The effect of anisotropic node splits on Wavelet decomposition of RF

Asaf Abas Uria Mor

02/09/2018

Single Decision Tree

Settings:
Given a sample set of a real (or vector) valued function f on some convex domain $\Omega_0 \in \mathbb{R}^n$: $\{x_i \in \Omega_0, f(x_i)\}_{i \in I}$.
Our goal is to find efficient representation of the underlying function f, overcoming the complexity, geometry and (usually) non-smoothness nature of the function in the sampled points.

Single Decision Tree

We divide Ω_0 into two subdomains, by intersecting it with a hyperplane. The subdivision is performed to minimize a given cost function.
We recursively repeat the subdivision process on each of the subdomains until some stopping criteria is met and which defines the leafs of the tree.

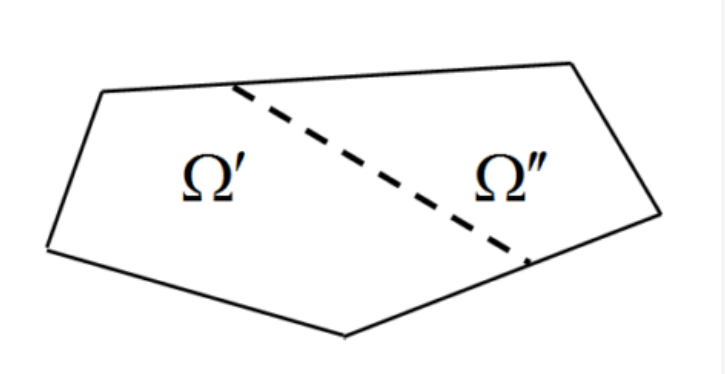

Figure: Subdivision illustration

One possible choice of cost function is one such minimizes the variance in both subdomains, i.e, a partition of Ω by a hyper-plane into two convex subdomains Ω', Ω'', minimizing the cost function

$$\sum_{x_i \in \Omega'} \left| f(x_i) - \overline{f(\Omega')} \right|^2 + \sum_{x_i \in \Omega''} \left| f(x_i) - \overline{f(\Omega'')} \right|^2$$

where $\overline{f(\Omega)}$ denotes the mean outcome of samples in Ω.

Minimizing the cost function

The problem can be formulated as an optimization problem:
find $\theta_0, ..., \theta_d$ which minimize

$$\sum_{x_i \in \Omega'} \left| f(x_i) - \overline{f(\Omega')} \right|^2 + \sum_{x_i \in \Omega''} \left| f(x_i) - \overline{f(\Omega'')} \right|^2$$

where $\Omega' = \left\{ x = (x_1, ..., x_d) \in \Omega; \sum_{i=1}^{d} \theta_i x_i < \theta_0 \right\}$ and $\Omega'' = \Omega \setminus \Omega'$
Note that the search space of this problem is far from ideal - slightly changing one of the θs can (and will) result a massive jump in the cost function values. Moreover, since the cost function is not continuous, we can't use standard optimization algorithms (CG, Newton, GD etc...)

The division challenge

The number of possible hyperplanes which will divide a region containing n data samples over d features is $\binom{n}{d} \geq \frac{n^d}{d^d}$ at worst (for 100 samples over 6 features this is 21433470...).

In many applications of decision trees, searching through all possible subdivisions is impractical. Thus, in practice - most prominent implementations of DT (scikit-learn&R included) restrict the search to subdivisions by an isotropic hyperplane, that is, splits parallel to one of the features.

