Guided Tour of Machine Learning in Finance

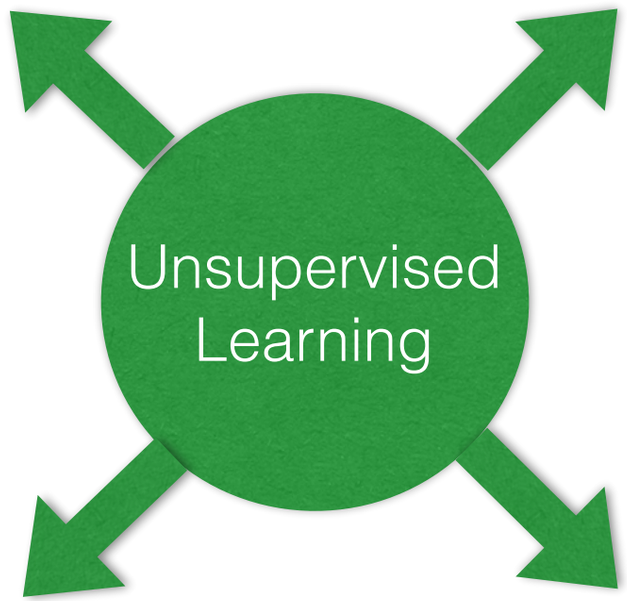
Week 3: Unsupervised Learning

Core concept of UL

Igor Halperin

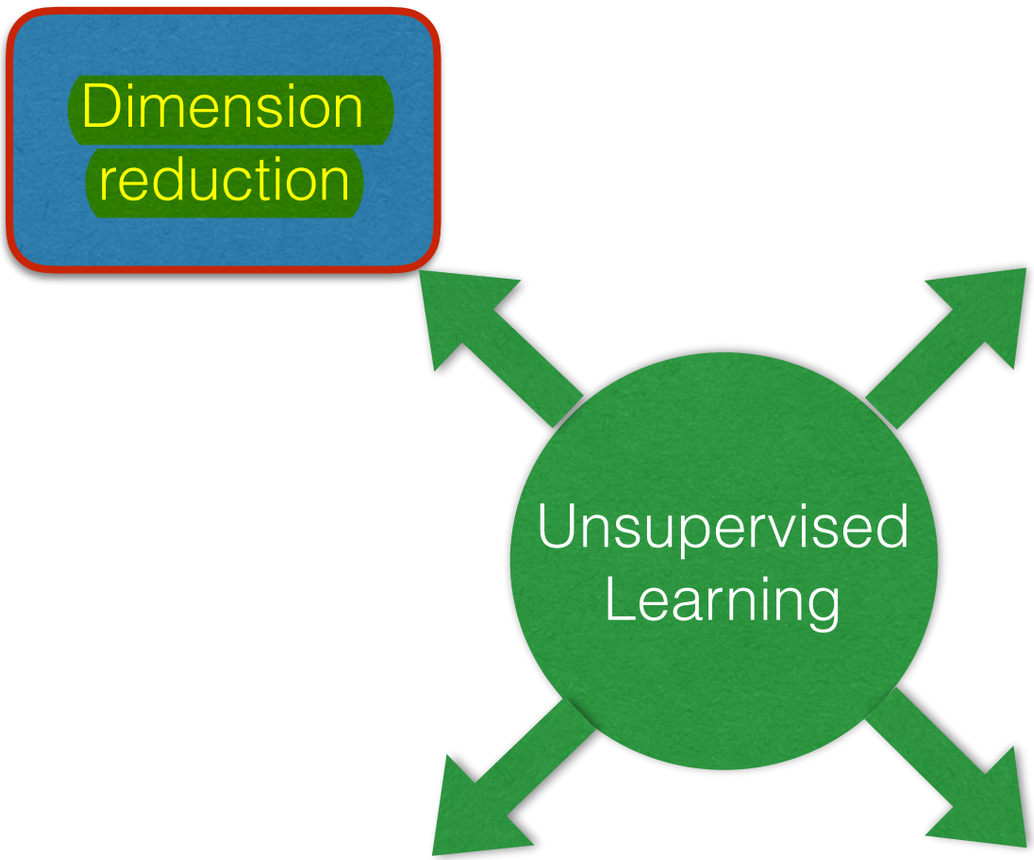
NYU Tandon School of Engineering, 2017

Unsupervised Learning



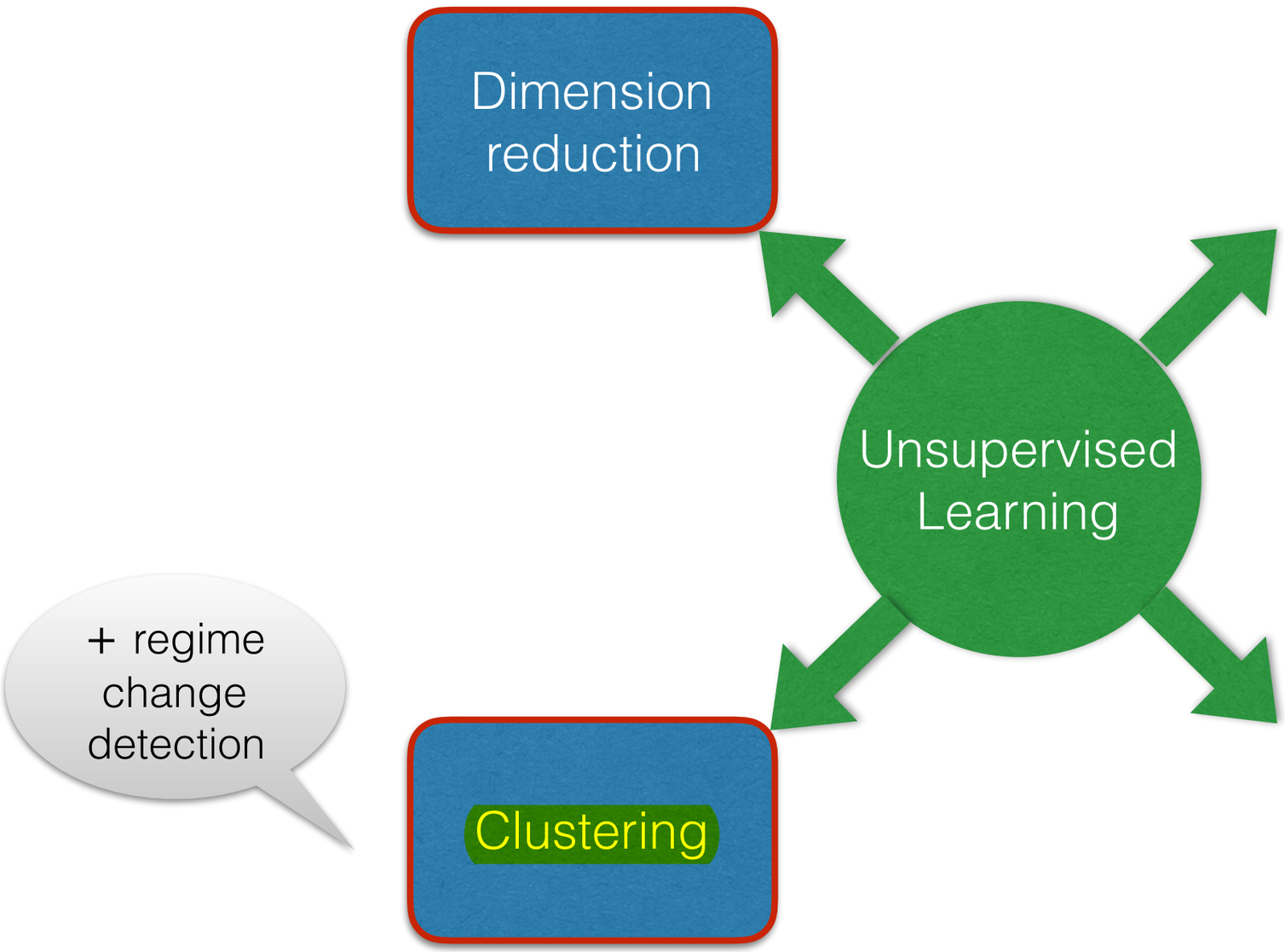
Learning = ability to generalize
(for UL) = ability to learn the data representation

Unsupervised Learning



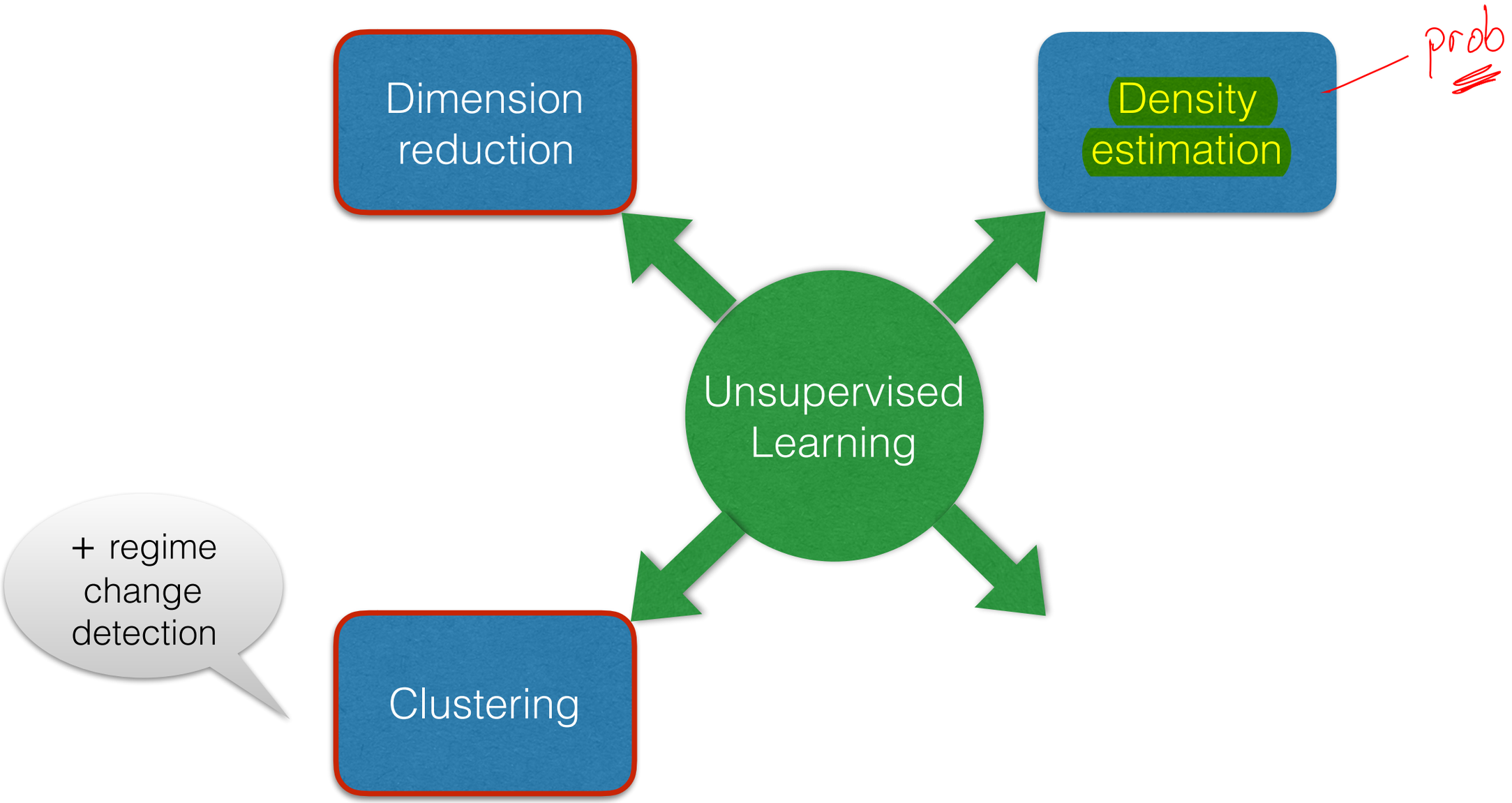
Learning = ability to generalize
(for UL) = ability to learn the data representation

Unsupervised Learning



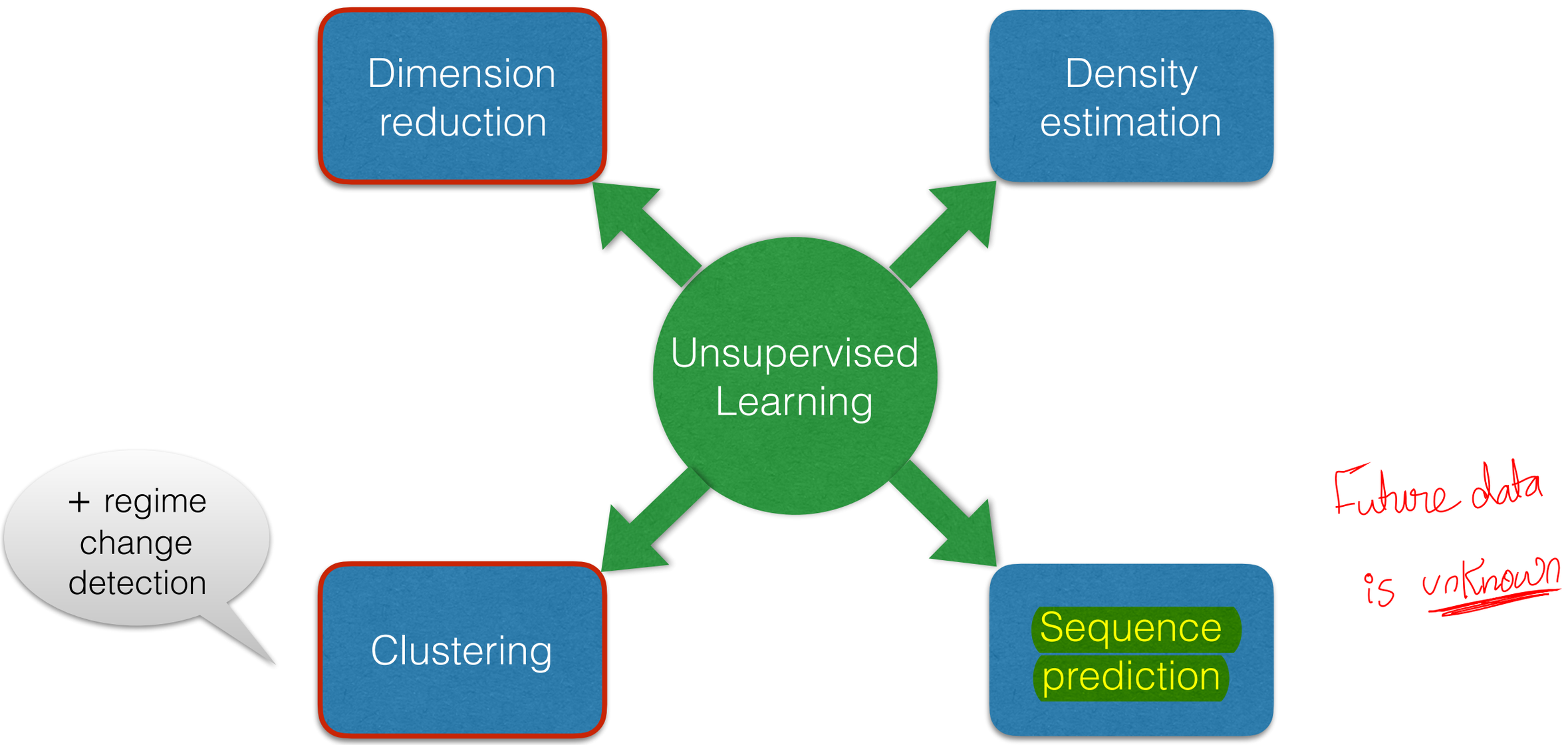
Learning = ability to generalize
(for UL) = ability to learn the data representation

Unsupervised Learning



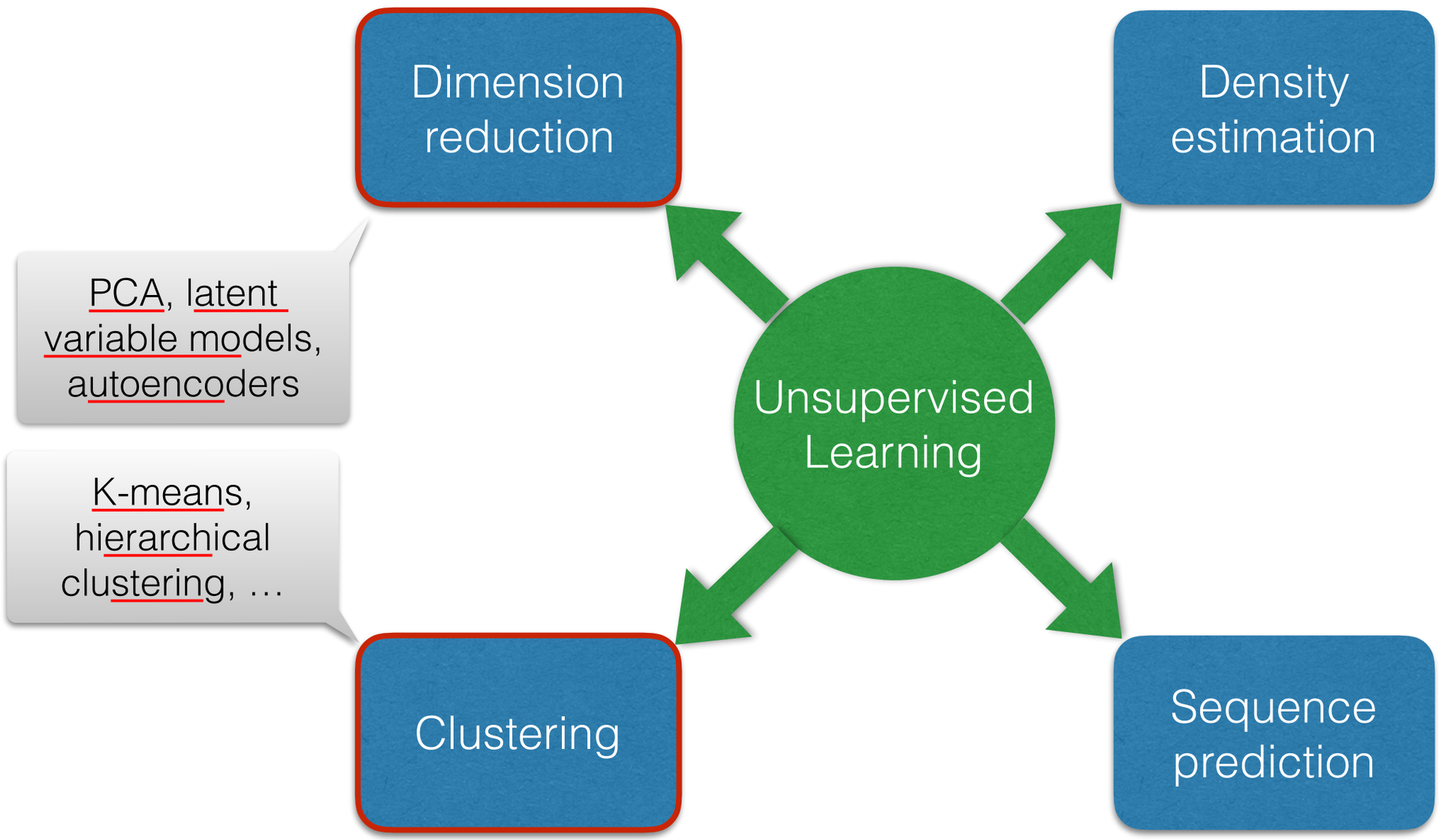
Learning = ability to generalize
(for UL) = ability to learn the data representation

Unsupervised Learning



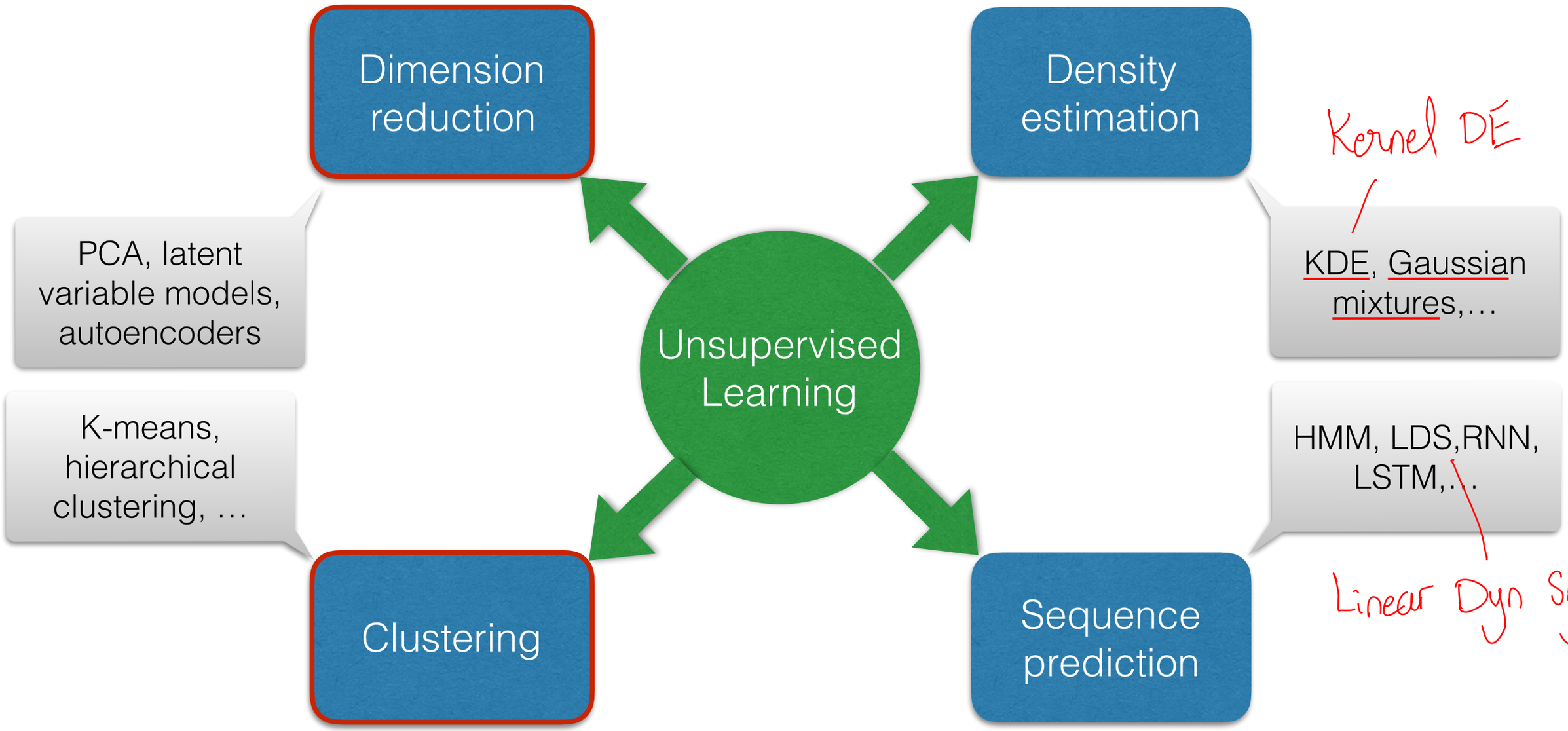
Learning = ability to generalize
(for UL) = ability to learn the data representation

