Guided Tour of Machine Learning in Finance

Week 3: Unsupervised Learning

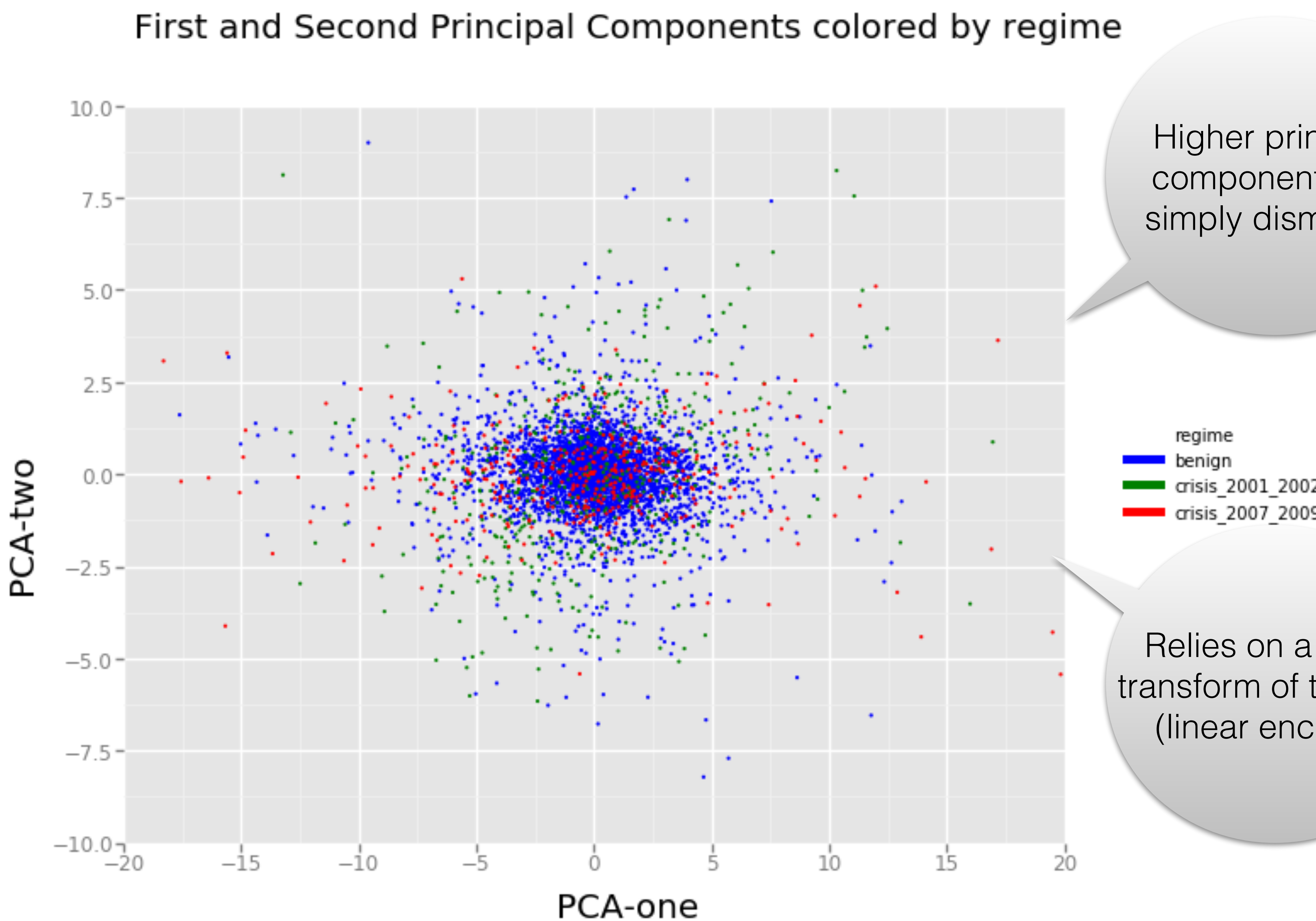
Dimension reduction and data visualization with t-SNE

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?