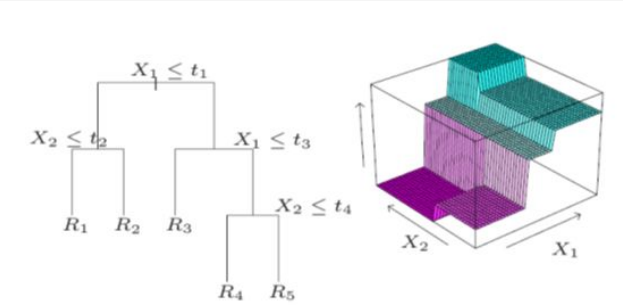

Figure: Isotropic split

Our work here aims to find a near optimal anisotropic split yielding better progress (and thus better approximates the data) compared to isotropic splits

Our approach

Since it is not feasible to go through all possible subdivisions, we turn to heuristics that will (hopefully) find a near optimal hyper-plane which (fingers crossed) outperforms the isotropic split. We propose three approaches built one on top of the other:

1. **Capture the geometry** as if there is **no problem**
2. **Solve the problem** as if there is **no geometry**
3. **Capture the geometry of the problem**

Our approach

Since it is not feasible to go through all possible subdivisions, we turn to heuristics that will (hopefully) find a near optimal hyper-plane which (fingers crossed) outperforms the isotropic split. We propose three approaches built one on top of the other:

1. **Capture the geometry** as if there is **no problem**
   1. Run KMeans ($k = 2$) on the outcomes - Y and label each sample with the predicted label
   2. Use SVM to best separate the labeled data in the features space
2. **Solve the problem** as if there is **no geometry**
3. **Capture the geometry of the problem**

Our approach

Since it is not feasible to go through all possible subdivisions, we turn to heuristics that will (hopefully) find a near optimal hyper-plane which (fingers crossed) outperforms the isotropic split. We propose three approaches built one on top of the other:

1. **Capture the geometry** as if there is **no problem**
2. **Solve the problem** as if there is **no geometry**
   1. Reduce the dimensionality of X space (PCA)
   2. Use first two PCs as starting points for standard gradient-free optimization (Nedler-Mead solver)
3. **Capture the geometry of the problem**

Our approach

Since it is not feasible to go through all possible subdivisions, we turn to heuristics that will (hopefully) find a near optimal hyper-plane which (fingers crossed) outperforms the isotropic split. We propose three approaches built one on top of the other:

1. **Capture the geometry** as if there is **no problem**
2. **Solve the problem** as if there is **no geometry**
3. **Capture the geometry of the problem**
   1. Try to find p - the multidimensional direction in the X space that explains the maximum multidimensional variance direction in the Y space (PLS)
   2. Find H - the hyperplane which p is it's normal
   3. Find best offset of H