Unsupervised Learning



Learning = ability to generalize
(for UL) = ability to learn the data representation

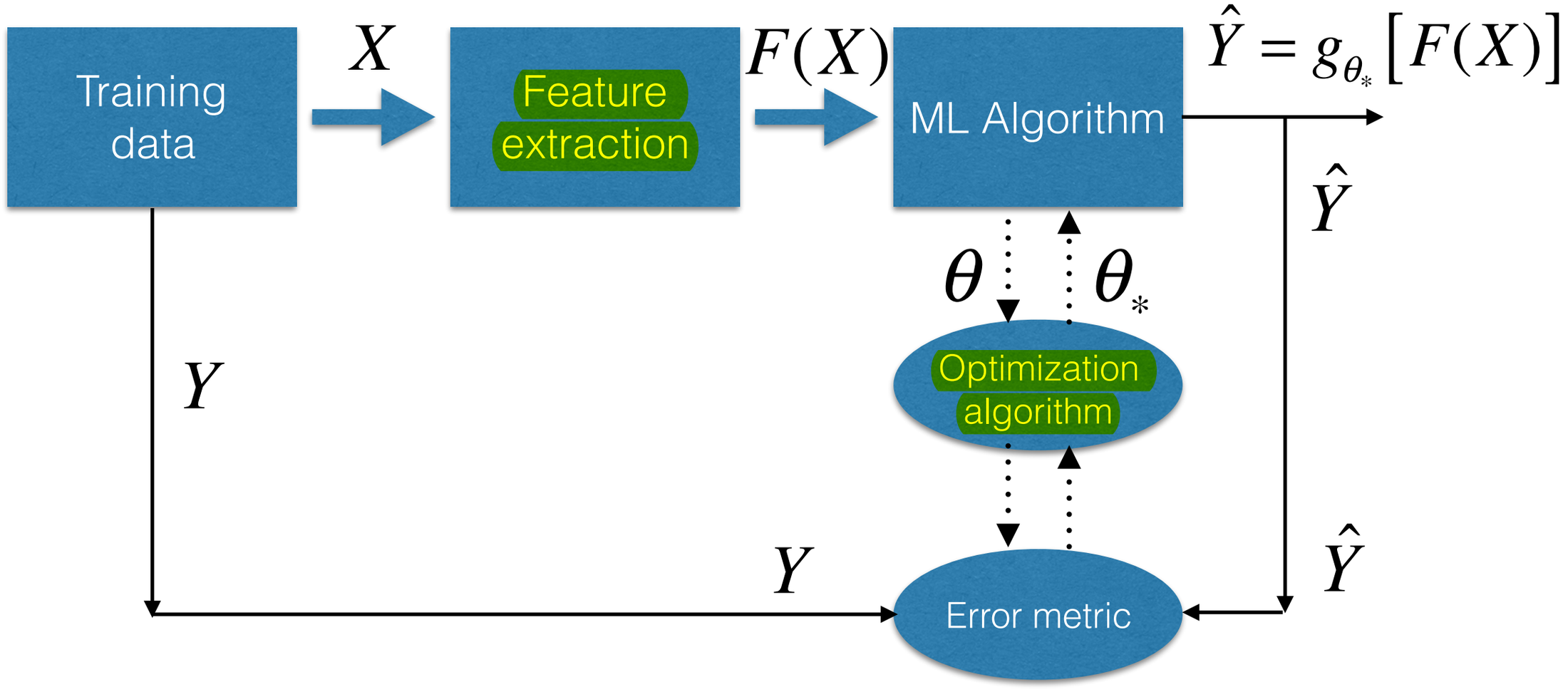
Unsupervised Learning



Learning = ability to generalize
(for UL) = ability to learn the data representation

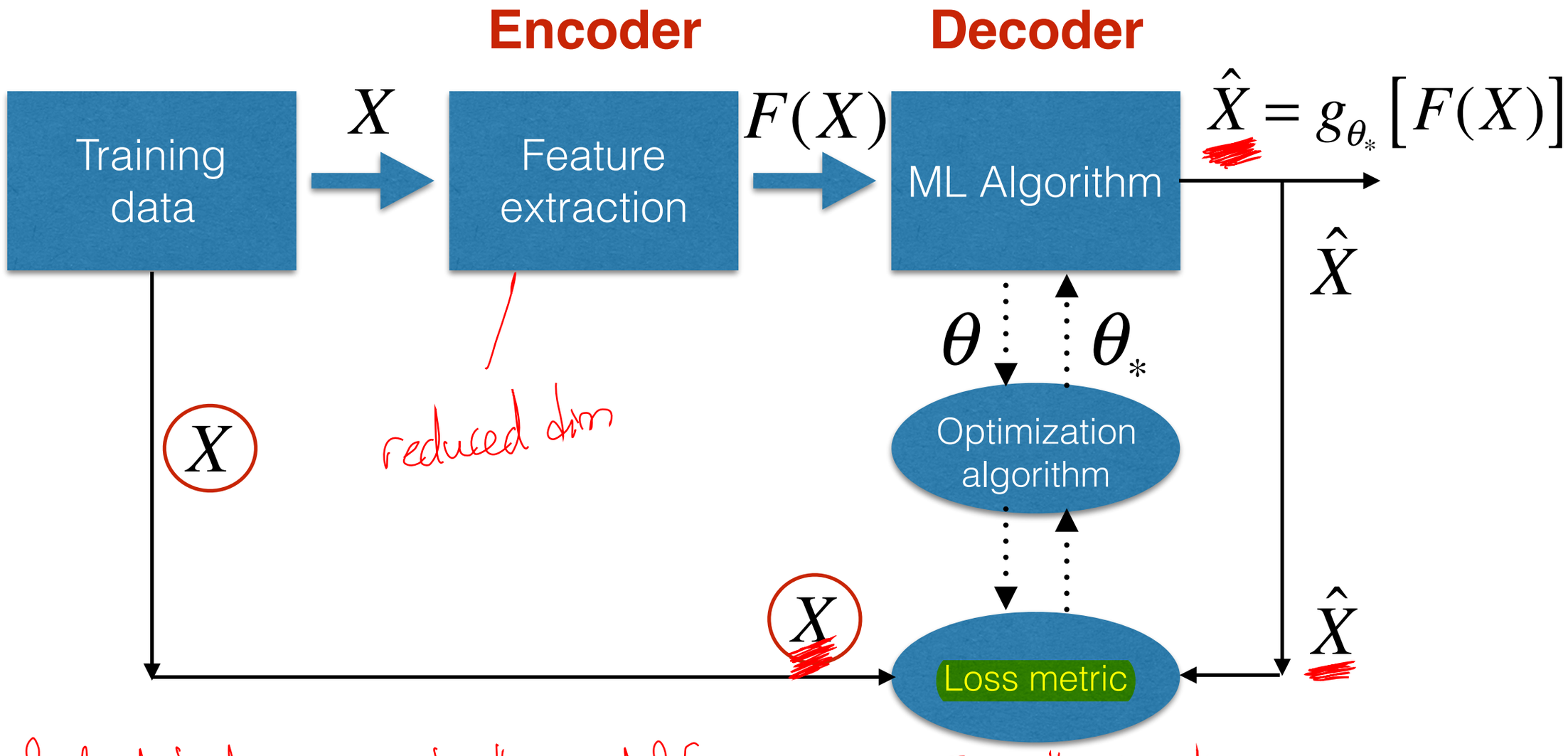
Supervised learning diagram

The general process for training supervised learning algorithms



Unsupervised learning diagram: Autoencoder

Autoencoder: replace Y with X!

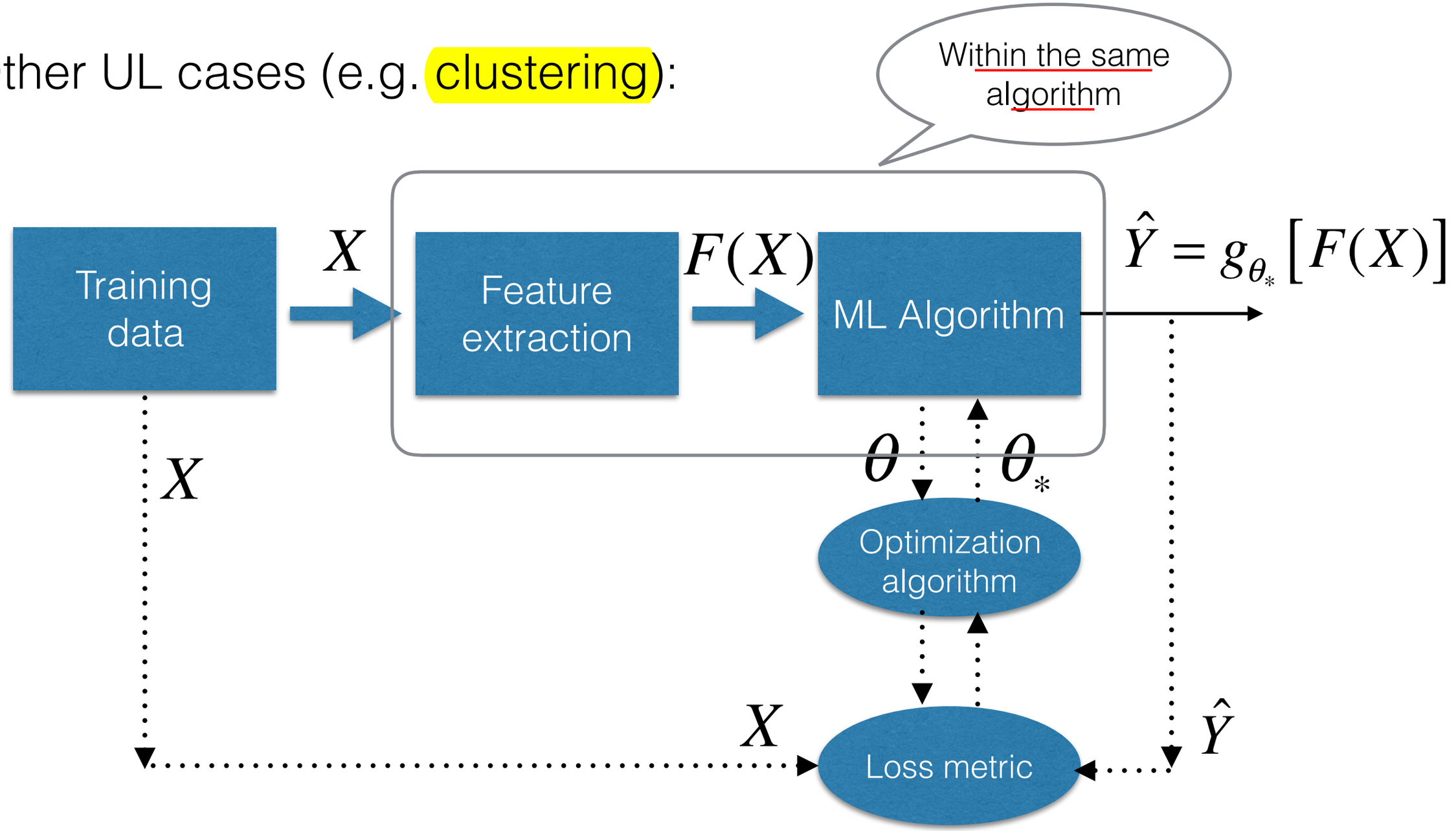


* Model Architecture prevents the model from memorizing the input.

This is done because of the reduced dim of encoded input

Unsupervised learning diagram

Other UL cases (e.g. clustering):



Control question

Q: Select all correct statements:

1. As there is no “teacher” in Unsupervised Learning, it can not have a well-defined and practically useful objective.
2. The general objective of Unsupervised Learning is to learn useful representations of data.
3. A loss function for Unsupervised Learning can be obtained by comparing true labels for available data with labels produced by the model.
4. A loss function for Unsupervised Learning can be obtained by comparing an actual data distribution with a recovered data distribution as obtained from the model.

Correct answers: 2, 4.

Guided Tour of Machine Learning in Finance

Week 3-1-2-1: Unsupervised Learning

Principal Component Analysis (PCA)

Igor Halperin

NYU Tandon School of Engineering, 2017

Stock returns

Analyze returns of 30 stocks from the DJI index:

$$r_i(t) = \log\left(\frac{S_i(t)}{S_i(t-\Delta t)}\right), \quad i = 1,\ldots,N_{stocks}$$

$$r_i(t) = \alpha_i + \beta_i r_M(t) + \varepsilon_{it}$$

a simple linear reg of all log-returns on a single predictor

Here:

$r_i(t)$: the log-return for stock i

$r_M(t)$: the log-return for the "market" M (the DJI index)

β_i : the "beta" - measure of correlation with the market

ε_{it} : the "noise" term (residual) $\Leftrightarrow \mathbb{E}[\varepsilon_{it}] = 0$

Δt : the time step (1m, 1h, 1d, 1w, 1m, 1y, etc.)

Market Factor

This is the regression equation of the Capital Asset Pricing Model (CAPM, see e.g. Cochrane, "Asset Pricing" 2005).

Eigen-portfolios as a UL taks

Trading signals for statistical arbitrage from the ML perspective

trying to filter out impact of the market from stock return

In statistical arbitrage strategies:

- Trading signals are systematic or rule-based
- Trading book has hero beta with the market (i.e. it is market-neutral)
- Use diversification to produce low volatility portfolios uncorrelated with the market

i.e market is too random to be predicted

$$\hat{r}_i(t) = \hat{\alpha}_i + \hat{\beta}_i\, r_M(t)$$

$$\hat{\varepsilon}_i(t) = r_i(t) - \hat{r}_i(t) = r_i(t) - \hat{\alpha}_i - \hat{\beta}_i\, r_M(t)$$

* We subtract market impact & focus on the residuals, which are parts of the equity return that are explained by the market returns.

* Residuals are used as features to produce trading signals.

Eigen-portfolios as a UL taks

Trading signals for statistical arbitrage from the ML perspective

In statistical arbitrage strategies:

- Trading signals are systematic or rule-based
- Trading book has hero beta with the market (i.e. it is market-neutral)
- Use diversification to produce low volatility portfolios uncorrelated with the market
- See Avellaneda and Lee (2008) for for details
- Multi-factor model: individual returns $R_i \ (i = 1,..,N)$ are driven by m factors $F_j \ (j = 1,..,m)$ and idiosyncratic components $\tilde{R}_i$ with $\mathbb{E}\left[\tilde{R}_i\right] = 0$

$$R_i = \alpha_i + \sum_{j=1}^{m} \beta_{ij} F_j + \tilde{R}_i$$

Eigen-portfolios as a UL taks

Trading signals for statistical arbitrage from the ML perspective

In statistical arbitrage strategies:

- Trading signals are systematic or rule-based
- Trading book has hero beta with the market (i.e. it is market-neutral)
- Use diversification to produce low volatility portfolios uncorrelated with the market
- See Avellaneda and Lee (2008) for for details
- Multi-factor model: individual returns $R_i \ (i = 1,..,N)$ are driven by m factors $F_j \ (j = 1,..,m)$ and idiosyncratic components $\tilde{R}_i$ with $\mathbb{E}\left[\tilde{R}_i\right] = 0$

$$R_i = \alpha_i + \sum_{j=1}^{m} \beta_{ij} F_j + \tilde{R}_i$$

- Factors F_j can be thought of as returns of "benchmark" portfolios representing **systematic factors** (e.g. industry and geography factors)

* We focus on a more data-driven approach.

Classic Finance

Control question

Select all correct answers

1. In the factor model, all individual returns are only correlated with "market" M with correlations β_i, but remain uncorrelated between themselves.
2. Correlation of two assets i and j in the factor model is given by $\beta_i \beta_j$.
3. Statistical arbitrage deals with "factor investing" to find factors F_j with highest returns.
4. Statistical arbitrage use residuals of stock returns "unexplained" by factors to extract trading signals.

Correct answer: 2, 4

Guided Tour of Machine Learning in Finance

Week 3-1-2-2: Unsupervised Learning

Principal Component Analysis (PCA)

Igor Halperin

NYU Tandon School of Engineering, 2017

PCA approach: eigen-portfolios

Try PCA as a way to **extract factors** directly from the data

Data: history of $N+1$ daily prices of M stocks $S_n^{(i)} \equiv S_{t_n}^{(i)}$ for $i = 1,\ldots,M$ measured at times $t = [t_0, t_1, \ldots, t_N]$

1. Compute **daily returns:**

$$R_{ni} = \frac{S_n^{(i)} - S_{n-1}^{(i)}}{S_{n-1}^{(i)}} \simeq \log \frac{S_n^{(i)}}{S_{n-1}^{(i)}}, \quad n = 1,\ldots,N, \quad i = 1,\ldots,M$$

2. **Standardized returns** (data normalization):

$$X_{ni} = \frac{R_{ni} - \bar{R}_i}{\bar{\sigma}_i}, \qquad \bar{R}_i = \frac{1}{N}\sum_{n=1}^{N} R_{ni}, \quad \bar{\sigma}_i^2 = \frac{1}{N-1}\sum_{n=1}^{N}\left(R_{ni} - \bar{R}_i\right)^2$$

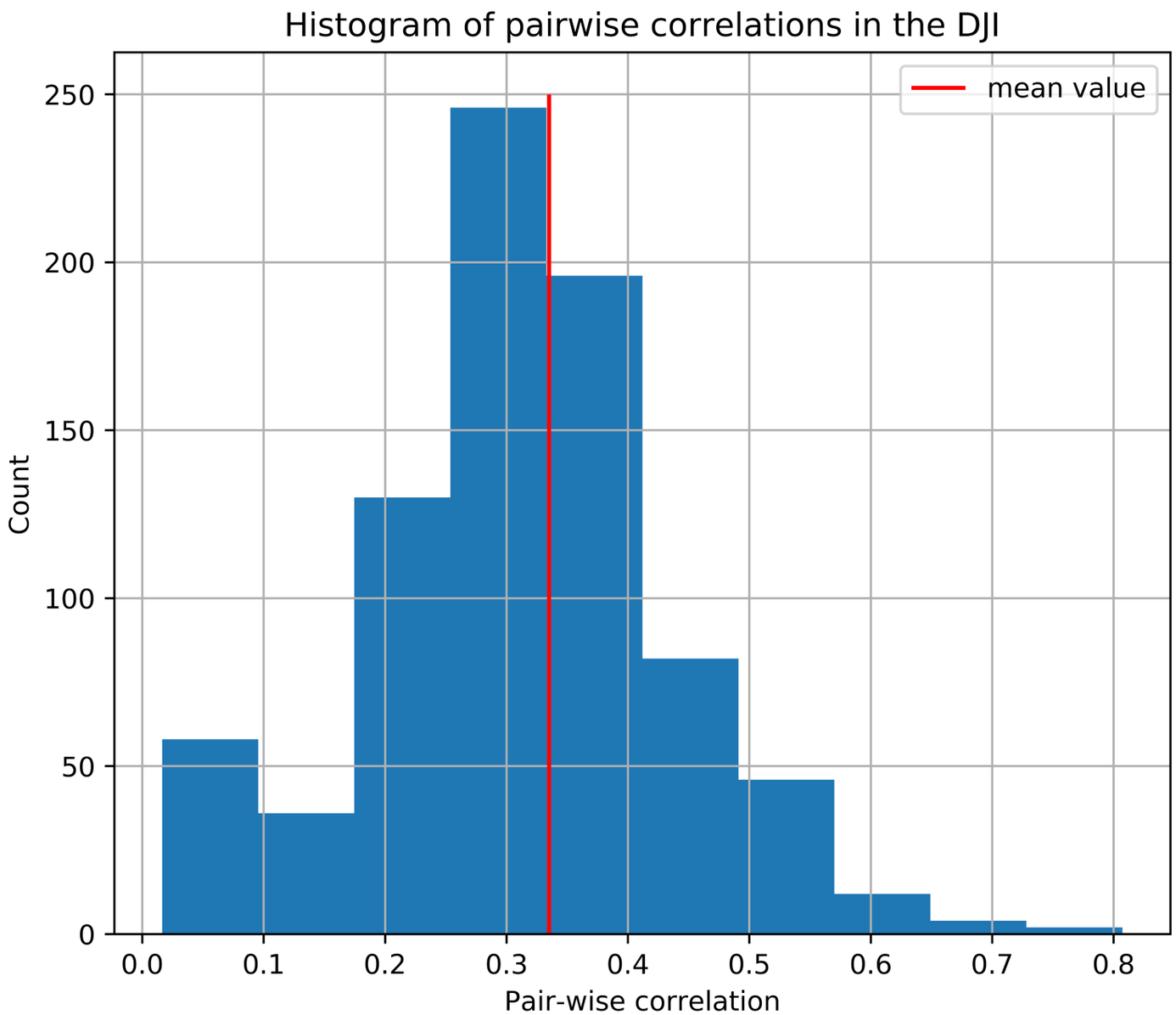
3. The empirical **correlation matrix** is the covariance matrix of standardized returns:

$$C_{ij} = \frac{1}{N-1}\sum_{n=1}^{N} X_{ni} X_{nj} = \frac{1}{N-1}\left(\mathbf{X}^T\mathbf{X}\right)_{ij}$$

This matrix is not diagonal!