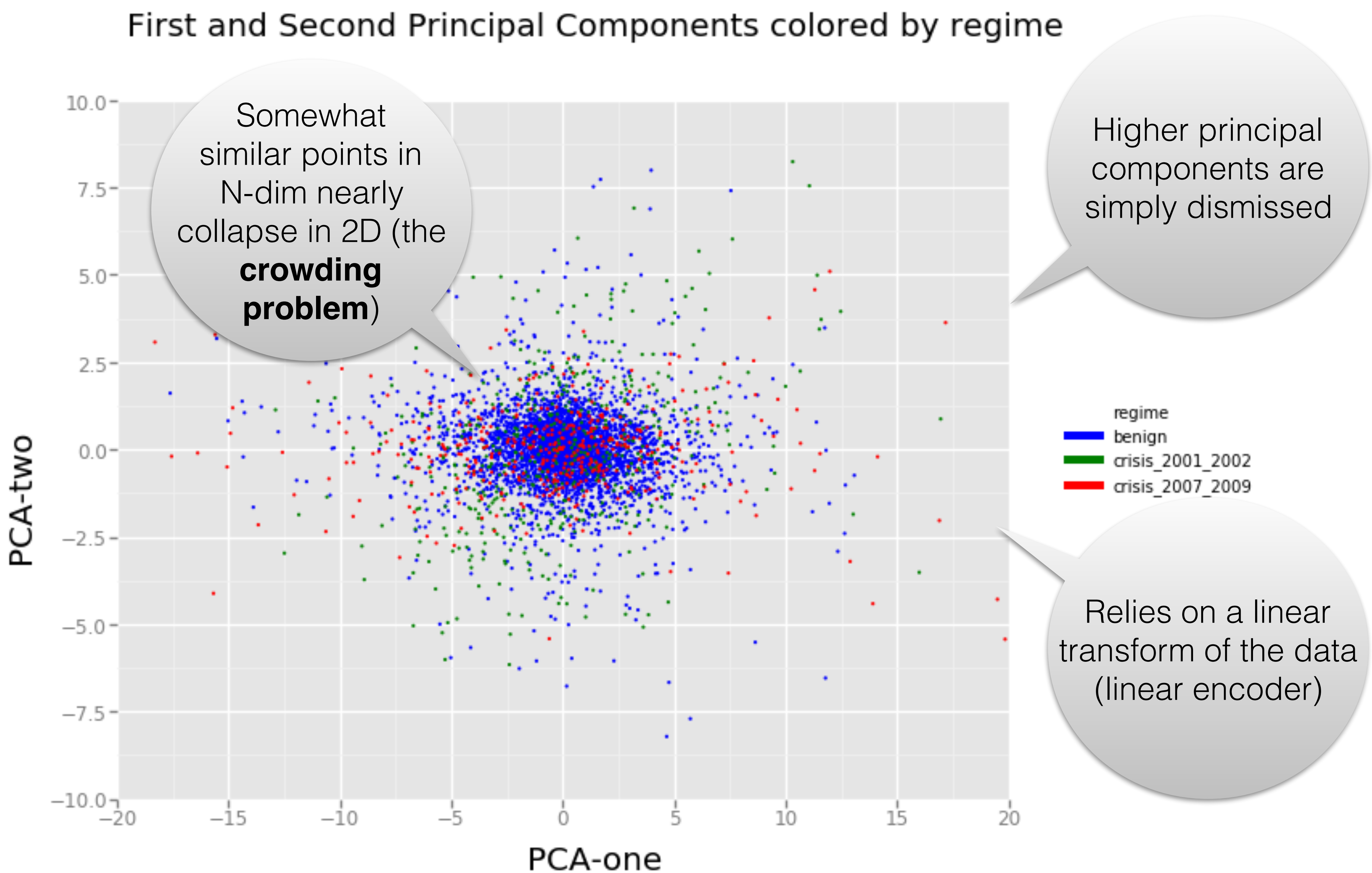


Higher principal components are simply dismissed

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

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:

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

5. Optimize C with respect to the choice of points $\left\{\mathbf{y}_i\right\}$ 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.

