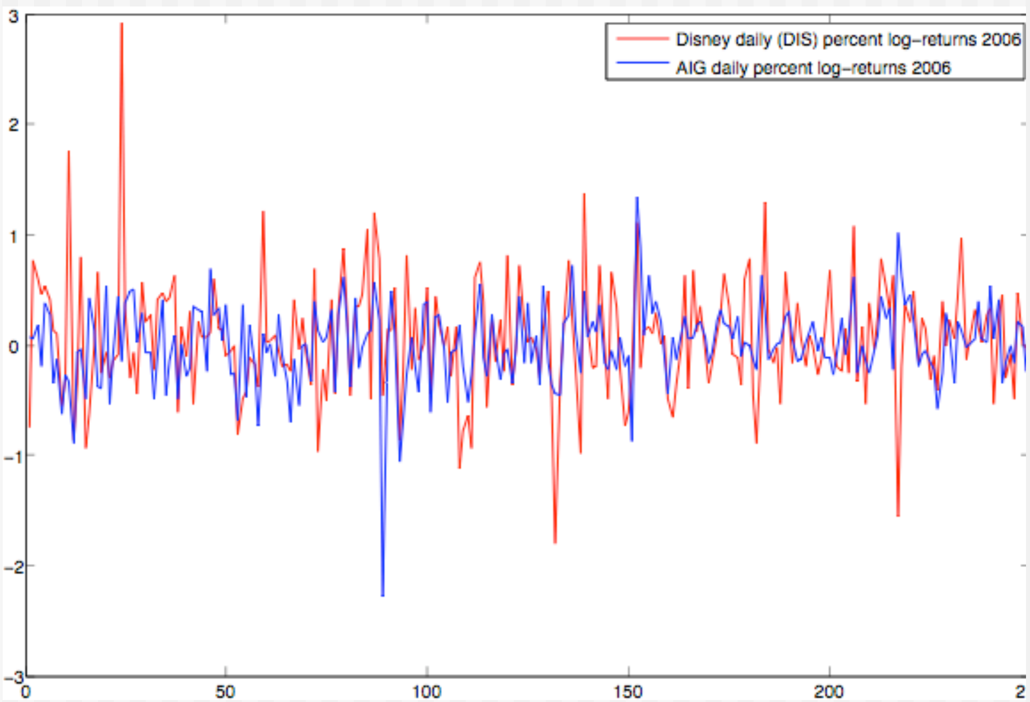
- Percentage daily log returns for large cap stocks over 2006 (250 days).

- Kernel inner products clustered

- No well defined decision boundary

Real Data

- Stylistically, data very similar
- Leads to poor spread in the kernel



Conclusion

- Results in simulated data motivate a further look into real data.

- Preprocessing of real return data necessary

- Try kernelized clustering algorithms to detect similarities.

- Test robustness of classifier with respect to length and origin of time series.

- Tune kernel parameters