Toy demo

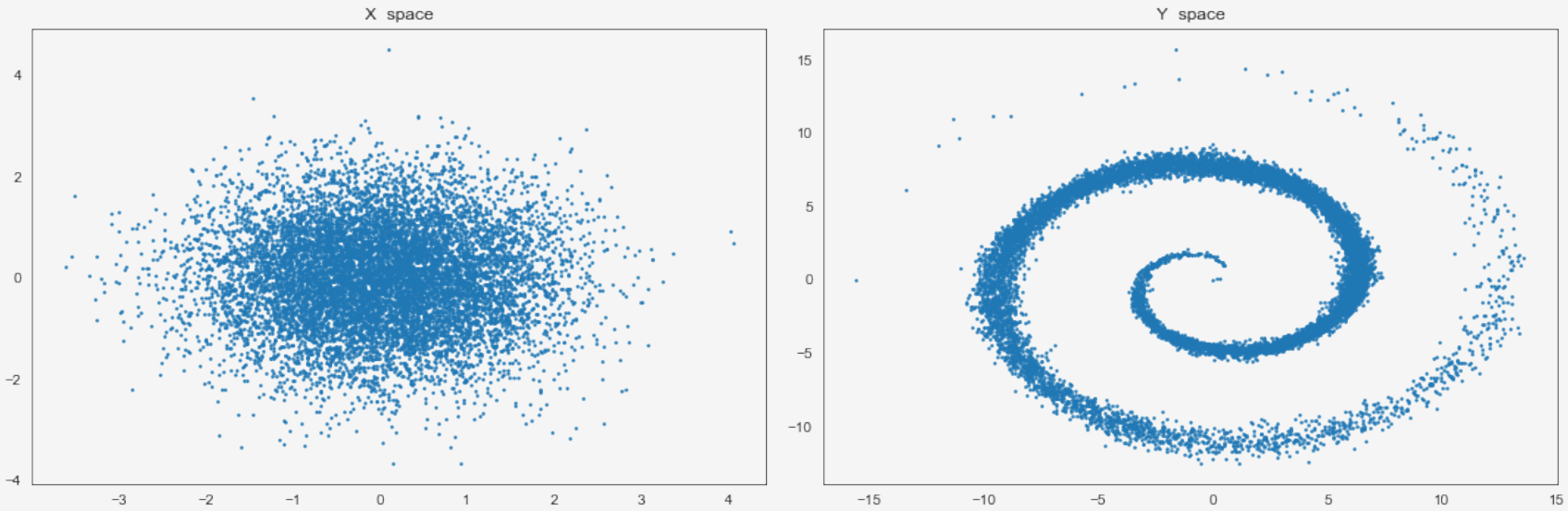

Figure: Iso splits

Toy demo

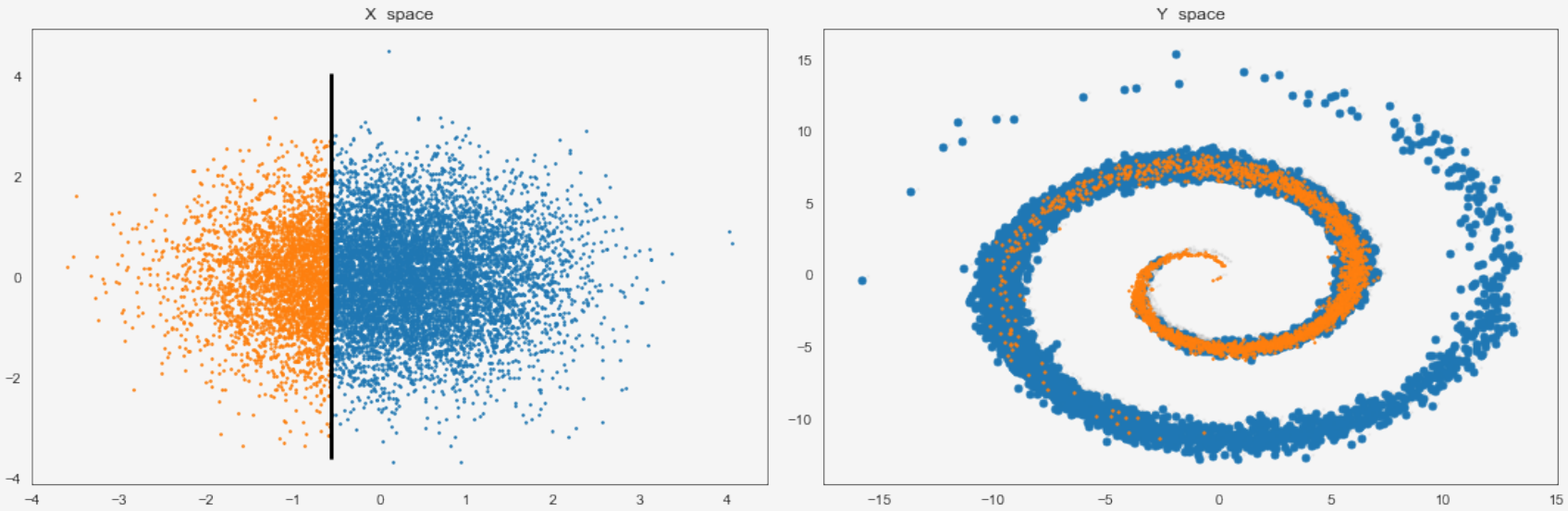