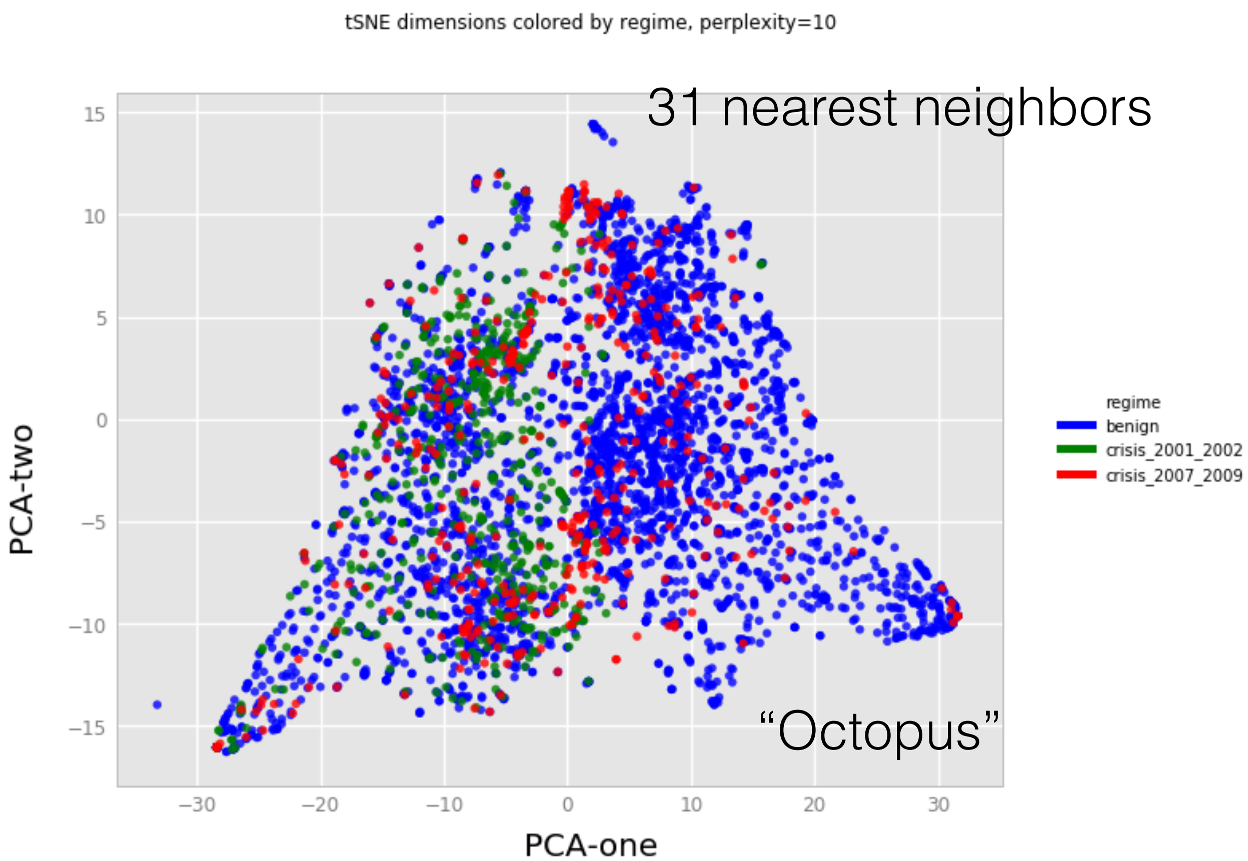
Algorithm 1: Simple version of t-Distributed Stochastic Neighbor Embedding.