Distribution of pairwise correlations in the DJI

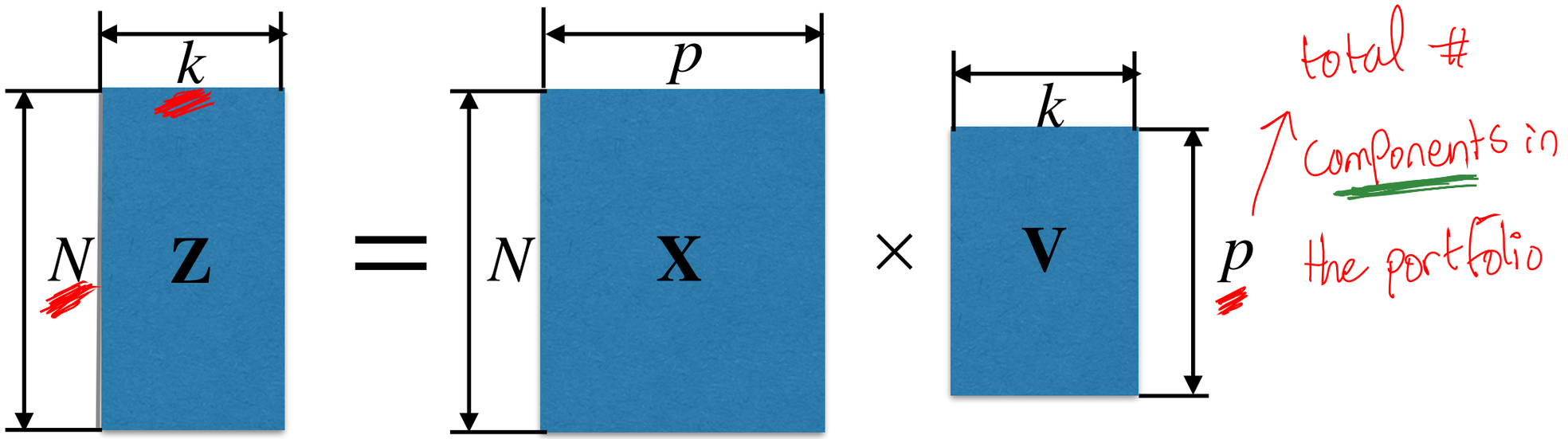
Histogram of pairwise return correlations for stocks in the DJI index:



mean value of distribution $\neq 0$

PCA as a coordinate transform

How to make correlation matrix $\mathbf{C}$ diagonal?
Introduce a linear transform (linear encoder) of the data $\mathbf{Z} = \mathbf{XV}$ parametrized by a $p \times k$ orthogonal matrix $\mathbf{V}$ with $\mathbf{VV}^T = 1$



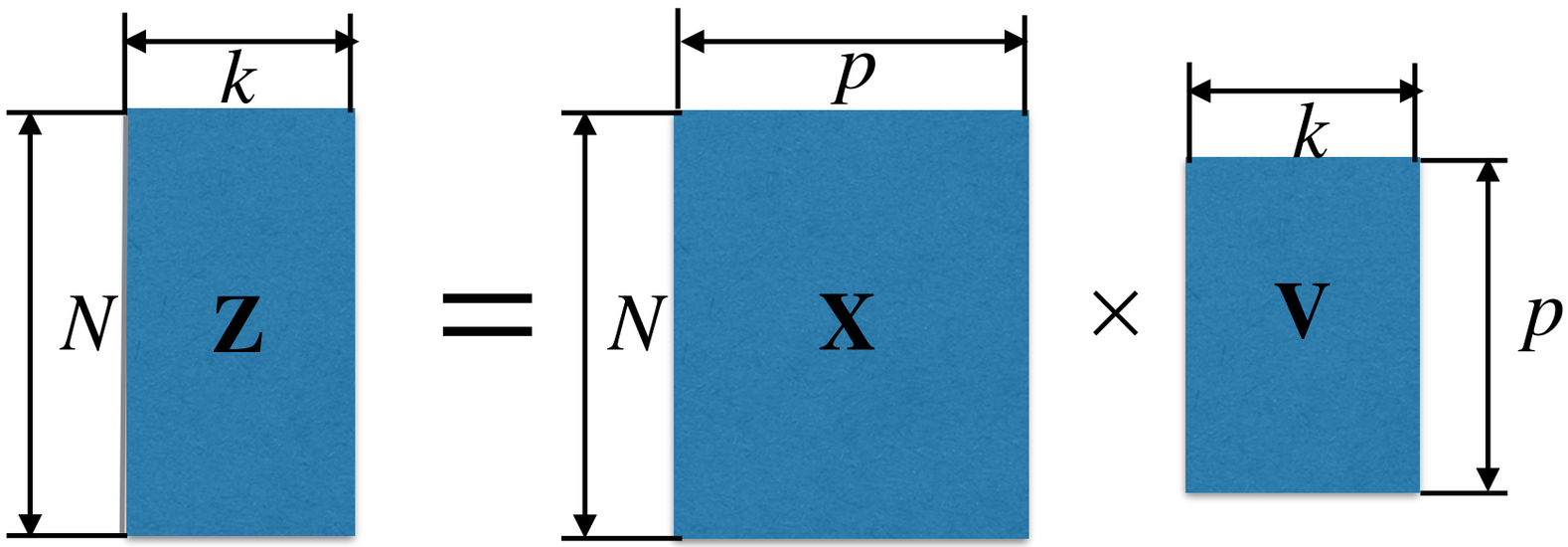
total # components in the portfolio

V is an orthogonal matrix that stores eigenvectors column-wise

$p \times k$ s.t $p \geq k$

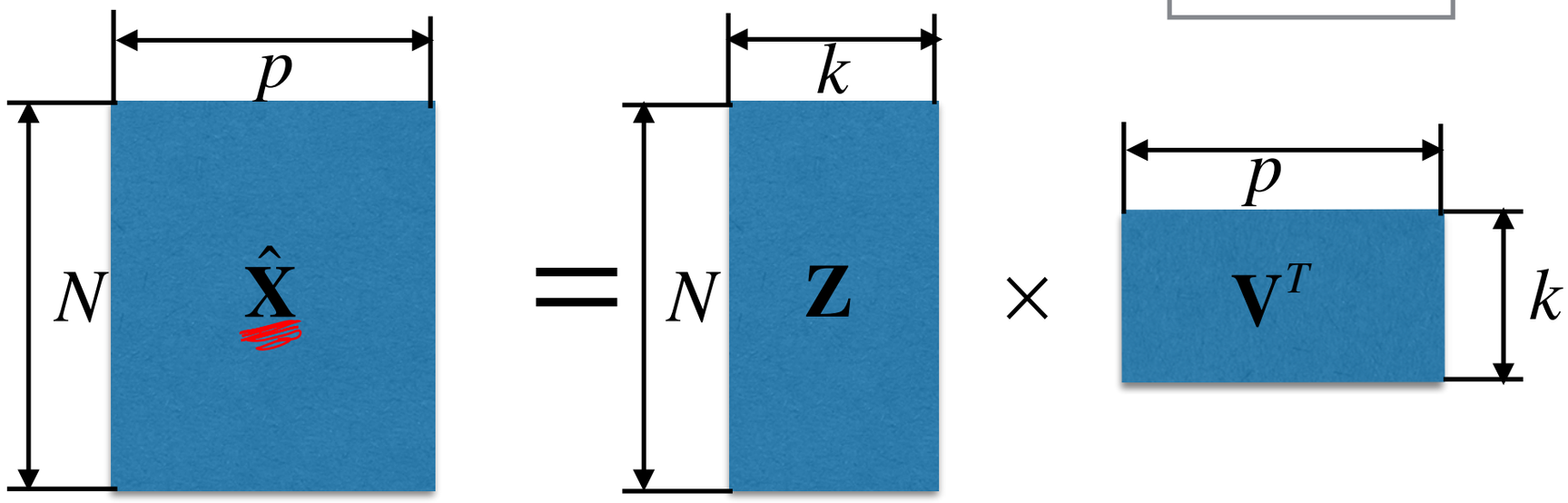
PCA as a coordinate transform

How to make correlation matrix $\mathbf{C}$ diagonal?
Introduce a linear transform (linear encoder) of the data $\mathbf{Z} = \mathbf{XV}$ parametrized by a $p \times k$ orthogonal matrix $\mathbf{V}$ with $\mathbf{VV}^T = 1$



A **decoded signal** (inverse transform) is obtained as $\hat{\mathbf{X}} = \mathbf{ZV}^T$

Same dim as X



PCA: Eigenvalue decomposition

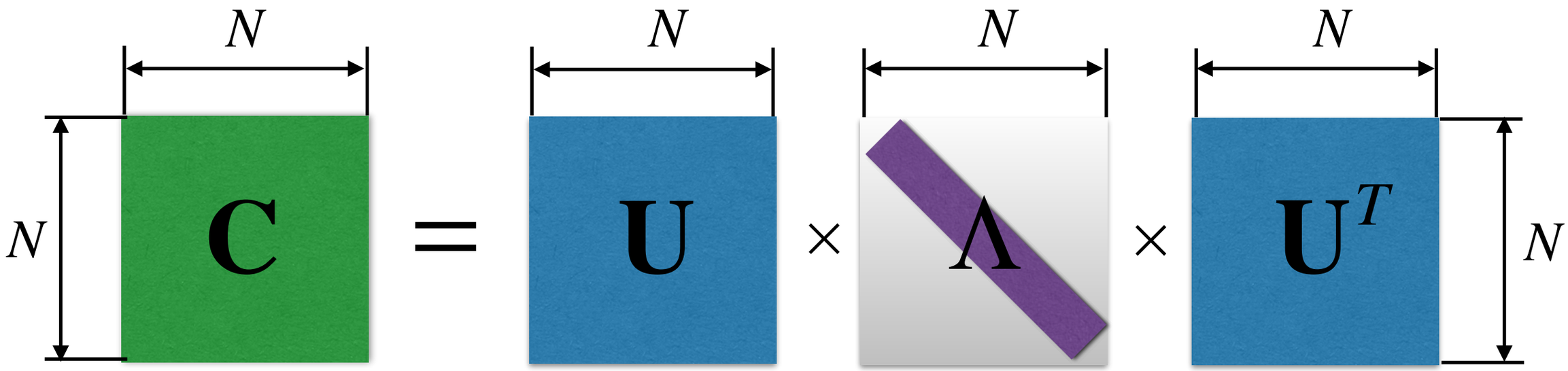
Eigenvalue decomposition of the correlation matrix (note that $\mathbf{C}$ is non-negative definite!)

$$\mathbf{C} = \mathbf{U}\Lambda\mathbf{U}^T$$

where (assuming that $N > p$):

$\Lambda = diag(\lambda_1, .., \lambda_p)$ $(\lambda_1 \geq ... \geq \lambda_p)$ is a diagonal matrix of ordered eigenvalues

$\mathbf{U}$ is a $N \times N$ orthogonal matrix ($\mathbf{U}\mathbf{U}^T = \mathbf{I}$) that stores eigenvectors column-wise



PCA: Eigenvalue decomposition

Eigenvalue decomposition of the correlation matrix (note that $\mathbf{C}$ is non-negative definite!)

$$\mathbf{C} = \mathbf{U}\Lambda\mathbf{U}^T$$

where (assuming that $N > p$):

$\Lambda = diag(\lambda_1, .., \lambda_p)$ $(\lambda_1 \geq ... \geq \lambda_p)$ is a diagonal matrix of ordered eigenvalues

$\mathbf{U}$ is a $N \times N$ orthogonal matrix ($\mathbf{U}\mathbf{U}^T = \mathbf{I}$) that stores eigenvectors column-wise

Use this to compute the covariance matrix of $\mathbf{Z} = \mathbf{X}\mathbf{V}$

$$Cov[\mathbf{Z}] = \frac{1}{N-1}\mathbf{Z}^T\mathbf{Z} = \frac{1}{N-1}\mathbf{V}^T\mathbf{X}^T\mathbf{X}\mathbf{V} = \mathbf{V}^T\mathbf{C}\mathbf{V} = \mathbf{V}^T\mathbf{U}\Lambda\left(\mathbf{V}^T\mathbf{U}\right)^T$$

PCA: Eigenvalue decomposition

Eigenvalue decomposition of the correlation matrix (note that $\mathbf{C}$ is non-negative definite!)

$$\mathbf{C} = \mathbf{U}\Lambda\mathbf{U}^T$$

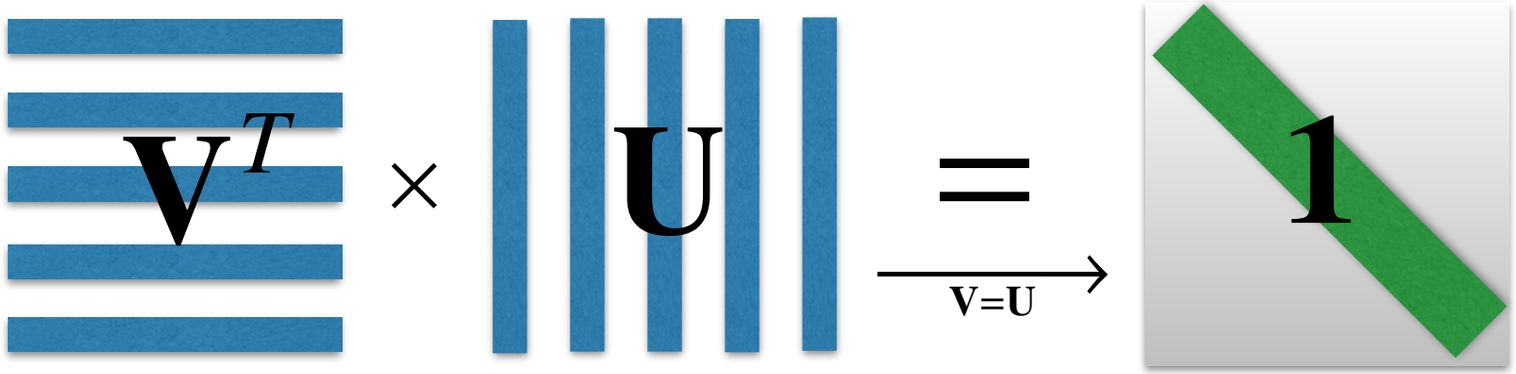
where (assuming that $N > p$):

$\Lambda = diag(\lambda_1, .., \lambda_p)$ $(\lambda_1 \geq ... \geq \lambda_p)$ is a diagonal matrix of ordered eigenvalues

$\mathbf{U}$ is a $N \times N$ orthogonal matrix ($\mathbf{U}\mathbf{U}^T = \mathbf{I}$) that stores eigenvectors column-wise

Use this to compute the covariance matrix of $\mathbf{Z} = \mathbf{X}\mathbf{V}$

$$Cov[\mathbf{Z}] = \frac{1}{N-1}\mathbf{Z}^T\mathbf{Z} = \frac{1}{N-1}\mathbf{V}^T\mathbf{X}^T\mathbf{X}\mathbf{V} = \mathbf{V}^T\mathbf{C}\mathbf{V} = \mathbf{V}^T\mathbf{U}\Lambda\left(\mathbf{V}^T\mathbf{U}\right)^T \xrightarrow[\mathbf{V}=\mathbf{U}]{} \Lambda$$



Z → made of uncorrelated components with adiagonal correlation matrix

PCA as a coordinate transform

Eigenvalue decomposition of the correlation matrix (note that $\mathbf{C}$ is non-negative definite!)

$$\mathbf{C} = \mathbf{U}\Lambda\mathbf{U}^T$$

where (assuming that $N > p$):

$\Lambda = diag(\lambda_1, .., \lambda_p)$ $(\lambda_1 \geq ... \geq \lambda_p)$ is a diagonal matrix of ordered eigenvalues

$\mathbf{U}$ is a $N \times N$ orthogonal matrix $(\mathbf{U}\mathbf{U}^T = \mathbf{I})$ that stores eigenvectors column-wise

Use this to compute the covariance matrix of $\mathbf{Z} = \mathbf{X}\mathbf{V}$

$$Cov[\mathbf{Z}] = \frac{1}{N-1}\mathbf{Z}^T\mathbf{Z} = \frac{1}{N-1}\mathbf{V}^T\mathbf{X}^T\mathbf{X}\mathbf{V} = \mathbf{V}^T\mathbf{C}\mathbf{V} = \mathbf{V}^T\mathbf{U}\Lambda\left(\mathbf{V}^T\mathbf{U}\right)^T \xrightarrow[\mathbf{V}=\mathbf{U}]{} \Lambda$$

Therefore, when $\mathbf{V} = \mathbf{U}$ (and $k = p$), encoding $\mathbf{Z} = \mathbf{X}\mathbf{V}$ **preserves the total variation of data**

$$TotVar[\mathbf{X}] = \frac{1}{N-1}Tr\left[\mathbf{X}^T\mathbf{X}\right] = \mathrm{Tr}[\mathbf{C}] = \mathrm{Tr}\left[\mathbf{U}\Lambda\mathbf{U}^T\right] = \mathrm{Tr}\left[\Lambda\mathbf{U}^T\mathbf{U}\right] = \mathrm{Tr}[\Lambda]$$

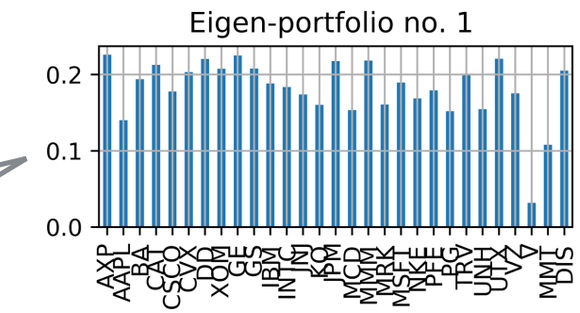
$$TotVar[\mathbf{Z}] = \frac{1}{N-1}Tr\left[\mathbf{Z}^T\mathbf{Z}\right] = \mathrm{Tr}\left[\mathbf{V}^T\mathbf{U}\Lambda\left(\mathbf{V}^T\mathbf{U}\right)^T\right] \xrightarrow[\mathbf{V}=\mathbf{U}]{} \mathrm{Tr}[\Lambda]$$

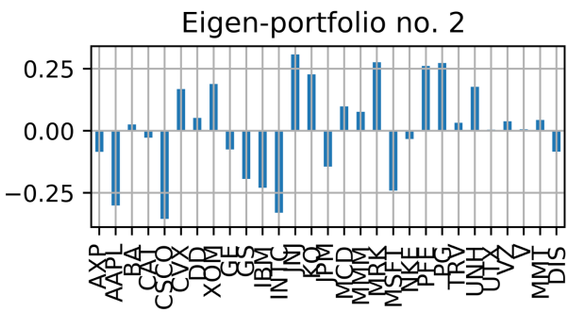
* When $K = p$, we preserve all variance in data by making a linear encoding $Z = XV$

PCA for stocks in DJI: eigenvectors

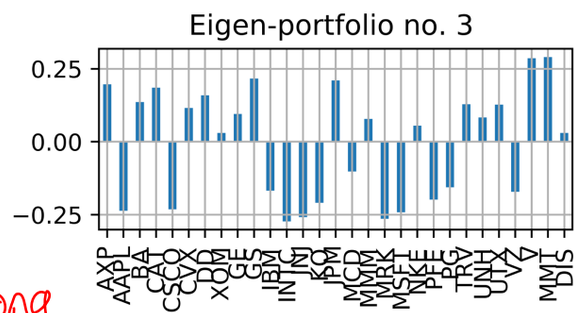
Eigenvectors = weights of stocks in "eigen-portfolios"

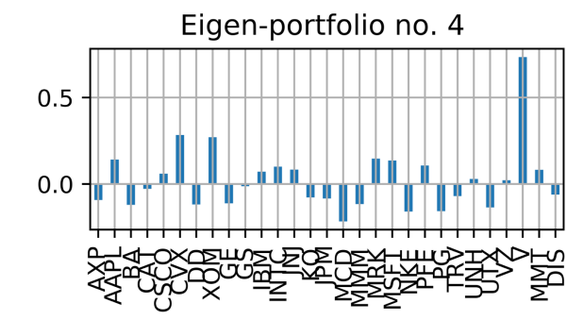
Approximates the market (DJI)

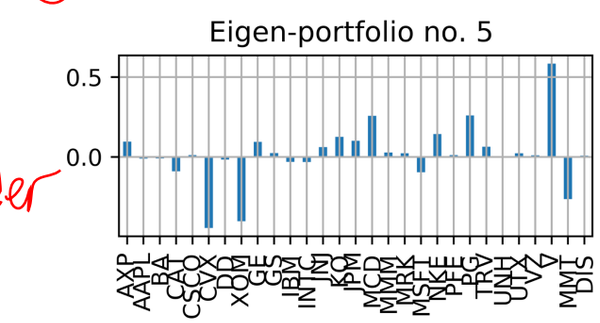


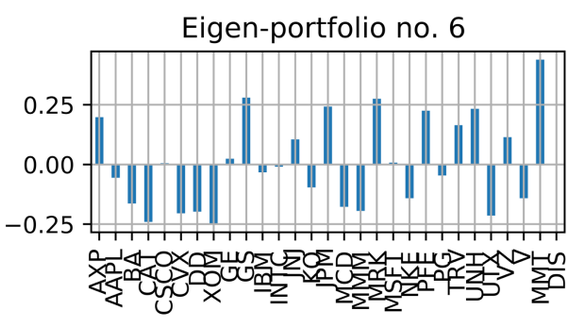


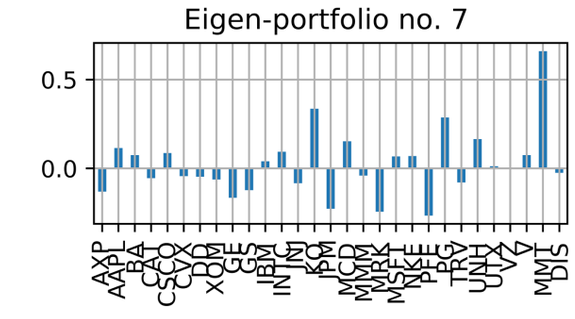
Orthogonal to the market (DJI)

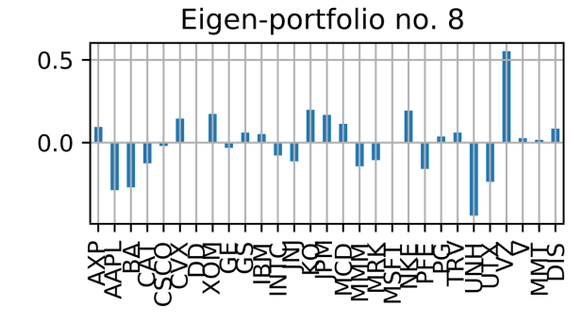












* We reduce our problem of optimal investment among universe of p correlated stocks ⟶ into a simpler one of p uncorrelated eigenportfolios

i.e. uncorrelated to market, and uncorrelated to the rest

uncorrelated ≠ independent

Control question

Select all correct answers

1. The PCA implements a Linear Encoder that converts correlated inputs $\mathbf{X}$ into uncorrelated features $\mathbf{Z}$ by a linear transform $\mathbf{Z} = \mathbf{XV}$
2. If the orthogonal matrix $\mathbf{V}$ has dimension $p \times p$, i.e. it keeps all eigenvectors of correlation matrix of $\mathbf{X}$, then $\mathbf{Z}$ preserves the total variation of $\mathbf{X}$.
3. When the data is noisy or non-stationary, $\mathbf{Z}$ can have a higher variance than $\mathbf{X}$.
4. The PCA is a probabilistic method that enables simulating from data.