Figure: Iso splits

Toy demo

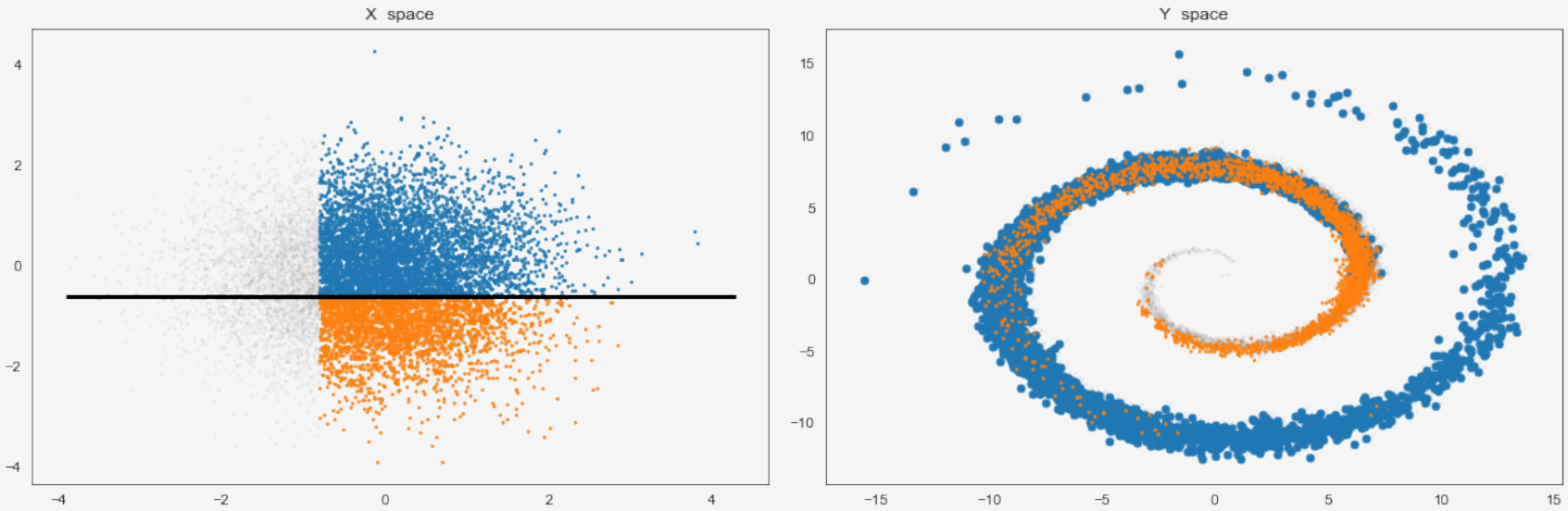

Figure: Iso splits

Toy demo