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

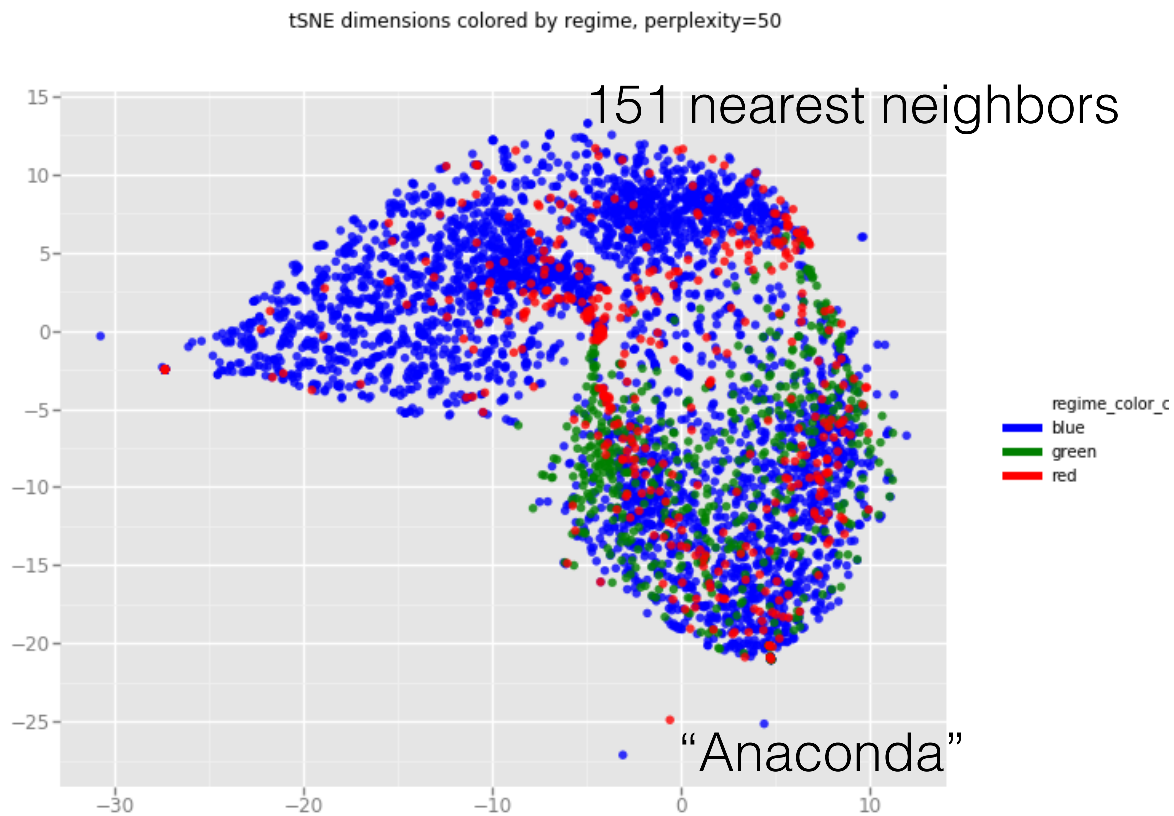


31 nearest neighbors

“Octopus”