Correct answer: 1, 2

Guided Tour of Machine Learning in Finance

Week 3: Unsupervised Learning

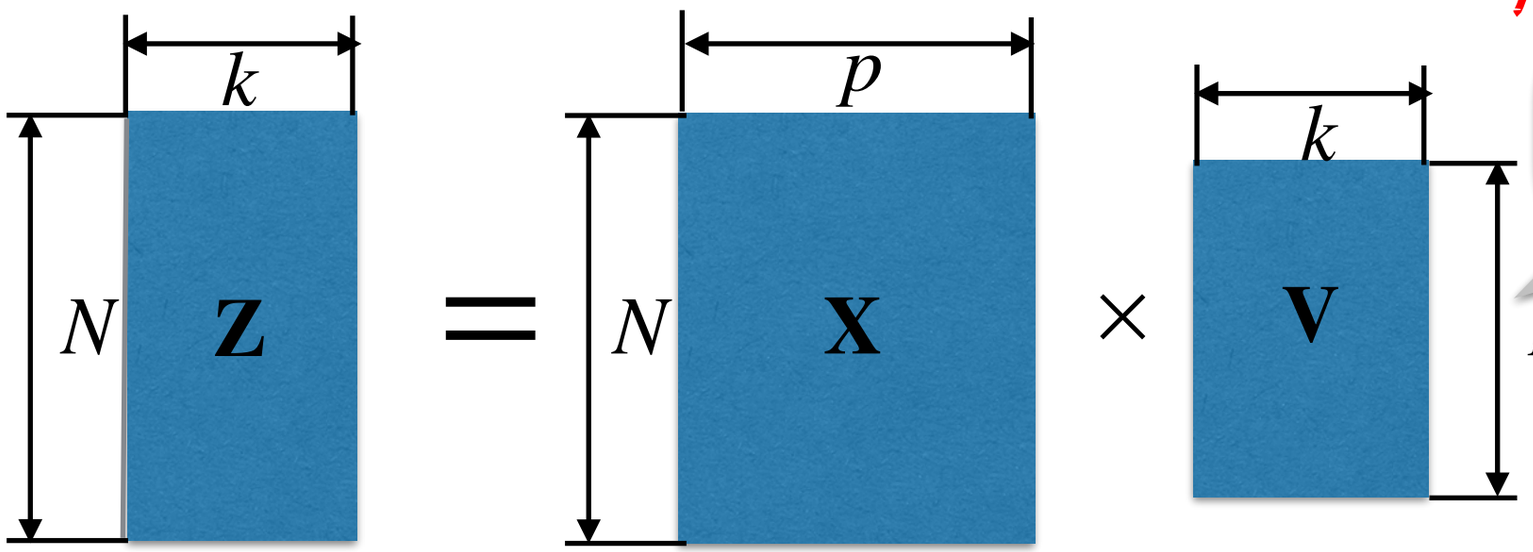
Dimension reduction with the PCA

Igor Halperin

NYU Tandon School of Engineering, 2017

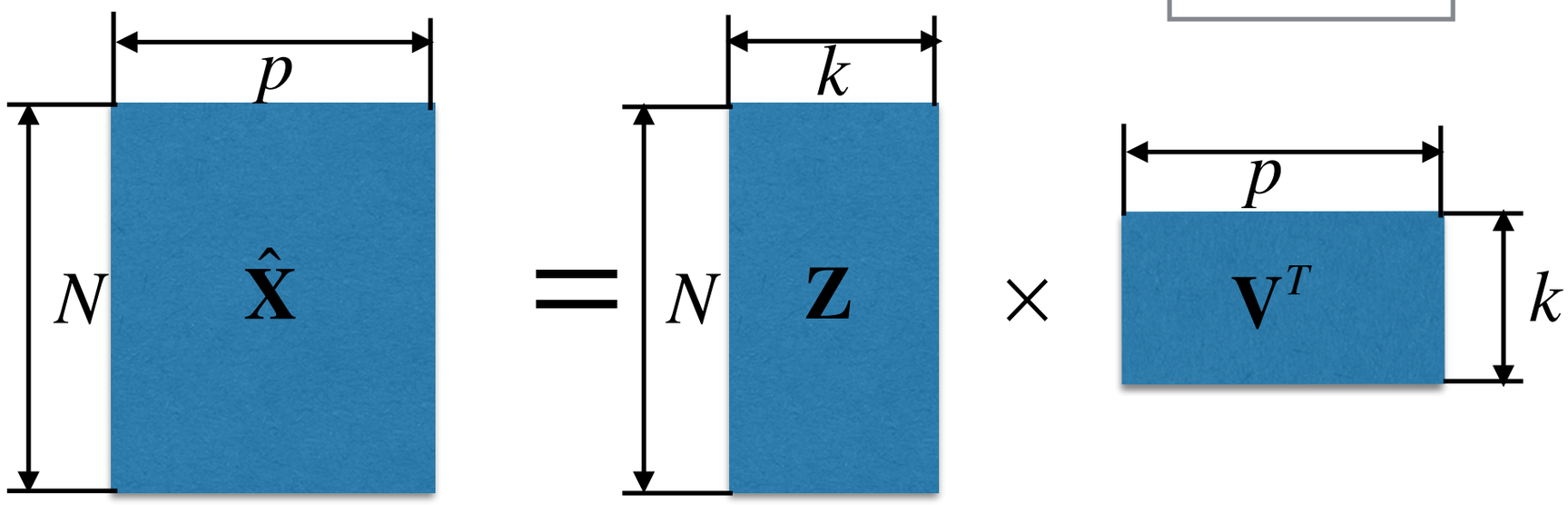
PCA as a dimension reduction method

We had a linear transform (linear encoder) of the data $\mathbf{Z} = \mathbf{XV}$ parametrized by a $p \times k$ orthogonal matrix $\mathbf{V}$ with $\mathbf{VV}^T = 1$



If $k = p$, $\mathbf{V} = \mathbf{U}$ this is just a rotation of the eigenvalue decomposition

A **decoded signal** (inverse transform) is obtained as $\hat{\mathbf{X}} = \mathbf{ZV}^T$



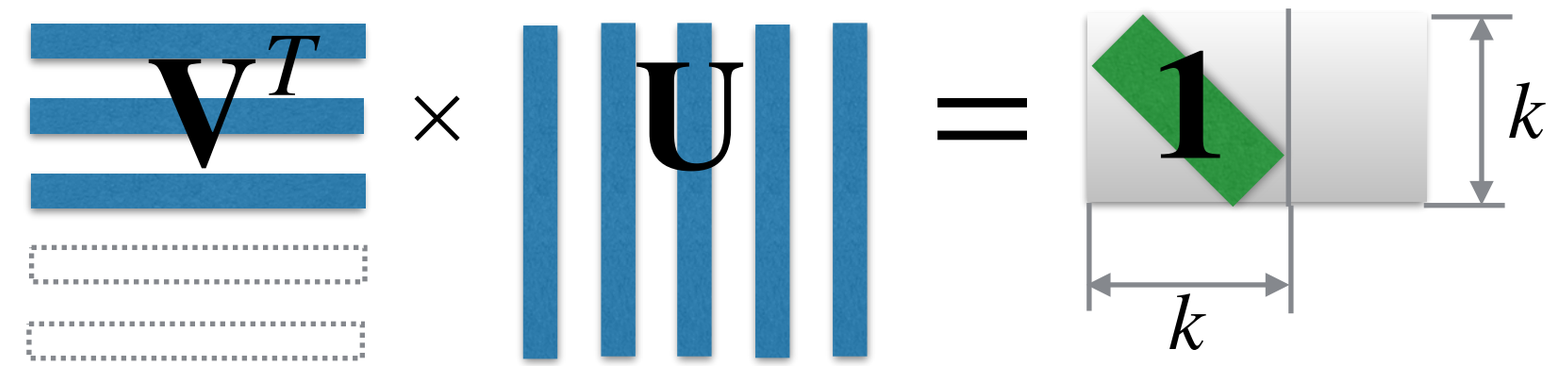
Dimension reduction with PCA

Dimension reduction: make the projection matrix $\mathbf{V}$ of the first $k \le p$ eigenvectors of $\mathbf{C}$

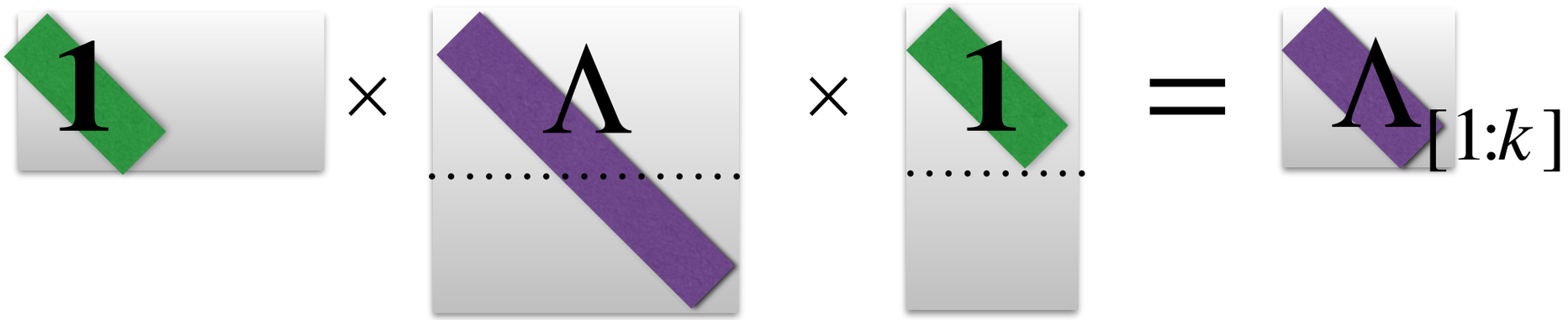
$$\mathbf{V} = \mathbf{U}^{[1:k]}$$

Use this to compute the covariance matrix of $\mathbf{Z} = \mathbf{XV}$

$$Cov[\mathbf{Z}] = \frac{1}{N-1}\mathbf{Z}^T\mathbf{Z} = \frac{1}{N-1}\mathbf{V}^T\mathbf{X}^T\mathbf{XV} = \mathbf{V}^T\mathbf{CV} = \mathbf{V}^T\mathbf{U}\Lambda\left(\mathbf{V}^T\mathbf{U}\right)^T \xrightarrow[\mathbf{V}=\mathbf{U}^{[1:k]}]{} \Lambda_{[1:k]}$$



$$\Leftarrow \mathbf{W_{ij}} = \mathbf{U_i^T U_j} = \delta_{ij},$$
$$i = 1,\ldots,k, \quad j = 1,\ldots,p$$



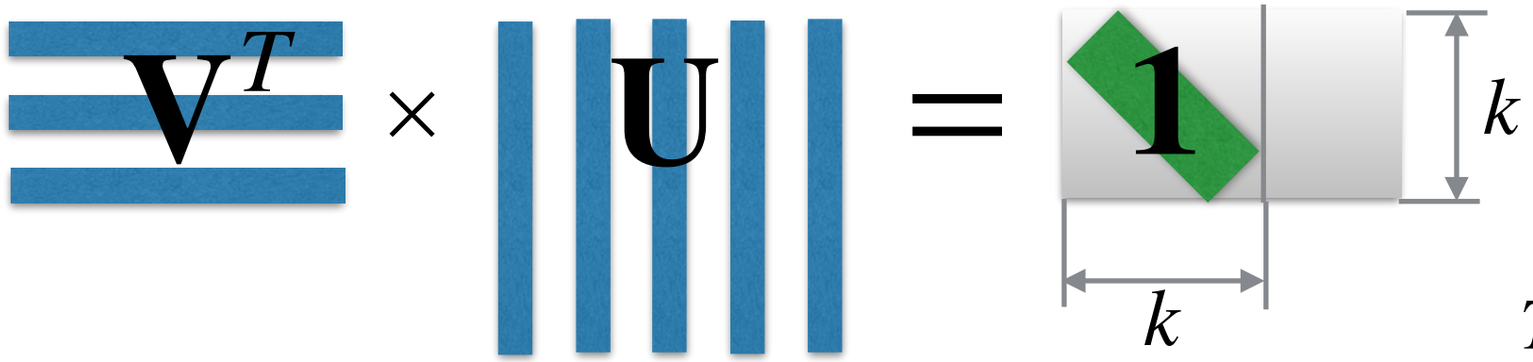
Dimension reduction with PCA

Dimension reduction: make the projection matrix $\mathbf{V}$ of the first $k \le p$ eigenvectors of $\mathbf{C}$

$$\mathbf{V} = \mathbf{U}^{[1:k]}$$

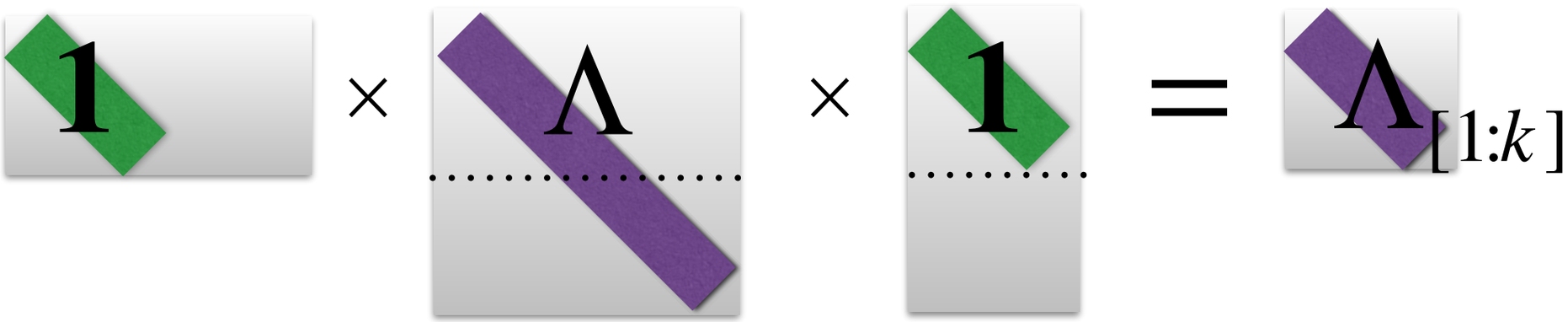
Use this to compute the covariance matrix of $\mathbf{Z} = \mathbf{XV}$

$$Cov[\mathbf{Z}] = \frac{1}{N-1}\mathbf{Z}^T\mathbf{Z} = \frac{1}{N-1}\mathbf{V}^T\mathbf{X}^T\mathbf{XV} = \mathbf{V}^T\mathbf{CV} = \mathbf{V}^T\mathbf{U}\Lambda\left(\mathbf{V}^T\mathbf{U}\right)^T \xrightarrow[\mathbf{V}=\mathbf{U}^{[1:k]}]{} \Lambda_{[1:k]}$$



* Only a part of total variation of $\mathbf{X}$ is preserved:

$$TotVar[\mathbf{Z}] = \mathrm{Tr}\left[\Lambda_{[1:k]}\right] = \sum_{i=1}^{k}\lambda_i$$



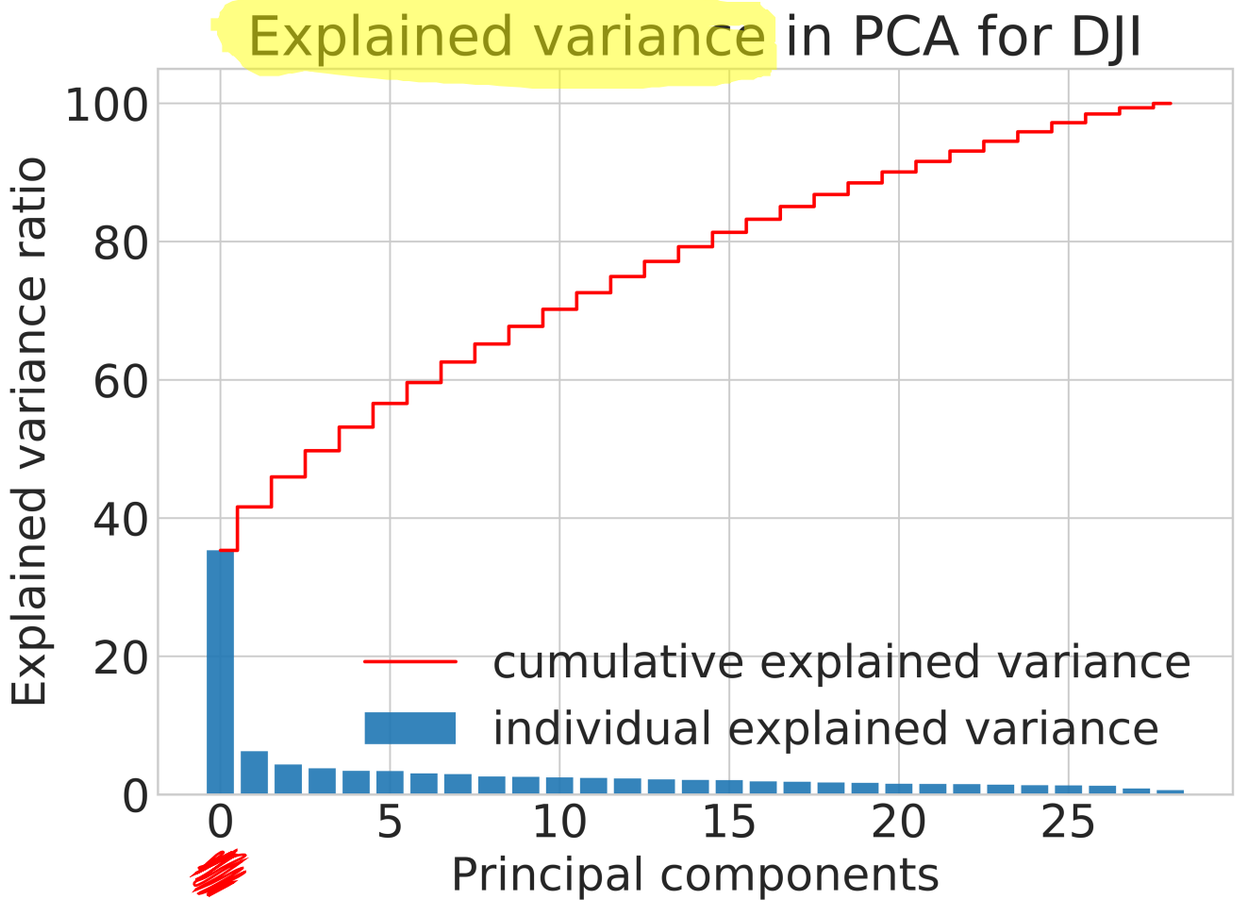
Variance explained by PCA

Dimension reduction: make the projection matrix $\mathbf{V}$ of the first $k \leq p$ eigenvectors of $\mathbf{C}$

$$\mathbf{V} = \mathbf{U}^{[1:k]}$$

Use this to compute the covariance matrix of $\mathbf{Z} = \mathbf{XV}$

$$Cov[\mathbf{Z}] = \frac{1}{N-1}\mathbf{Z}^T\mathbf{Z} = \frac{1}{N-1}\mathbf{V}^T\mathbf{X}^T\mathbf{XV} = \mathbf{V}^T\mathbf{CV} = \mathbf{V}^T\mathbf{U}\Lambda\left(\mathbf{V}^T\mathbf{U}\right)^T \xrightarrow[\mathbf{V}=\mathbf{U}^{[1:k]}]{} \Lambda_{[1:k]}$$