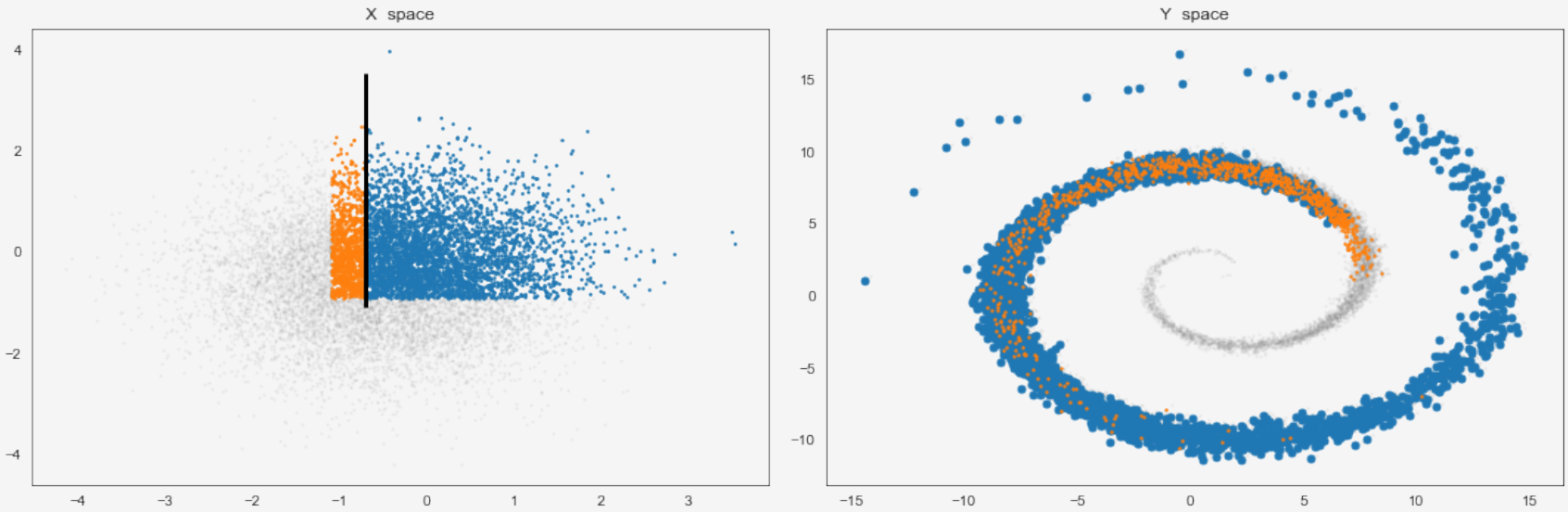

Figure: Iso splits

Toy demo

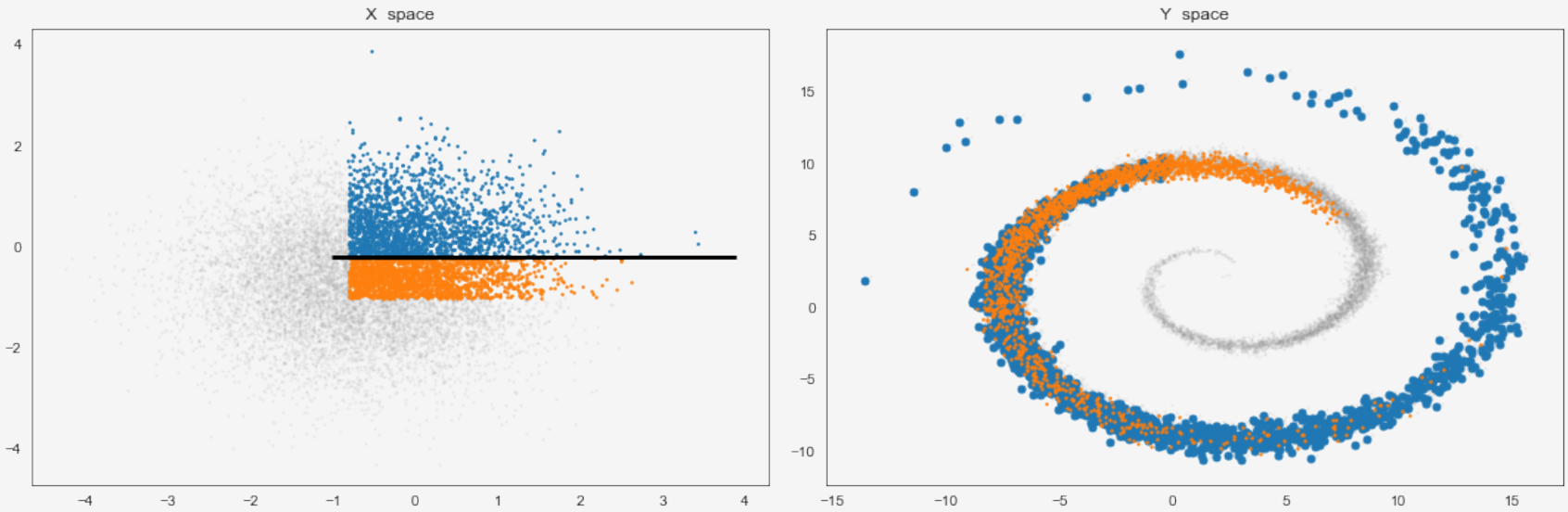