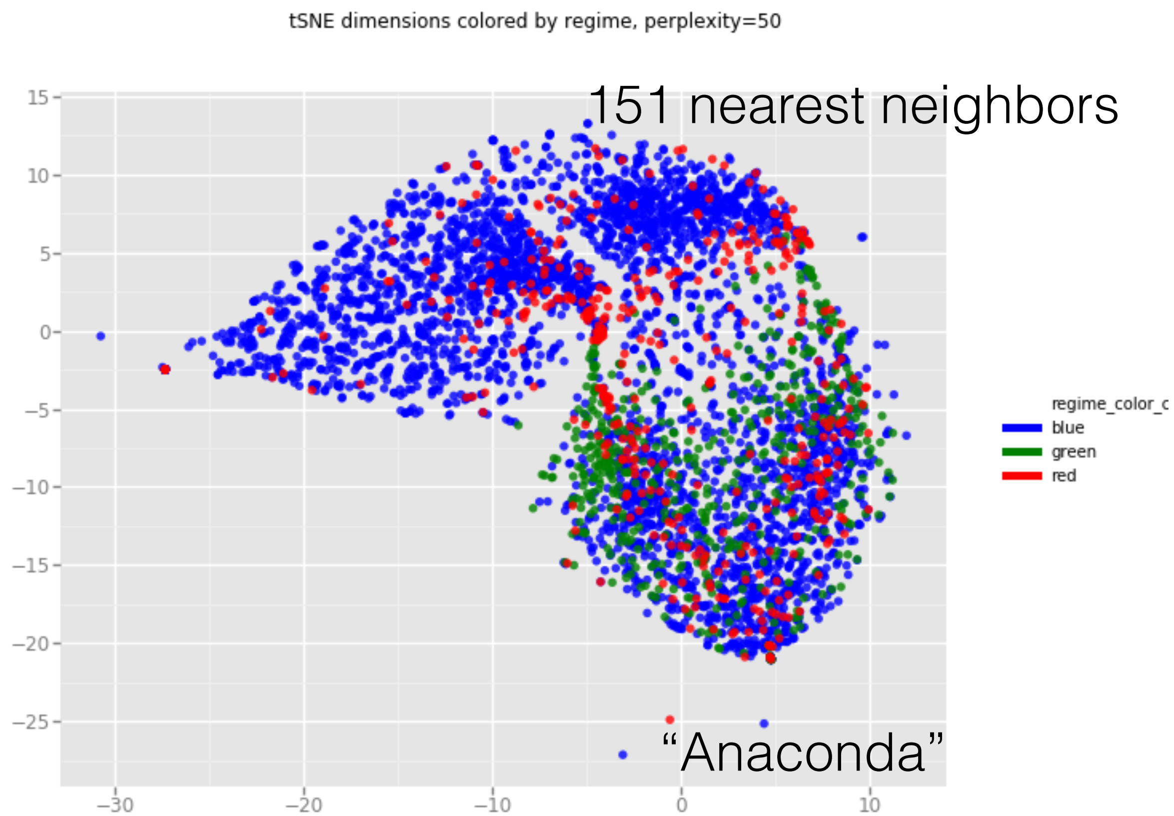
t-SNE for stock returns

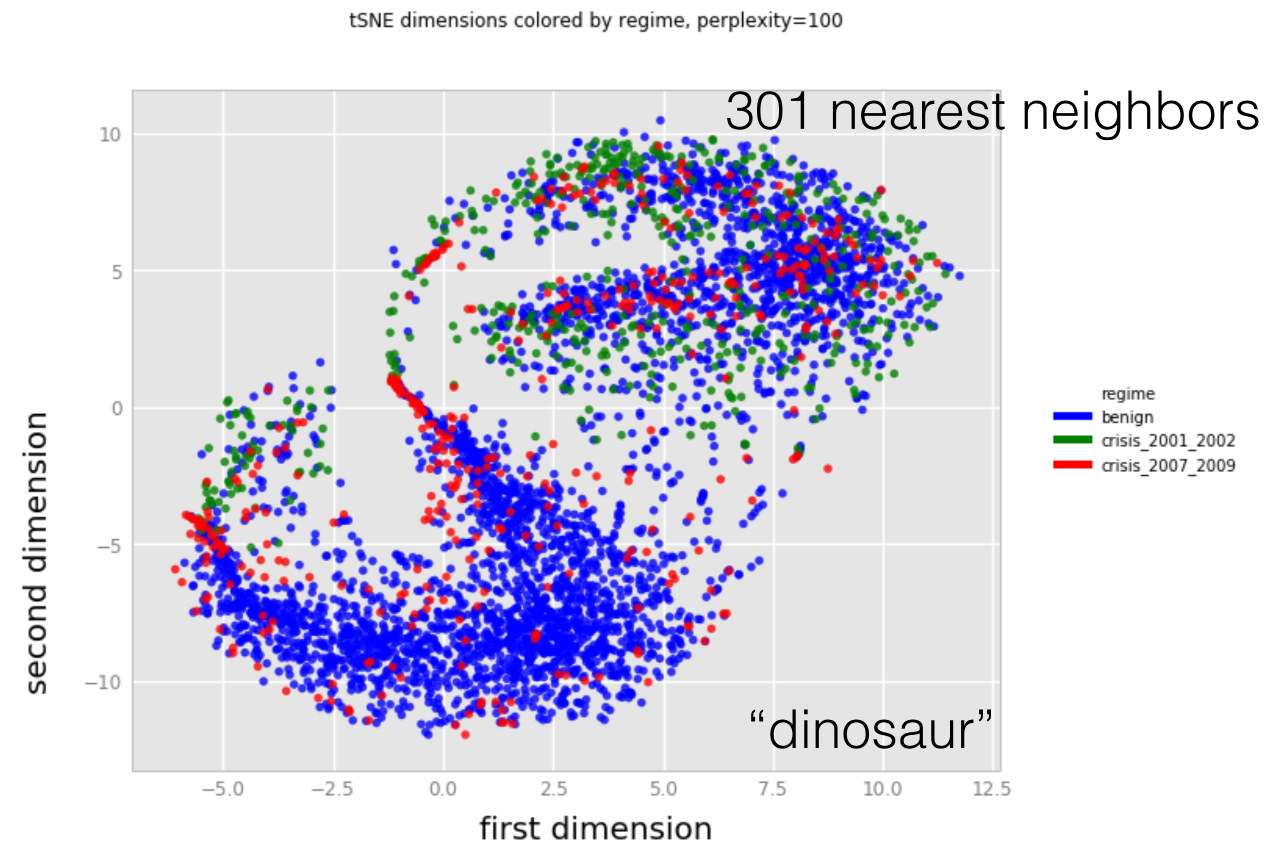
Apply t-SNE to the DJI stock return data, for different values of perplexity