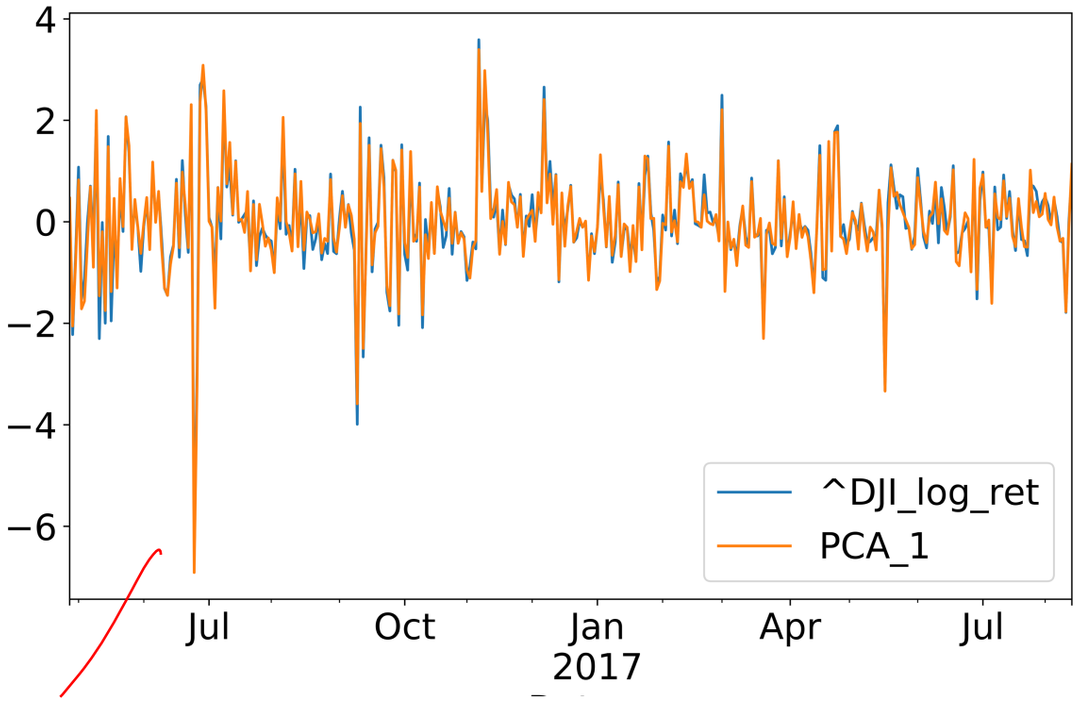
Only a part of total variation of $\mathbf{X}$ is preserved:

$$TotVar[\mathbf{Z}] = \mathrm{Tr}\left[\Lambda_{[1:k]}\right] = \sum_{i=1}^{k}\lambda_i$$

$$\frac{\sum_{1}^{k}\sigma}{\sum_{1}^{N}\sigma}\ \%$$

PCA projections vs the market

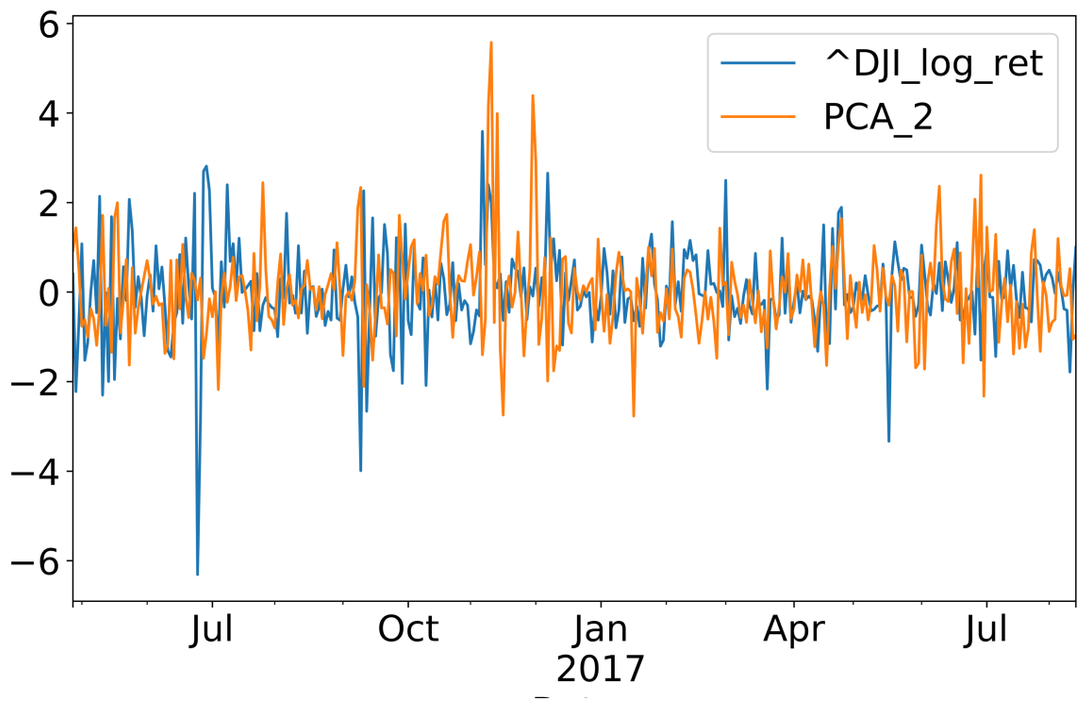
Compare the actual returns on the DJI index with returns of the two leading eigen-portfolios



PC 1

large values of returns exhibit obvious mismatch

Correlated with the market



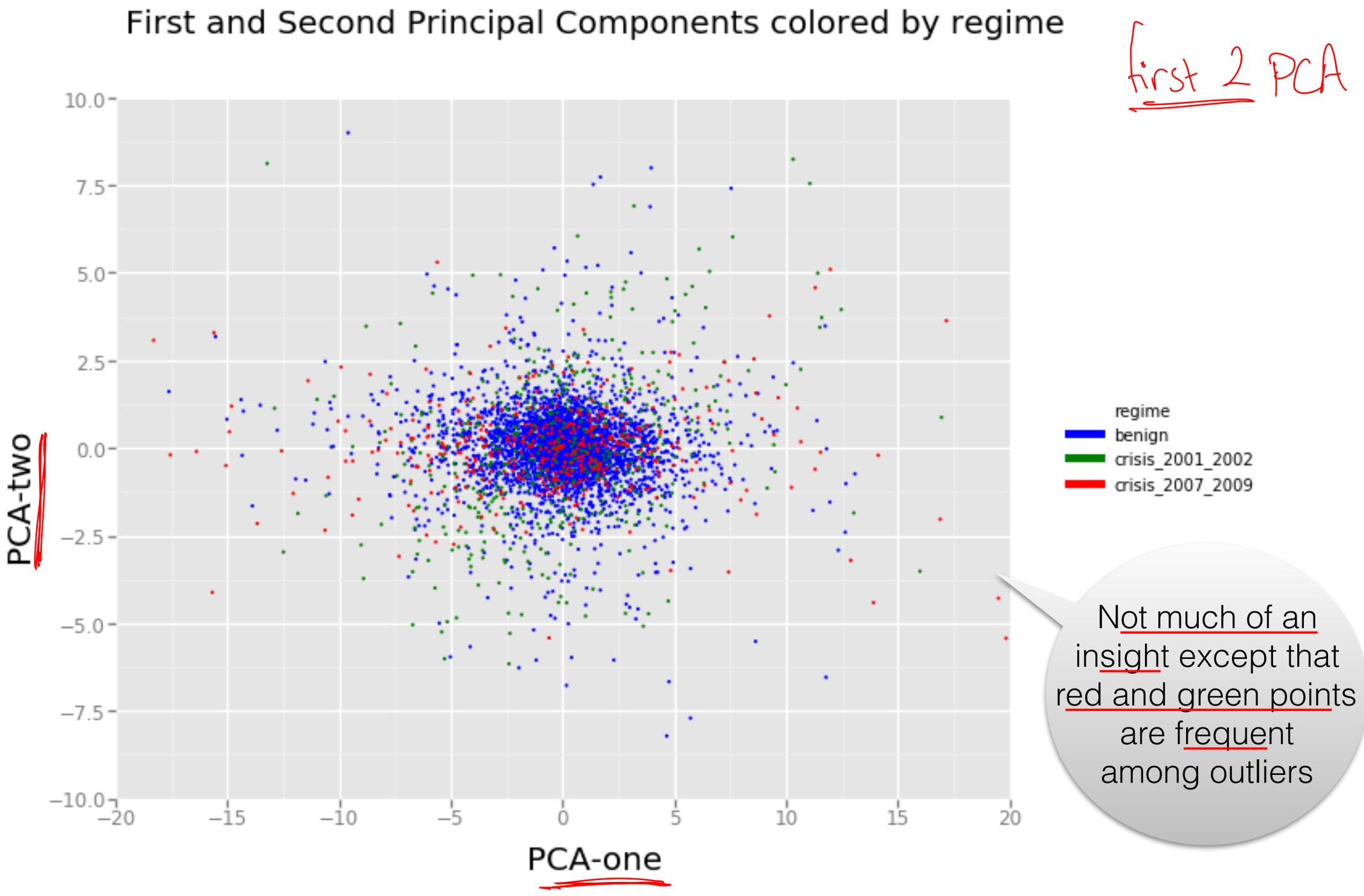
PC 2

much more mismatch

Uncorrelated with the market

PCA projections: 2D visualization

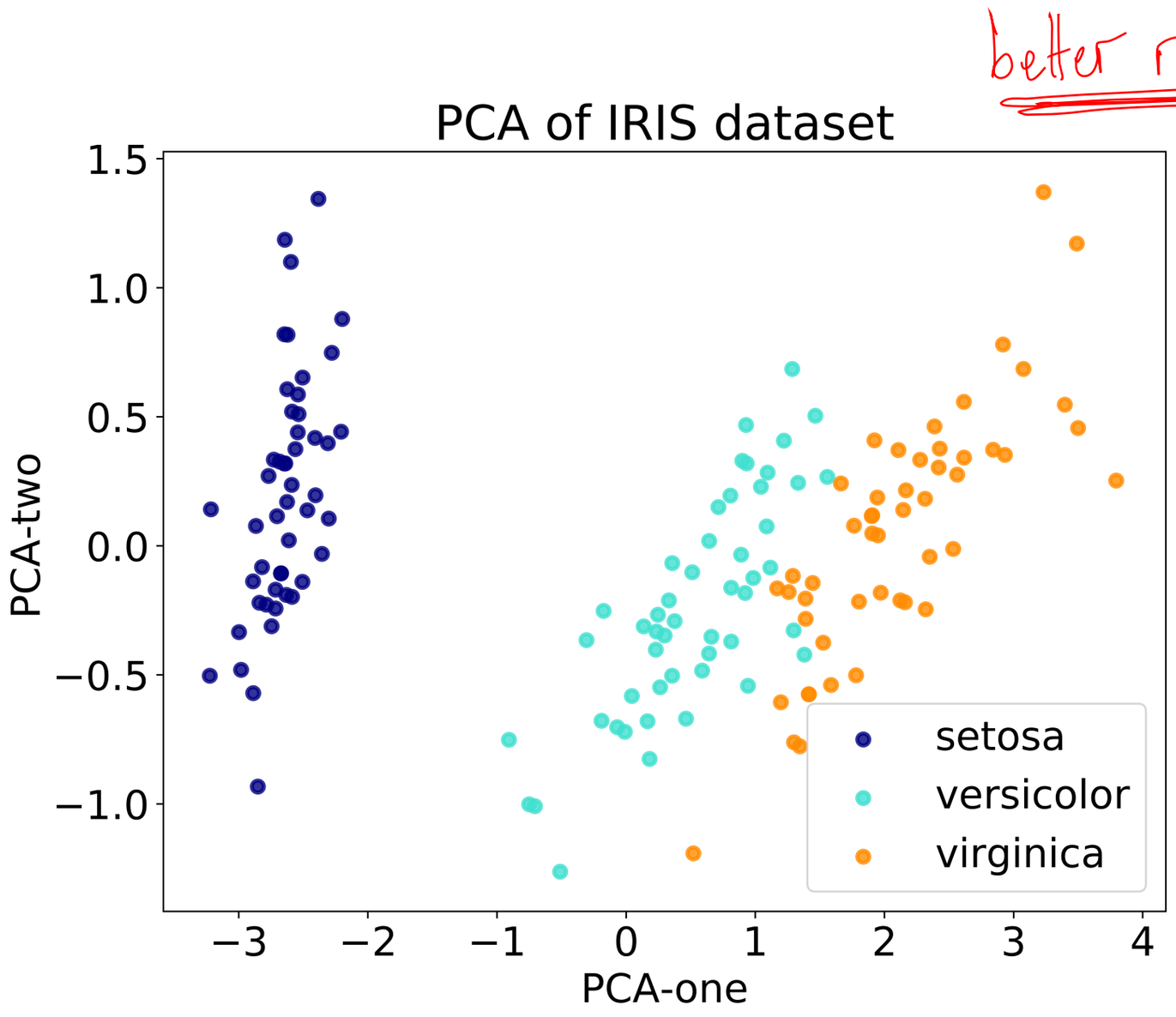
Compare the actual returns on the DJI index with returns of the two leading eigen-portfolios



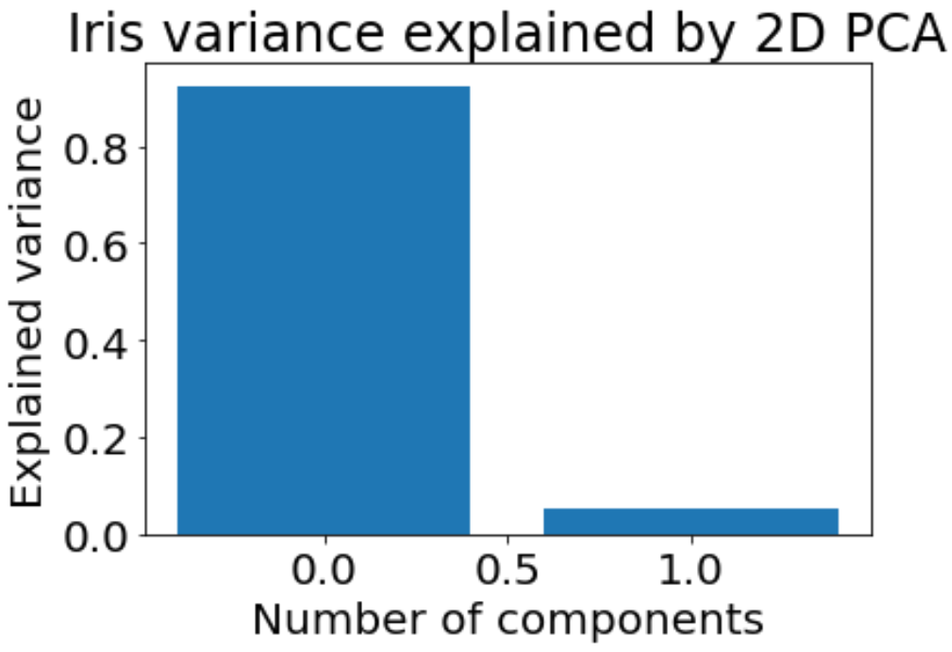
PCA 2D visualization for the Iris dataset

Compare with the 2D PCA projection for the Iris dataset: 3 types of Iris, 4 features: Sepal Length, Sepal Width, Petal Length and Petal Width, 150 examples, see http://scikit-learn.org/stable/auto_examples/datasets/plot_iris_dataset.html.

better results



Different Iris types are well separated by a 2D projection



Quizz: Why does the 2D PCA perform worse for stocks than for Iris?

1. Because stock prices are produced by human interactions, while variability of the Iris species is produced by nature
2. The Iris data is stationary, and is more homogeneous/has less outliers than the stock data
3. Smaller dimension of the feature space
4. Stronger correlations between features
5. The labels are real for Iris (they are Iris' types), but "noisy" for the "regime" label in the stock analysis

Multiple choice answers:

1. Answer 1
2. Answer 3
3. Answers 2 and 4
4. Answer 2 and 5
5. Answers 3, 4, 5
6. Answers 2, 3, 4, 5
7. All of the above

Correct answer: 6

PCA is linear method, any non-linearity will not be captured

2D PCA for stocks vs Iris

Why does the 2D PCA perform much worse for stocks than for Iris?

- Smaller dimension of the feature space
- Stronger correlations between the features (the first PC explains 92% of total variance!)
- The Iris data is stationary, and is more homogeneous/has less outliers than the stock data
- The labels are real for Iris (they are Iris' types), but "noisy" for the "regime" label in the stock analysis

* PCA is expected to work well for problems that are largely determined by lowest moments of distributions i.e mean & covariance

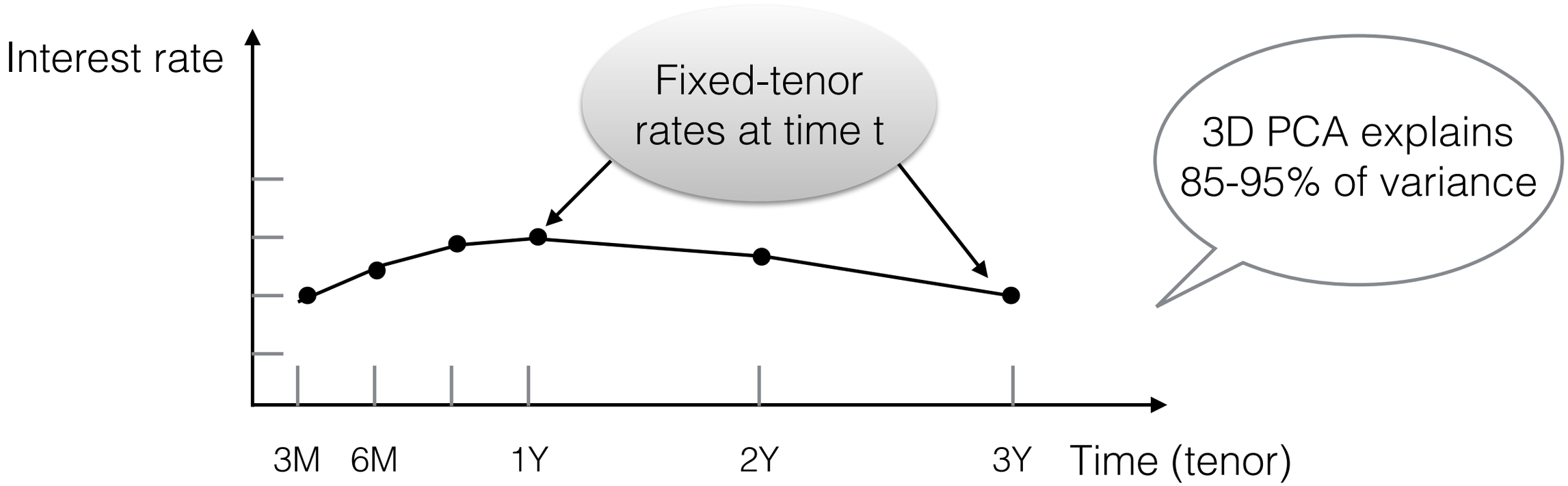
Higher moments like skewness & kurtosis won't.

Such app rely on analysis of the total variance in the data versus variance explained by first principal components.

Financial applications of the PCA

Numerous financial applications of the PCA:

- Systematic trading: construction of market-neutral portfolios with low volatility, extraction of trading signals
- Systemic risk detection methods (e.g. M. Kritzman et. al. “Principal Components as a Measure of Systemic Risk”, 2010)
- Risk management: sector risk exposure management of trading book or banking book portfolios, de-noising of empirical correlation matrices
- 2D PCA for data visualization
- PCA for yield curve modeling (fixed income, commodities)



Guided Tour of Machine Learning in Finance

Week 3: Unsupervised Learning

Dimension reduction and data visualization with t-SNE

Igor Halperin

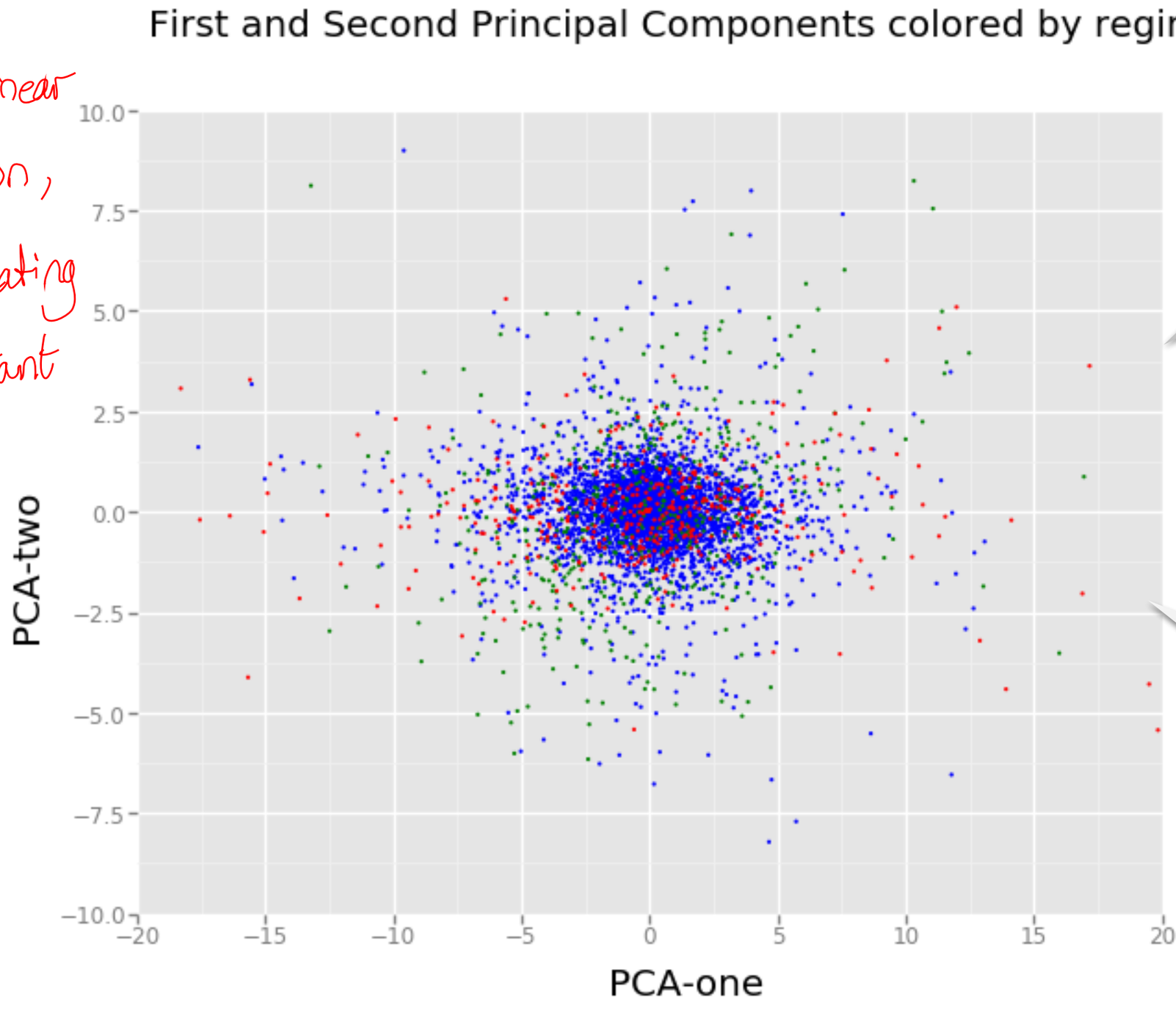
NYU Tandon School of Engineering, 2017

2D visualization: what can be improved?

Can we find a better 2D visualization of the DJI stock returns than the 2D PCA?