Figure: Iso splits

Toy demo

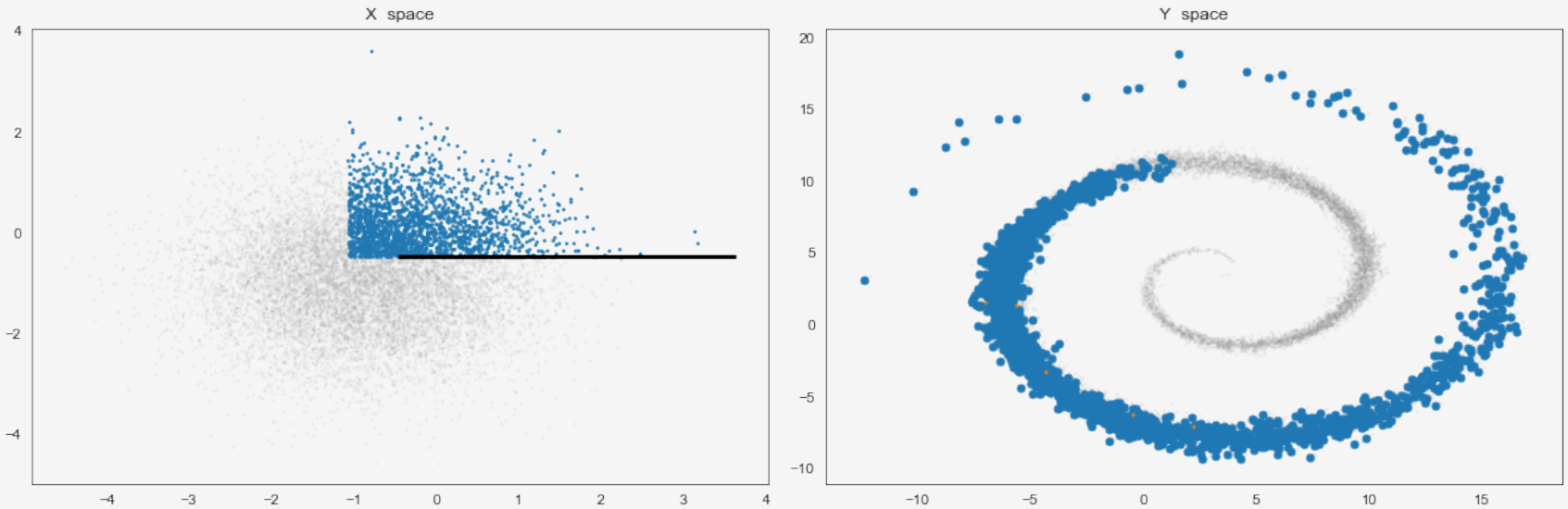

Figure: Iso splits

Toy demo

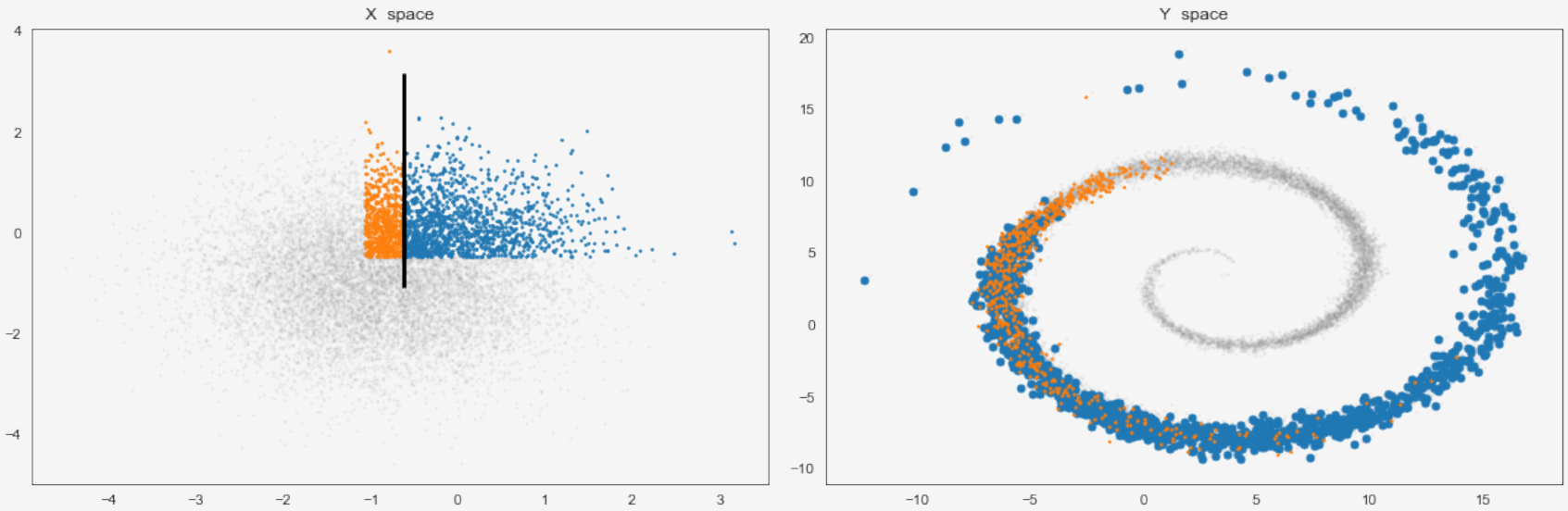