t-SNE for stock returns

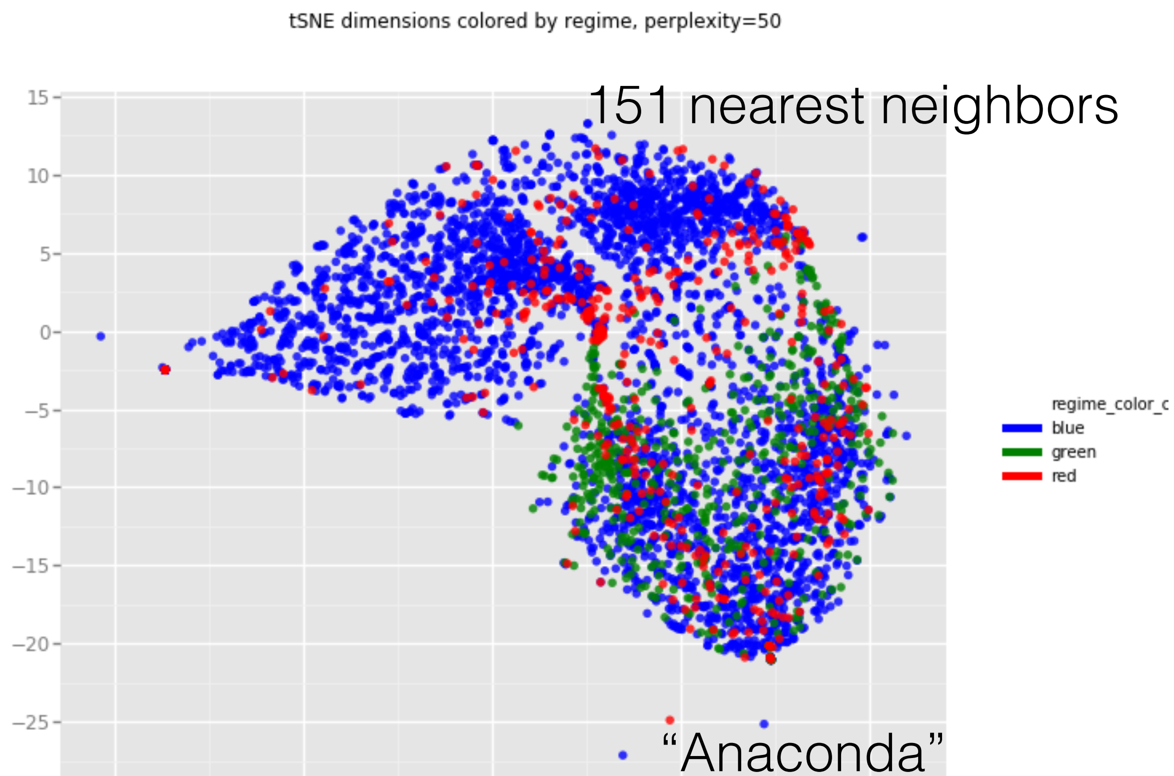
Apply t-SNE to the DJI stock return data, for different values of perplexity