PCA

relies on linear transformation, while truncating less important dim

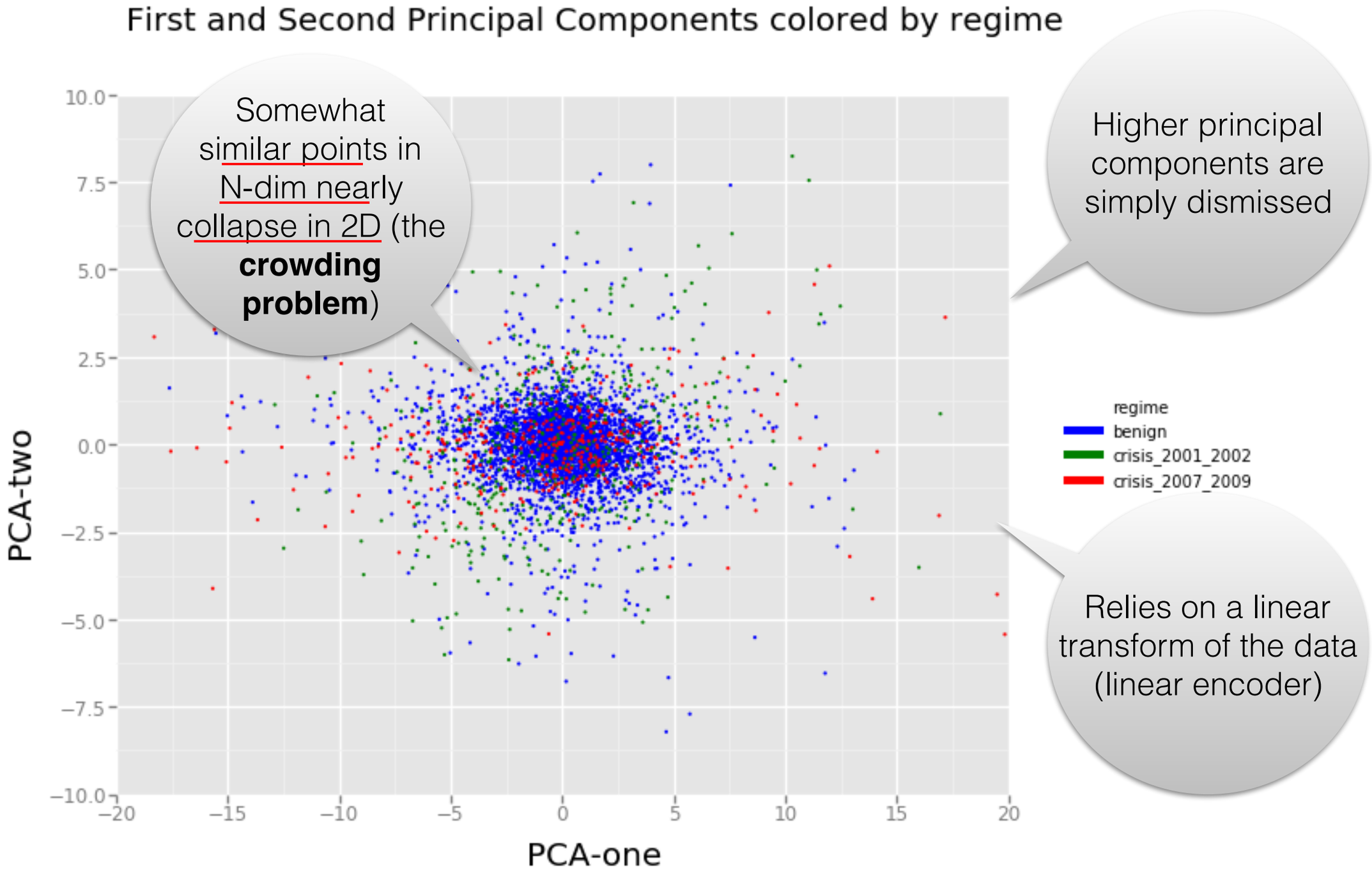


Higher principal components are simply dismissed

Relies on a linear transform of the data (linear encoder)

2D visualization: the “crowding problem”

Can we find a better 2D visualization of the DJI stock returns than the 2D PCA?



Quiz: What do you think is the origin of the crowding problem?

1. The crowding problem is specific to stock analyses because of high level of noise in financial data
2. Two dimensions are just not enough to explain the amount of variation in multi-dimensional data
3. When you project multi-dimensional data on a plane, the need to display outliers forces non-outliers to crowd together
4. Because in higher dimensions there are typically more outliers than in lower dimensions, the crowding problem becomes worse with increased dimensions
5. A uniform distribution in an N-dimensional hyper-cube of size r has the number of points proportional to the volume of the hyper-cube $V \sim r^N$, therefore a *fixed* number of uniformly distributed observed points in N dimensions will generally translate into a *non-uniform* distributions of their projections onto a 2D plan.

Multiple choice answers:

1. Answer 1
2. Answer 2
3. Answers 3 and 4
4. Answer 5
5. Answers 3, 4, 5
6. All of the above

Quiz: What do you think is the origin of the crowding problem?

1. The crowding problem is specific to stock analyses because of high level of noise in financial data
2. Two dimensions are just not enough to explain the amount of variation in multi-dimensional data
3. When you project multi-dimensional data on a plane, the need to display outliers forces non-outliers to crowd together
4. Because in higher dimensions there are typically more outliers than in lower dimensions, the crowding problem becomes worse with increased dimensions
5. A uniform distribution in an N-dimensional hyper-cube of size r has the number of points proportional to the volume of the hyper-cube $V \sim r^N$, therefore a *fixed* number of uniformly distributed observed points in N dimensions will generally translate into a *non-uniform* distributions of their projections onto a 2D plan.

Multiple choice answers:

1. Answer 1
2. Answer 2
3. Answers 3 and 4
4. Answer 5
5. Answers 3, 4, 5
6. All of the above

Correct answer: 5

tSNE vs PCA

t-SNE: **t-distributed Stochastic Neighbor Embedding** (van der Maaten and Hinton 2008)

	PCA	t-SNE
Type of algorithm	Deterministic	Stochastic
Projection onto a low dimensional space	Linear	Non-linear
Global or local approach?	Global	Local/global
Handling discarded dimensions	Hard truncation of extra PCA dimensions	No truncation of dimensions
Unique solution?	Yes	No
Interpretability of results	Straightforward (PCA is just a rotation of axes…)	Subjective

matching 2 distributions ← data
projection

Probabilistic dimension reduction with t-SNE

1. Define **similarity** of point $\mathbf{x}_j \in \mathbb{R}^D$ to point $\mathbf{x}_i \in \mathbb{R}^D$ by assuming that it is generated by a **Gaussian distribution** centered at $\mathbf{x}_i$

$$p_{j|i} = \frac{\exp\left(-\left\|\mathbf{x_i} - \mathbf{x_j}\right\|^2 / (2\sigma_\mathbf{i}^2)\right)}{\sum_{k \neq i} \exp\left(-\left\|\mathbf{x_i} - \mathbf{x_j}\right\|^2 / (2\sigma_\mathbf{i}^2)\right)}$$

and define the **joint probability** as $p_{ij} = \left(p_{j|i} + p_{i|j}\right)/\left(2n\right)$ (n is the total number of data points) - guaranteed to be between 0 and 1. Data-specific variances σ_i^2 will be specified later…

2. Assume that their projections onto a 2D plane are samples from a **Student t-distribution** with one degree of freedom (= the Laplace distribution)

$$q_{ij} = \frac{\left(1 + \left\|\mathbf{y}_i - \mathbf{y}_j\right\|^2\right)^{-1}}{\sum_{k \neq i} \left(1 + \left\|\mathbf{y}_k - \mathbf{y}_j\right\|^2\right)^{-1}}$$

A fat-tailed distribution

Large distances are suppressed as a power law

Probabilistic dimension reduction with t-SNE

3. Fix the data-dependent variances σ_i^2 from the requirement that distributions of points $\mathbf{x}_j$ centered around $\mathbf{x}_i$ have a fixed perplexity (specified by the user!. Perplexity controls the effective number of neighbors for each point.)

$$Perp(\mathbf{P}_i) = 2^{H(\mathbf{P}_i)}, \quad H(\mathbf{P}_i) = -\sum_j p_{j|i} \log_2 p_{j|i}$$

4. Use the Kullback-Leibler (KL) divergence to quantify the collective dissimilarity between the set of points in D dimensions with their projections onto a plane:

$$C = KL(P \| Q) = \sum_i \sum_j \mathrm{p}_{ij} \log \frac{p_{ij}}{q_{ij}}$$

5. Optimize C with respect to the choice of points $\{\mathbf{y}_i\}$ using a gradient descent method, with the following value of the gradient:

$$\frac{\partial C}{\partial y_i} = 4 \sum_j \left(p_{ij} - q_{ij} \right)(y_i - y_j), \quad i = 1, \ldots, n$$

The basic t-SNE algorithm

Laurens van der Maaten and Geoffrey Hinton, “Visualizing Data using t-SNE”, Journal of Machine Learning Research 9 (2008), 2579-2605.

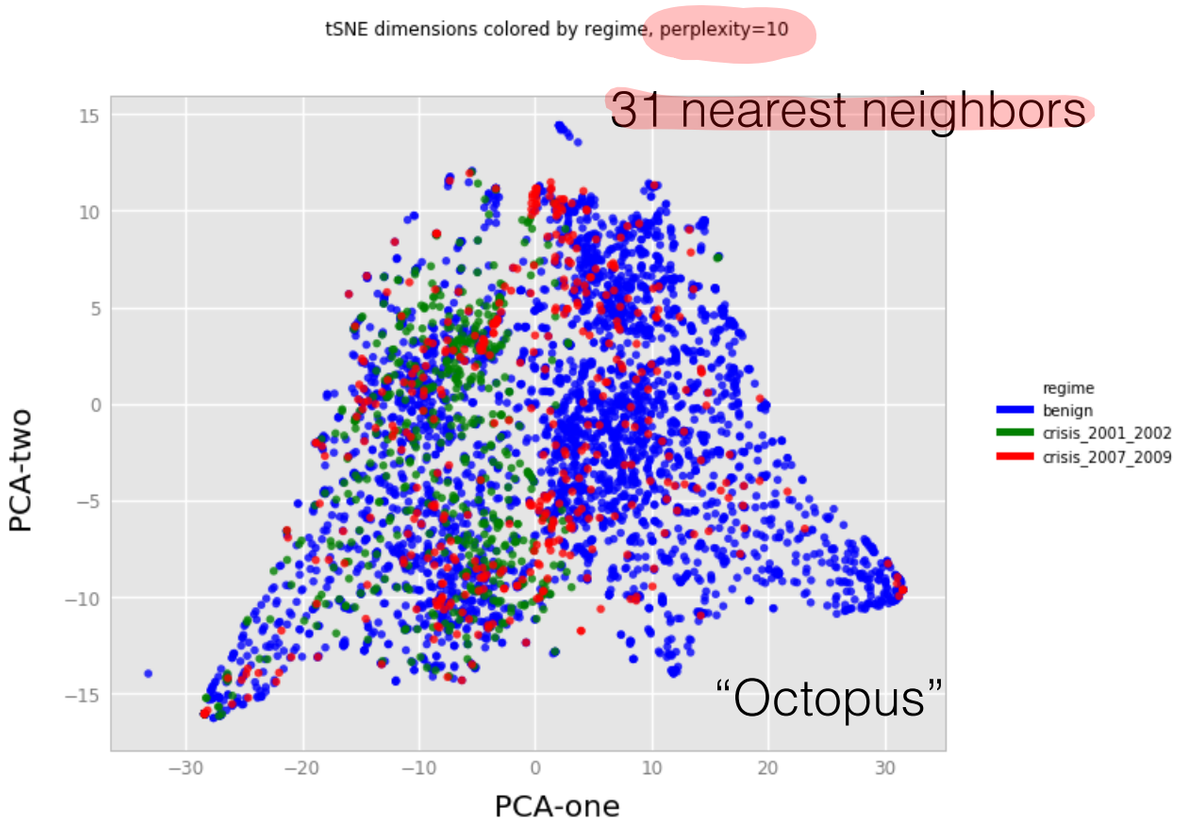
Algorithm 1: Simple version of t-Distributed Stochastic Neighbor Embedding.

Data: data set $\mathcal{X} = \{x_1, x_2, ..., x_n\}$,
cost function parameters: perplexity $Perp$,
optimization parameters: number of iterations T, learning rate η, momentum $\alpha(t)$.
Result: low-dimensional data representation $\mathcal{Y}^{(T)} = \{y_1, y_2, ..., y_n\}$.
begin
  compute pairwise affinities $p_{j|i}$ with perplexity $Perp$ (using Equation 1)
  set $p_{ij} = \frac{p_{j|i} + p_{i|j}}{2n}$
  sample initial solution $\mathcal{Y}^{(0)} = \{y_1, y_2, ..., y_n\}$ from $\mathcal{N}(0, 10^{-4}I)$
  for *t=1* **to** T **do**
    compute low-dimensional affinities q_{ij} (using Equation 4)
    compute gradient $\frac{\delta C}{\delta \mathcal{Y}}$ (using Equation 5)
    set $\mathcal{Y}^{(t)} = \mathcal{Y}^{(t-1)} + \eta \frac{\delta C}{\delta \mathcal{Y}} + \alpha(t)\left(\mathcal{Y}^{(t-1)} - \mathcal{Y}^{(t-2)}\right)$
  end
end

Momentum term

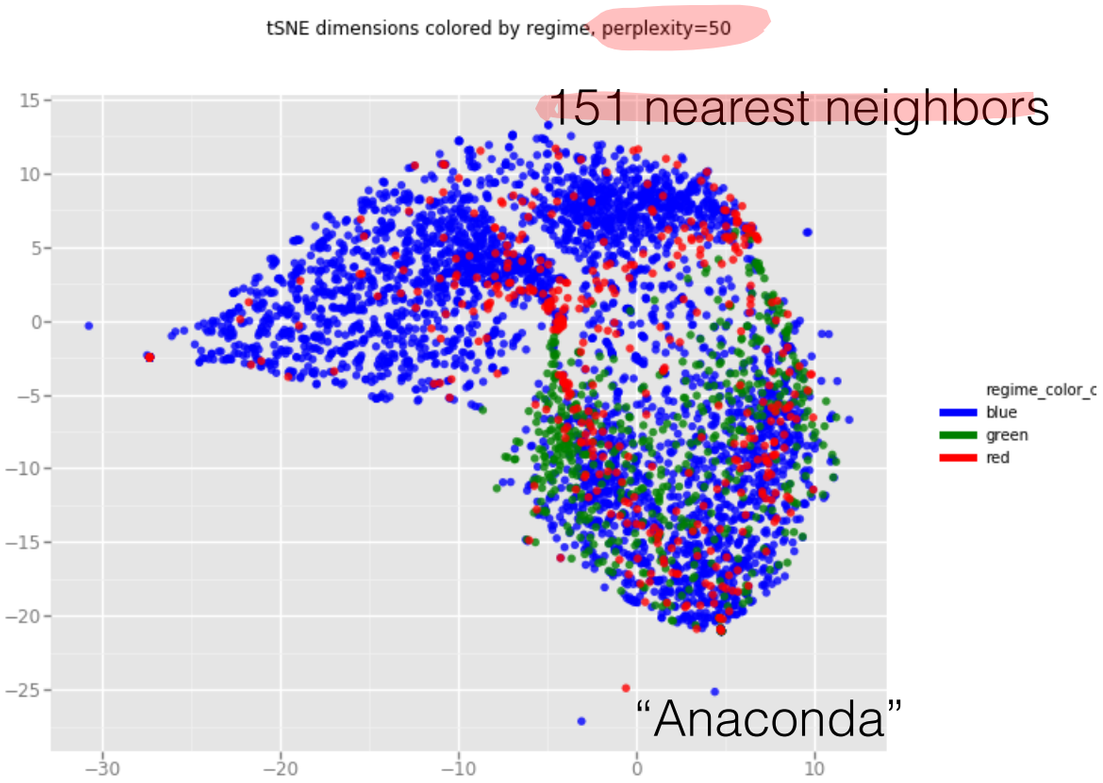
t-SNE for stock returns

Apply t-SNE to the DJI stock return data, for different values of perplexity



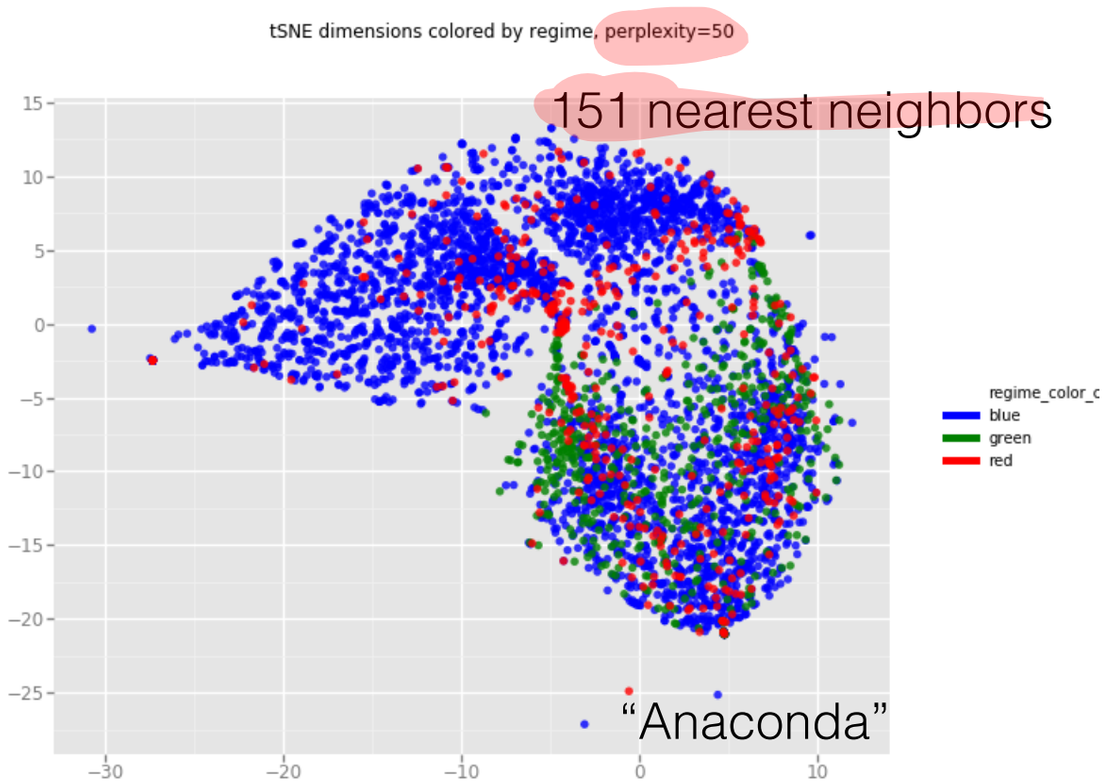
t-SNE for stock returns

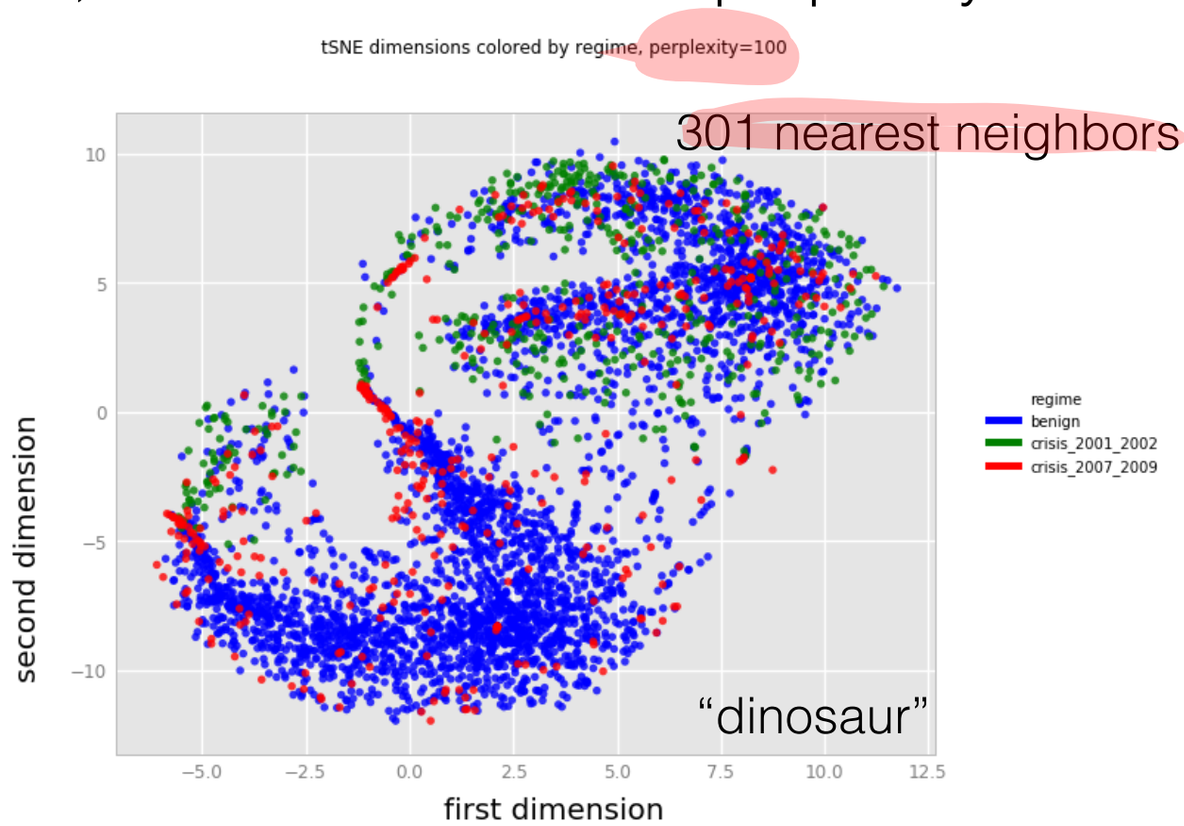
Apply t-SNE to the DJI stock return data, for different values of perplexity



t-SNE for stock returns

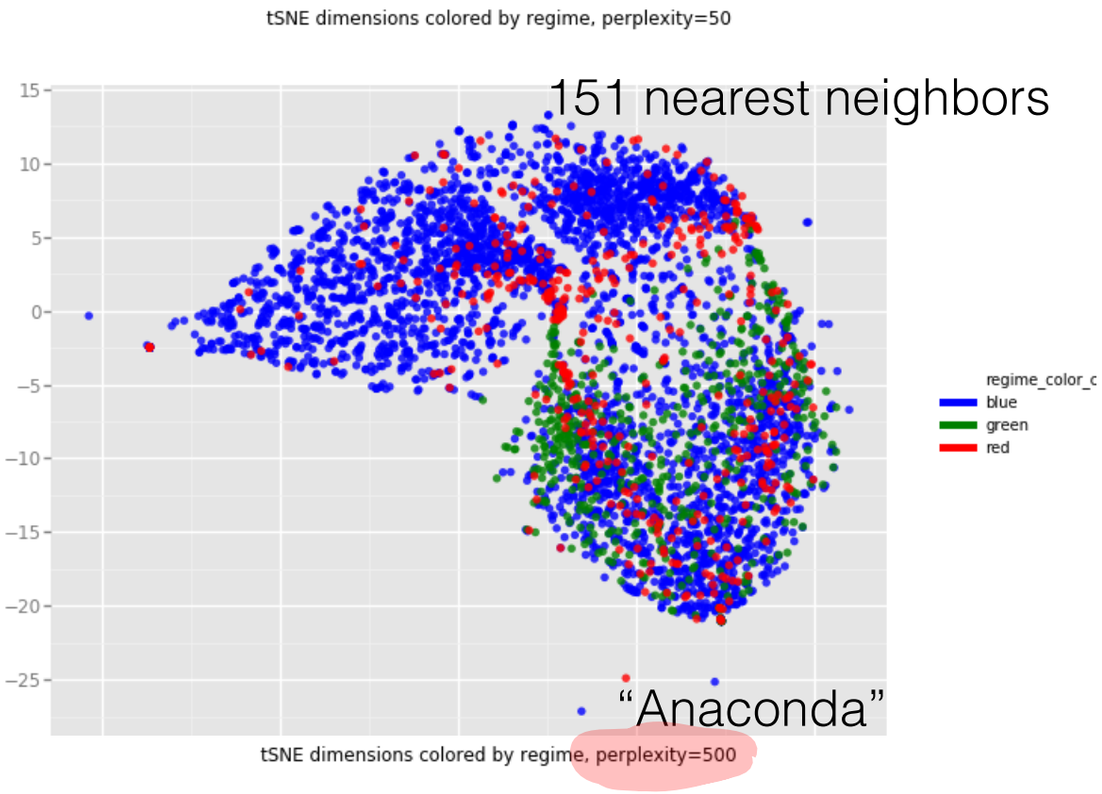
Apply t-SNE to the DJI stock return data, for different values of perplexity

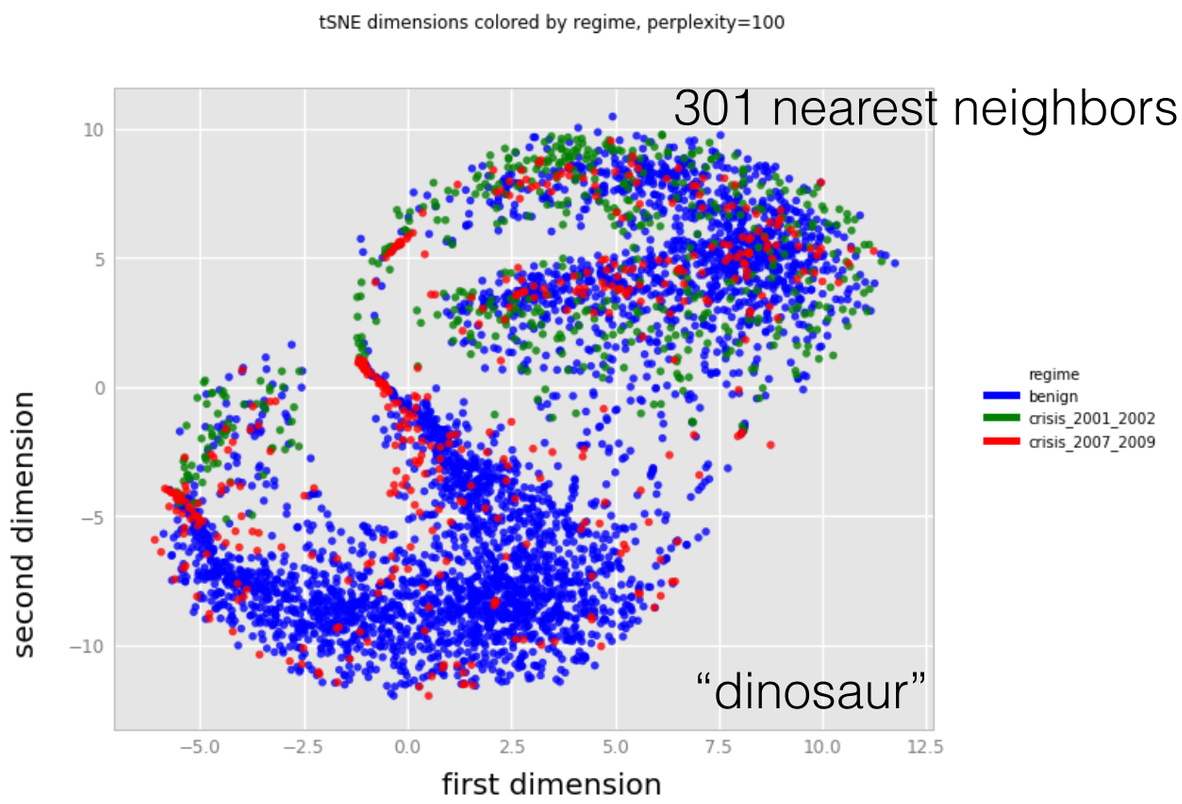


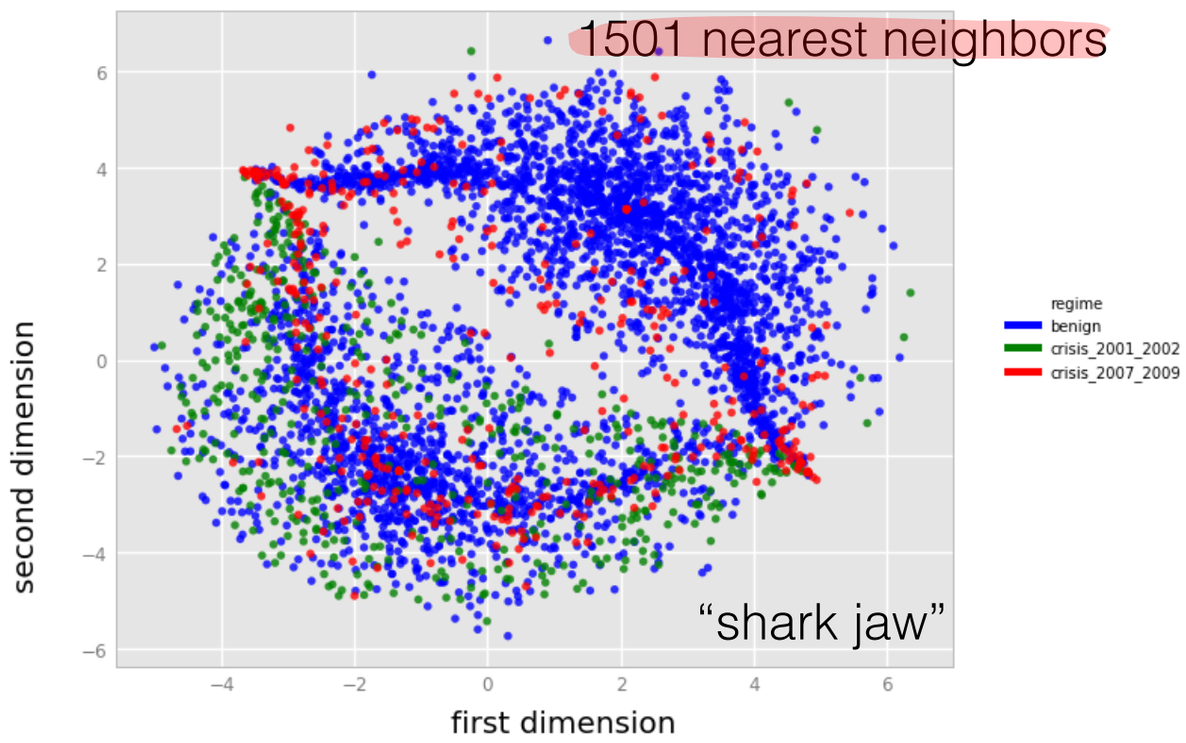


t-SNE for stock returns

Apply t-SNE to the DJI stock return data, for different values of perplexity

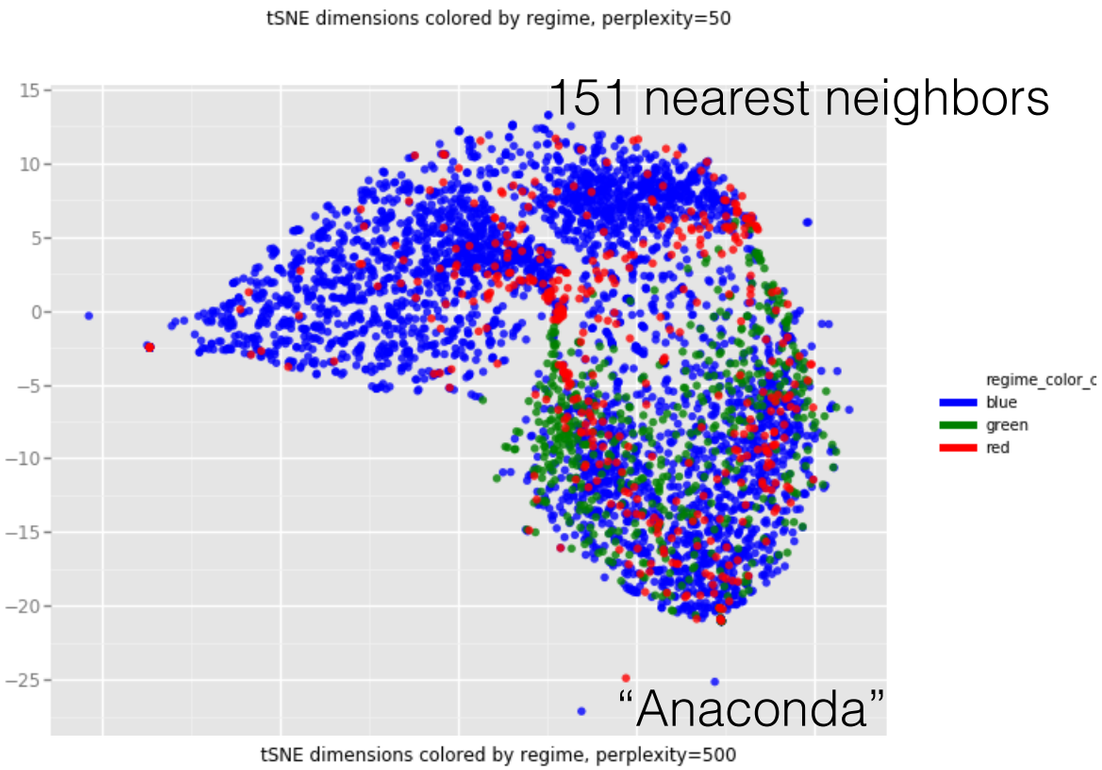


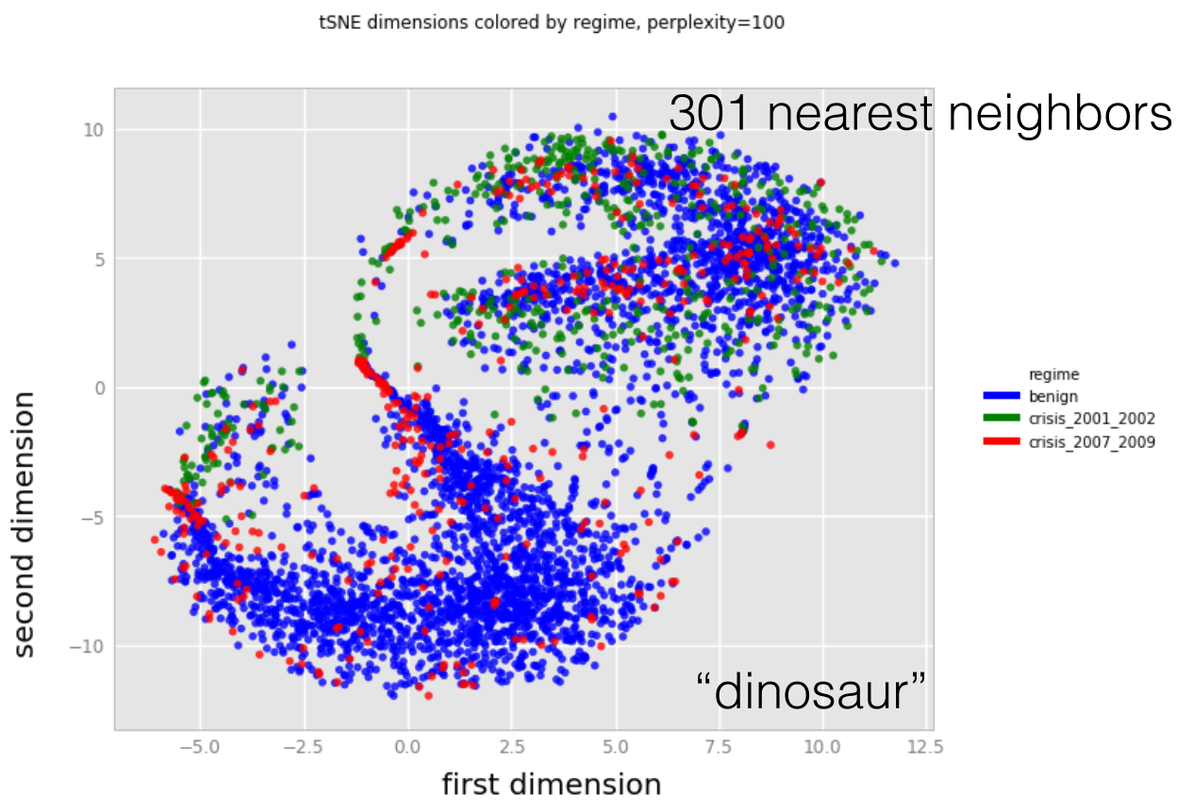


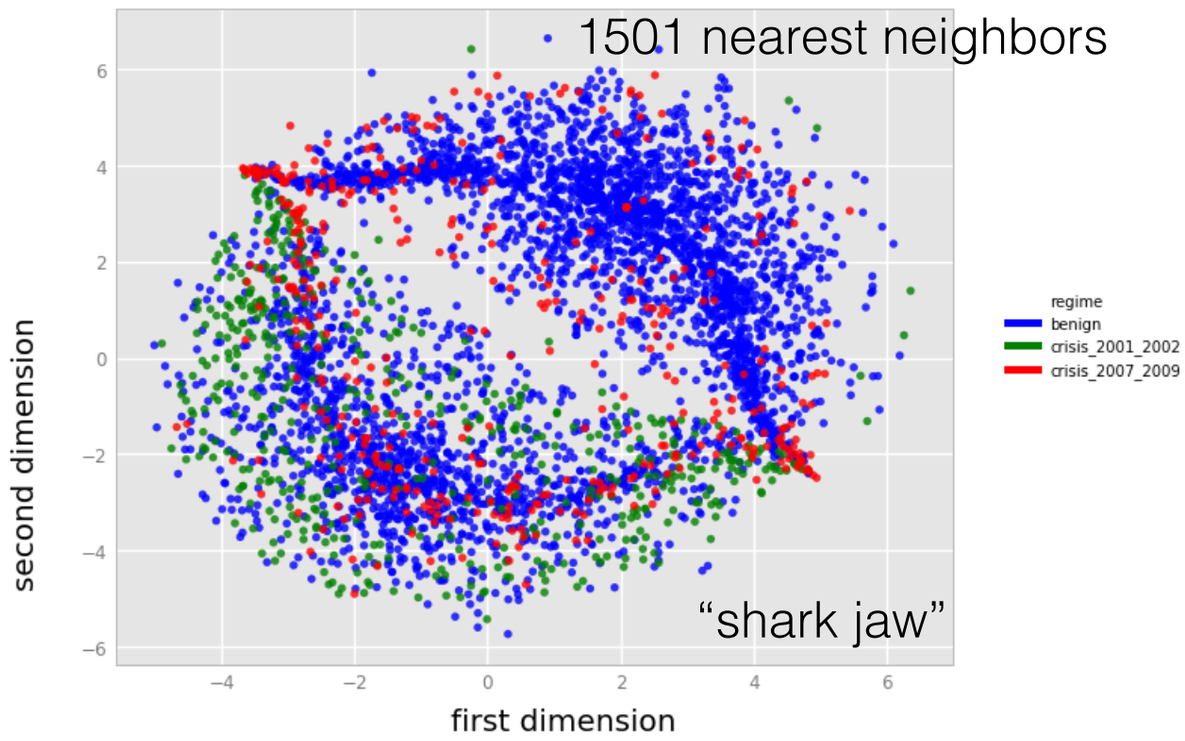


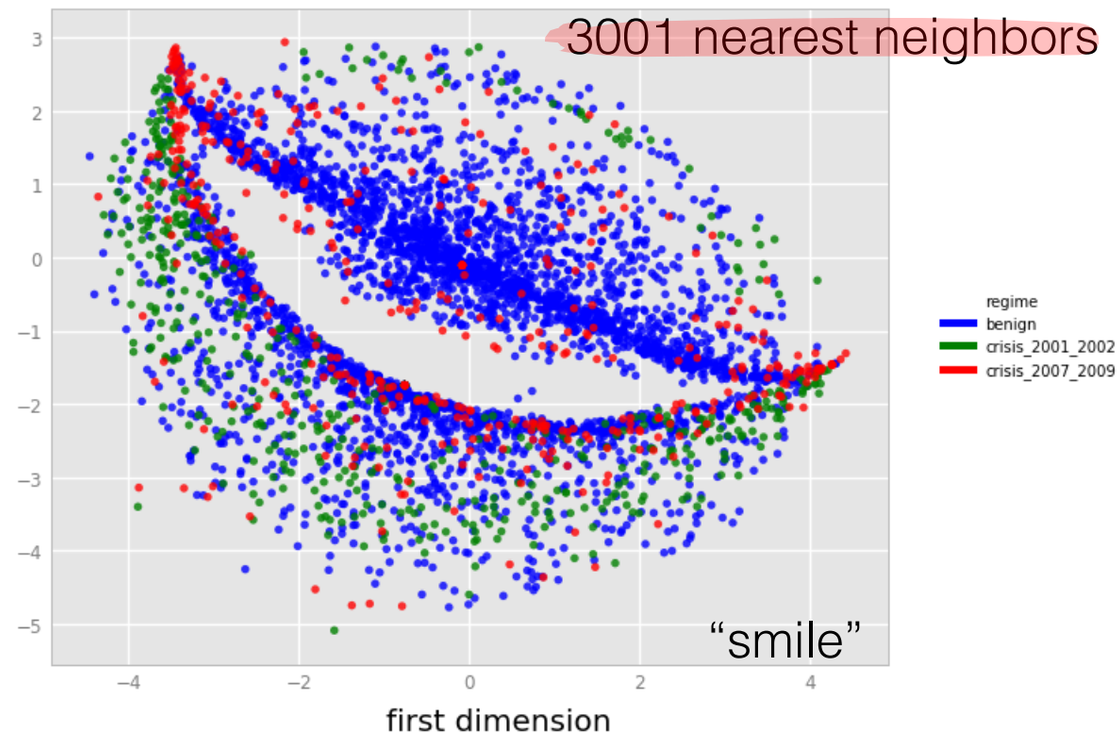
t-SNE for stock returns

Apply t-SNE to the DJI stock return data, for different values of perplexity









t-SNE algorithm: further notes

- Visualization results depend on perplexity. What perplexity would be optimal?

- In its basic form, t-SNE provides visualization of a given dataset but not the a dimensional reduction recipe/algorithm for a new, unseen data. New data points cannot be embedded into an existing low dimensional representation.

- t-SNE preserves nearest neighbors but not distance. Running distance or density based clustering algorithms on outputs of the t-SNE can be problematic!