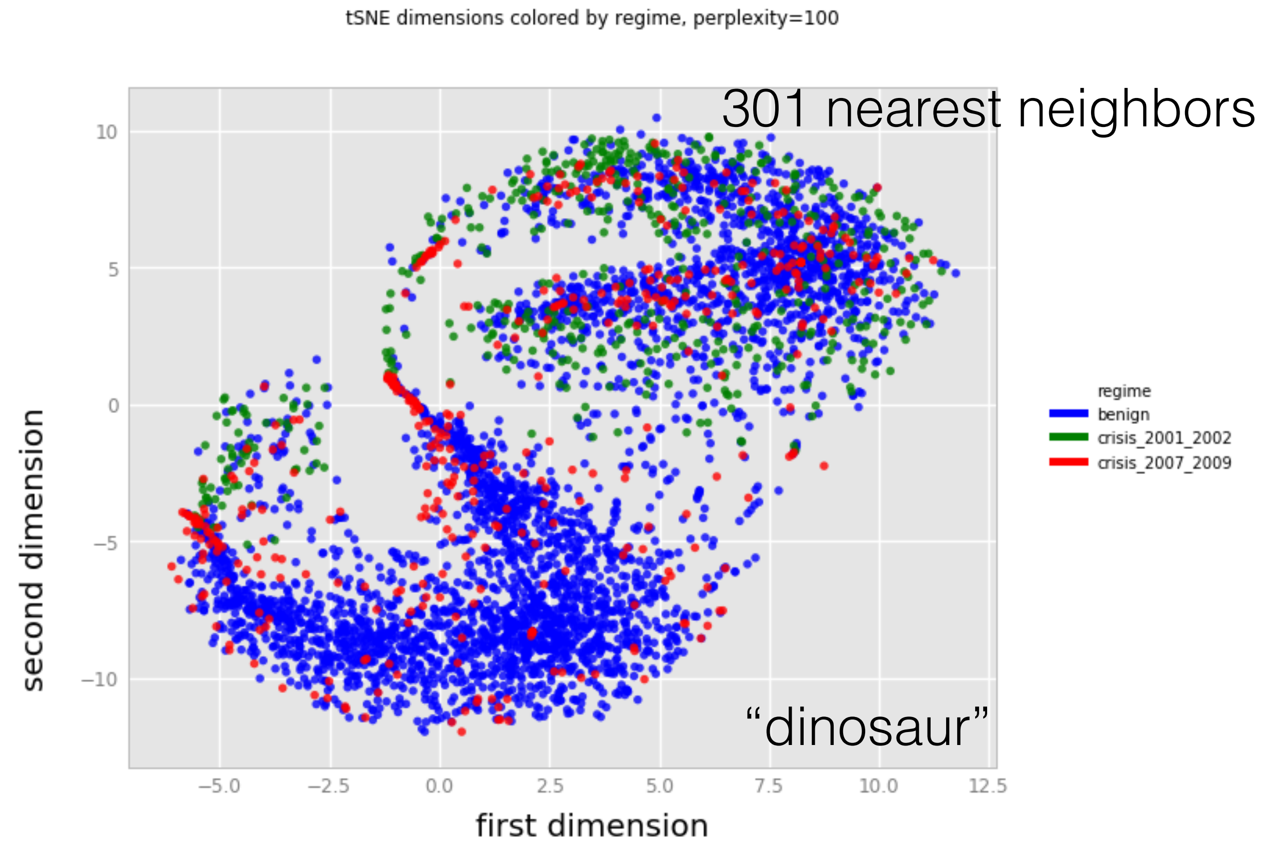


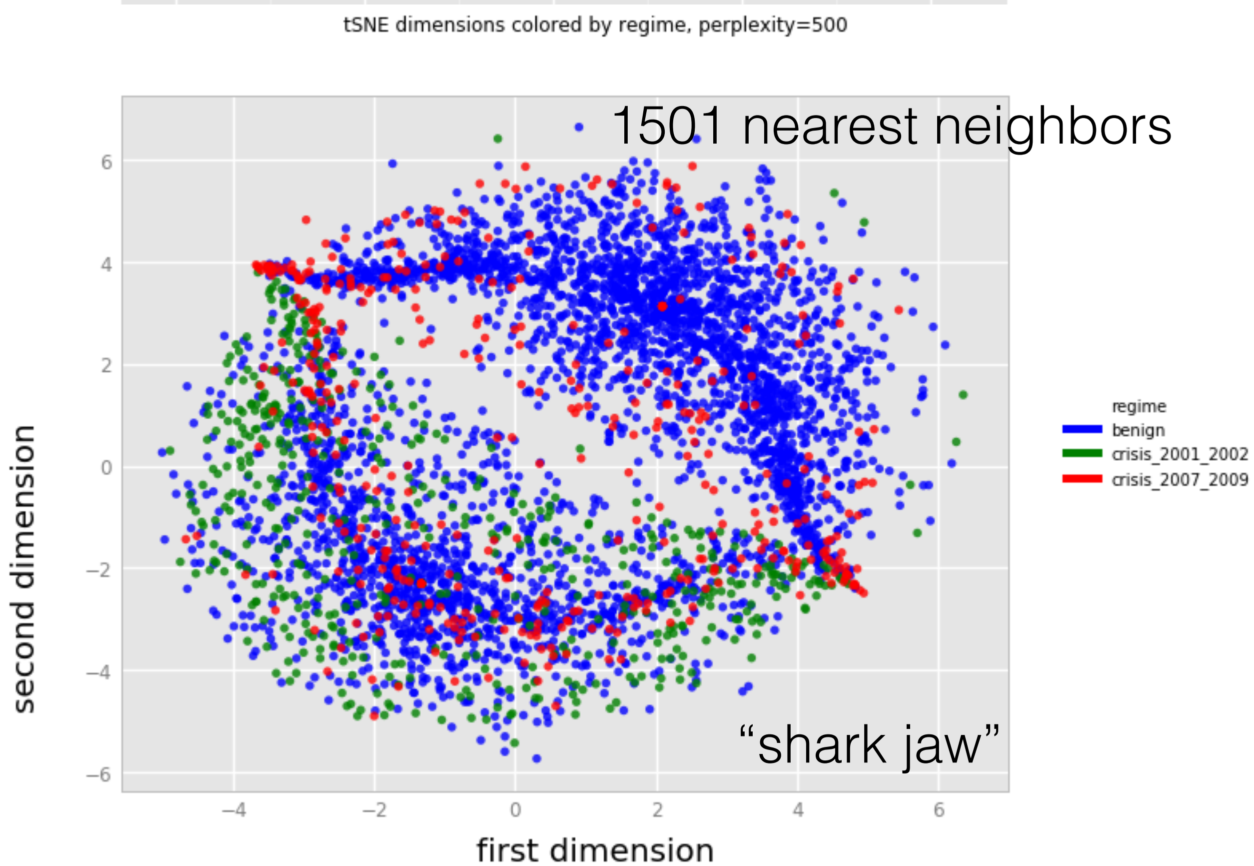


t-SNE for stock returns

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