- The basic algorithm is not well scalable with the number of data points (has complexity (scales as N^2). Additional tricks are required to apply it to large datasets, e.g. a tree-based algorithm of van der Maaten (2014) has $O(N \log N)$ complexity

Guided Tour of Machine Learning in Finance

Week 3: Unsupervised Learning

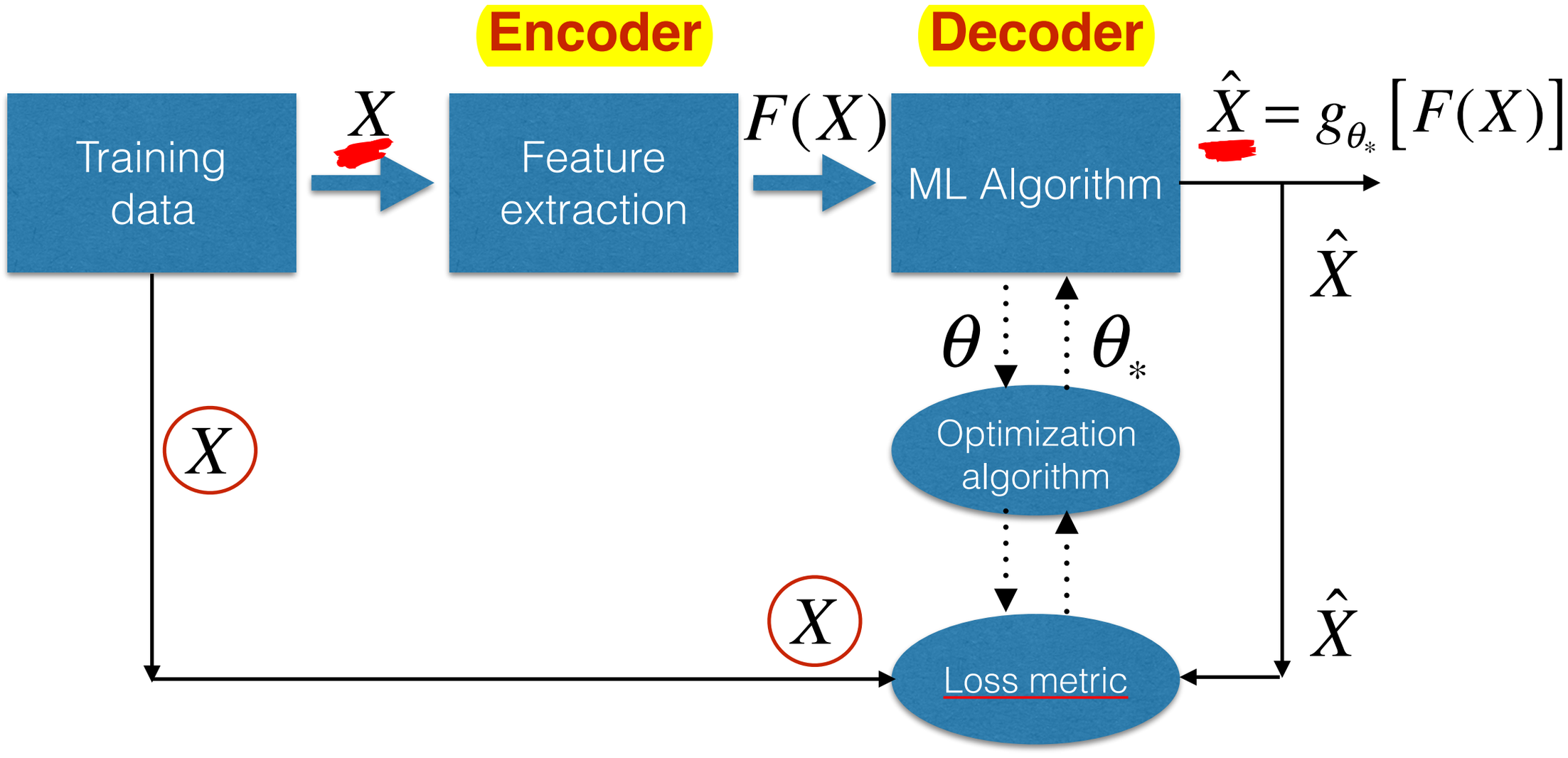
3-1-5-Autoencoder

Igor Halperin

NYU Tandon School of Engineering, 2017

Unsupervised learning diagram: Autoencoder

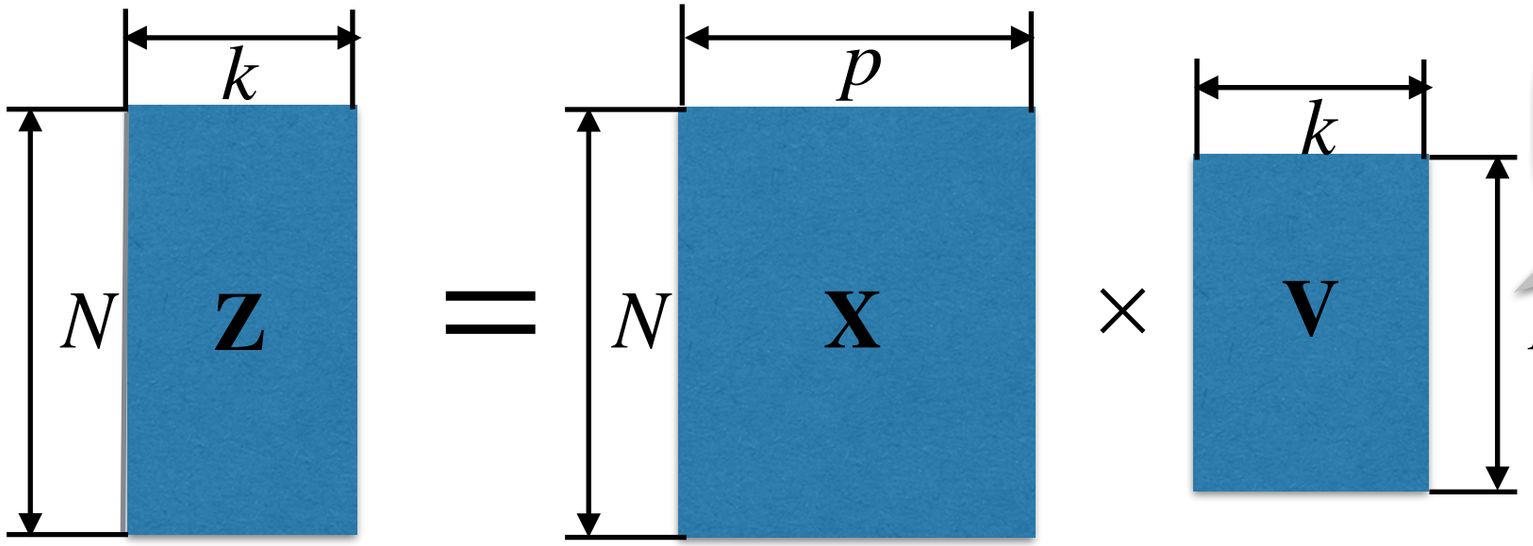
Autoencoder: replace Y with X!



PCA as a dimension reduction method

We had a linear transform (linear encoder) of the data $\mathbf{Z} = \mathbf{XV}$ parametrized by a $p \times k$ orthogonal matrix $\mathbf{V}$ with $\mathbf{VV}^T = 1$

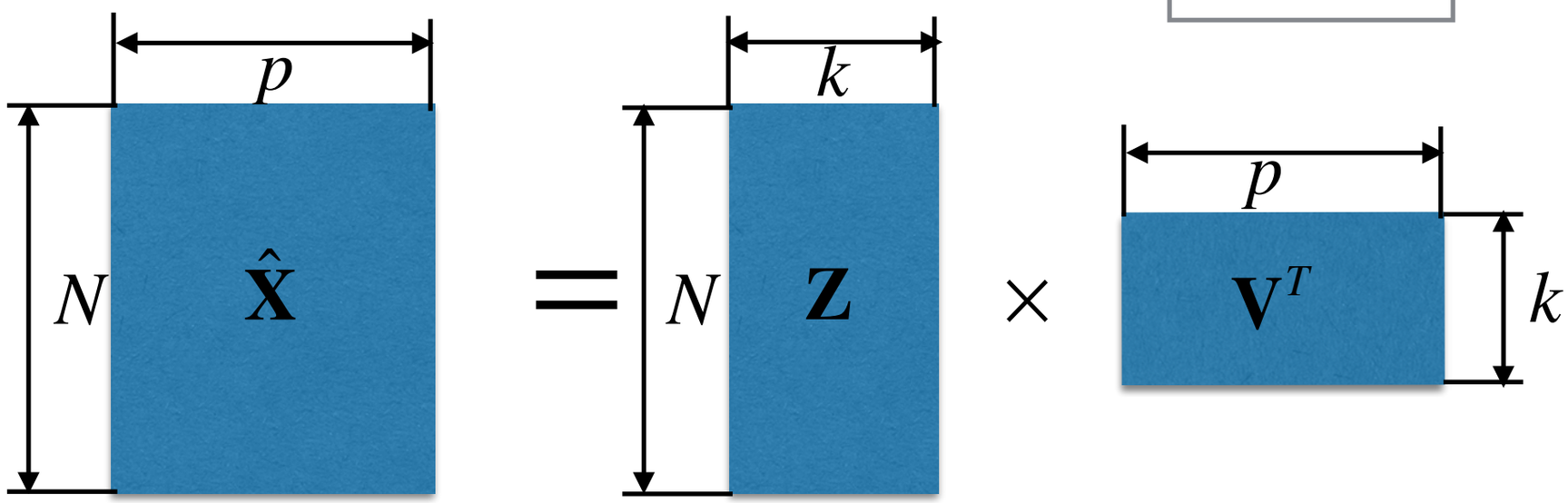
Special Linear Autoencoder



If $k = p, \mathbf{V} = \mathbf{U}$ this is just a rotation of the eigenvalue decomposition

A **decoded signal** (inverse transform) is obtained as $\hat{\mathbf{X}} = \mathbf{ZV}^T$

i.e training includes min of some loss



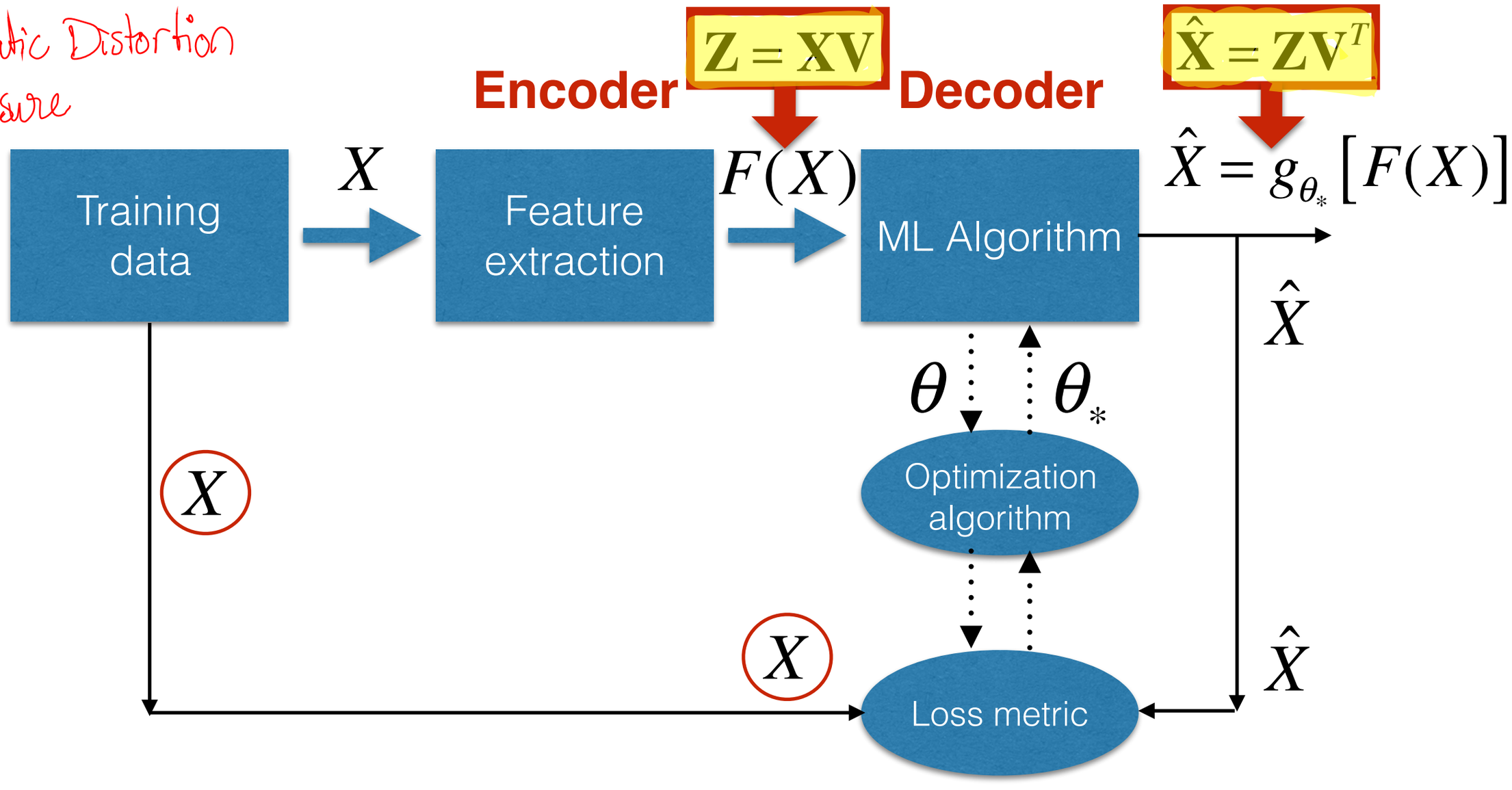
2nd interpretation → reconstruction error minimization

min the MSE error bet X & reconstructed $\hat{X}$

↓

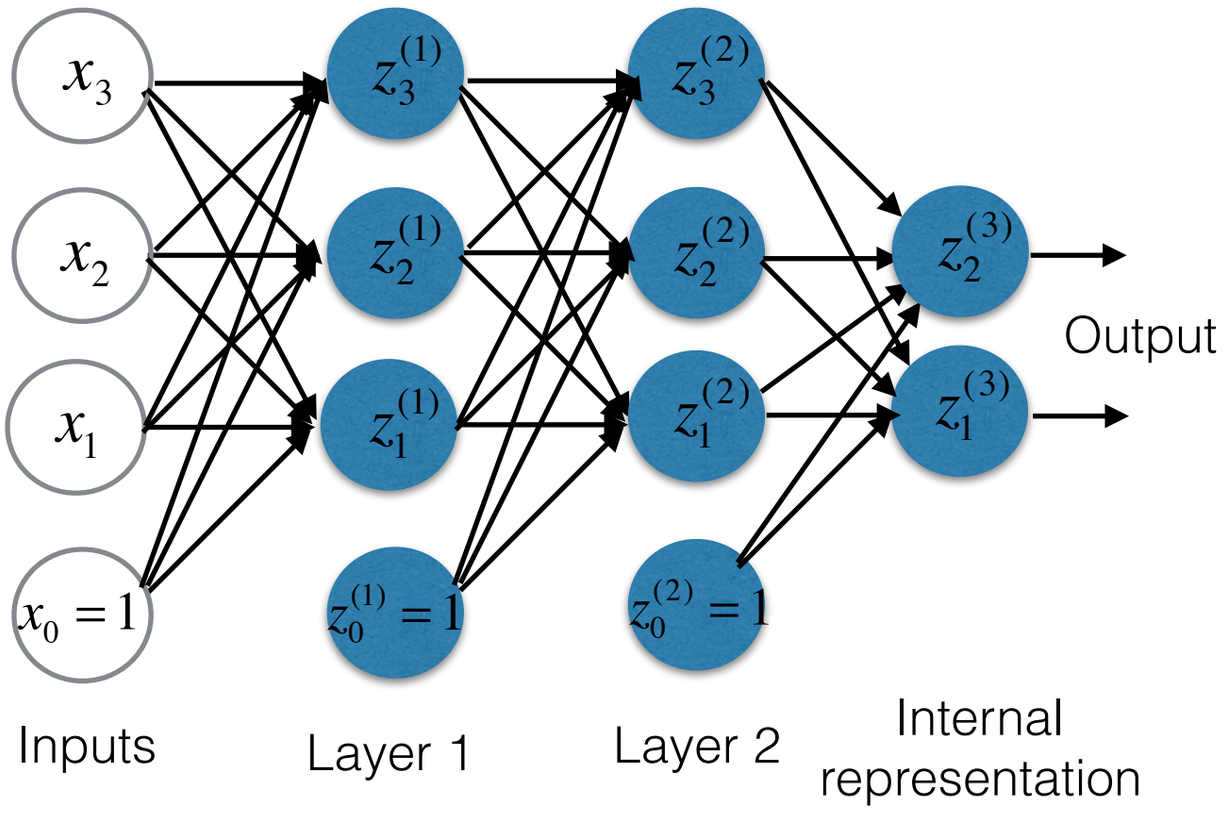
Quadratic Distortion Measure

Autoencoder



Neural Encoder

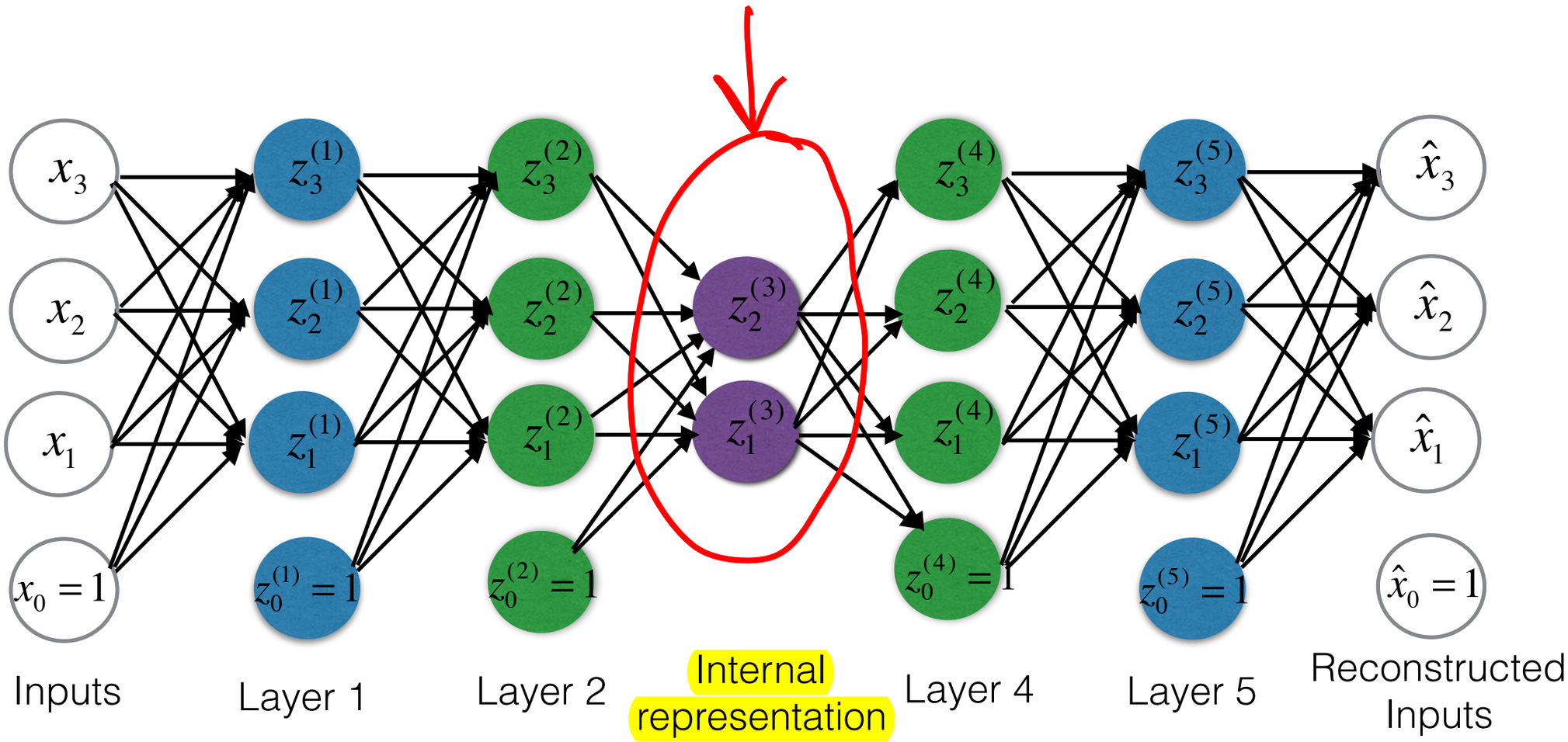
- **A Neural Encoder** is composed of layers of artificial neurons (perceptrons), with a final layer providing a low-dimensional representation of data



- For linear activation functions in all layers, this is equivalent to a single linear layer, which is equivalent to the Linear Encoder of the PCA
- If we use non-linear activations, then the depth of the network matters. For complex data, many layers might be needed to build a good internal representation.

Neural Encoder-Decoder

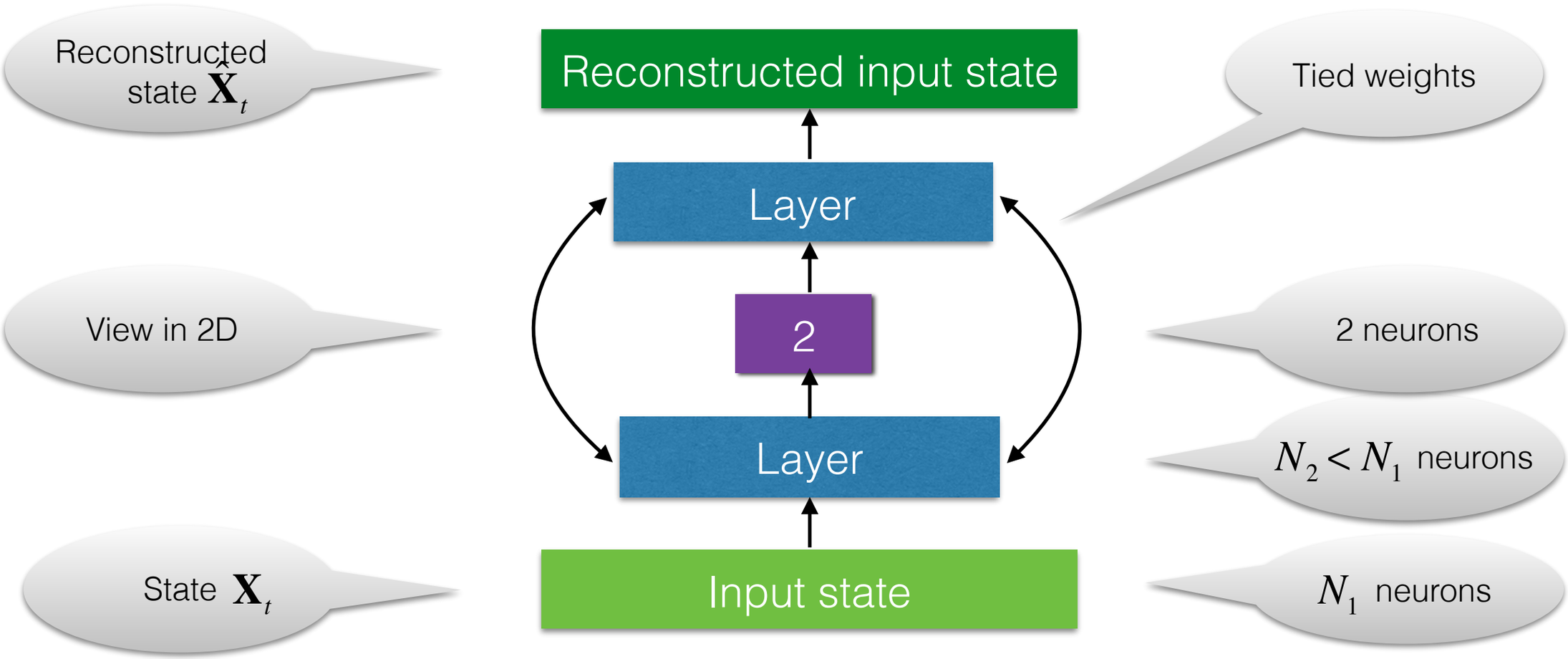
- **A Neural Encoder/Decoder pair implements an Autoencoder:**



- If the inner-most layer has only two neurons, they activation can be plotted for visualization of data with the autoencoder.
- A Stacked (multi-level) autoencoder may have layers with gradually decreasing number of neurons per layer
- Tied weights in the Encoder and Decoder are often used in practice

Visualizing data with autoencoder

Autoencoder provides a way to visualize a high-dimensional input vector by a non-linear projection onto a 2D space of neuron activations for the inner-most layer with 2 neurons:



Autoencoder is trained by minimizing the reconstruction error between input state $\mathbf{X}_t$ and reconstructed state $\mathbf{X}_t$

Control question

Select all correct answers

1. The PCA implements a Linear Encoder that converts correlated inputs $\mathbf{X}$ into uncorrelated features $\mathbf{Z}$ by a linear transform $\mathbf{Z} = \mathbf{XV}$
2. If the orthogonal matrix $\mathbf{V}$ has dimension $p \times p$, i.e. it keeps all eigenvectors of correlation matrix of $\mathbf{X}$, then $\mathbf{Z}$ preserves the total variation of $\mathbf{X}$.
3. When the data is noisy or non-stationary, $\mathbf{Z}$ can have a higher variance than $\mathbf{X}$.
4. The PCA is a probabilistic method that enables simulating from data.

Correct answer: 1, 2