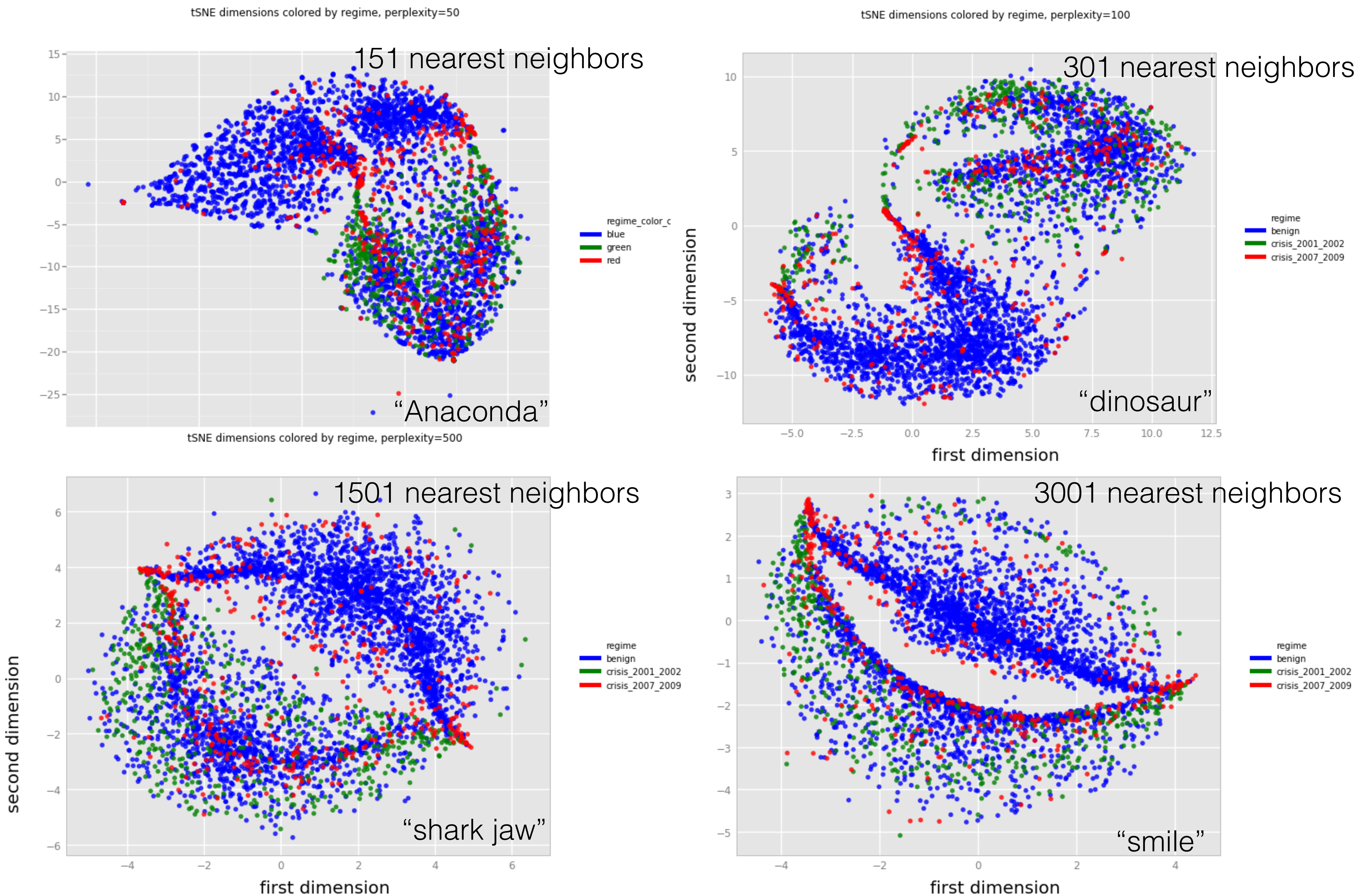






t-SNE for stock returns

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 $O\left(N\log N\right)$ complexity