Figure: Iso splits

Toy demo

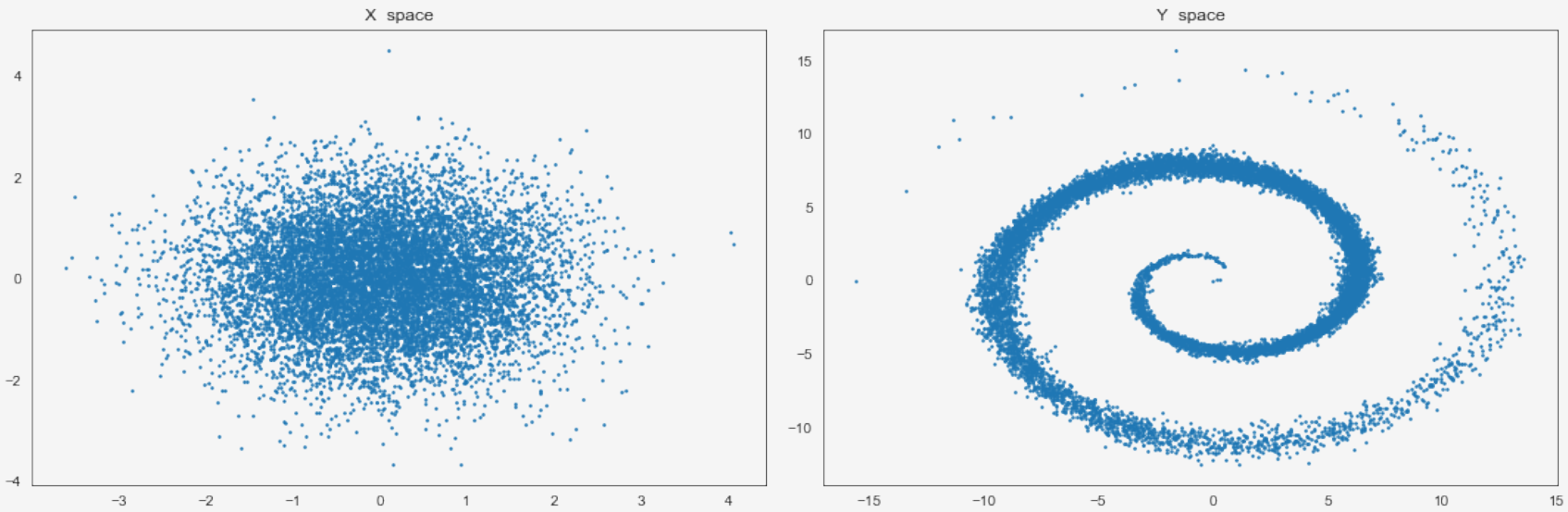

Figure: 2Means

Toy demo

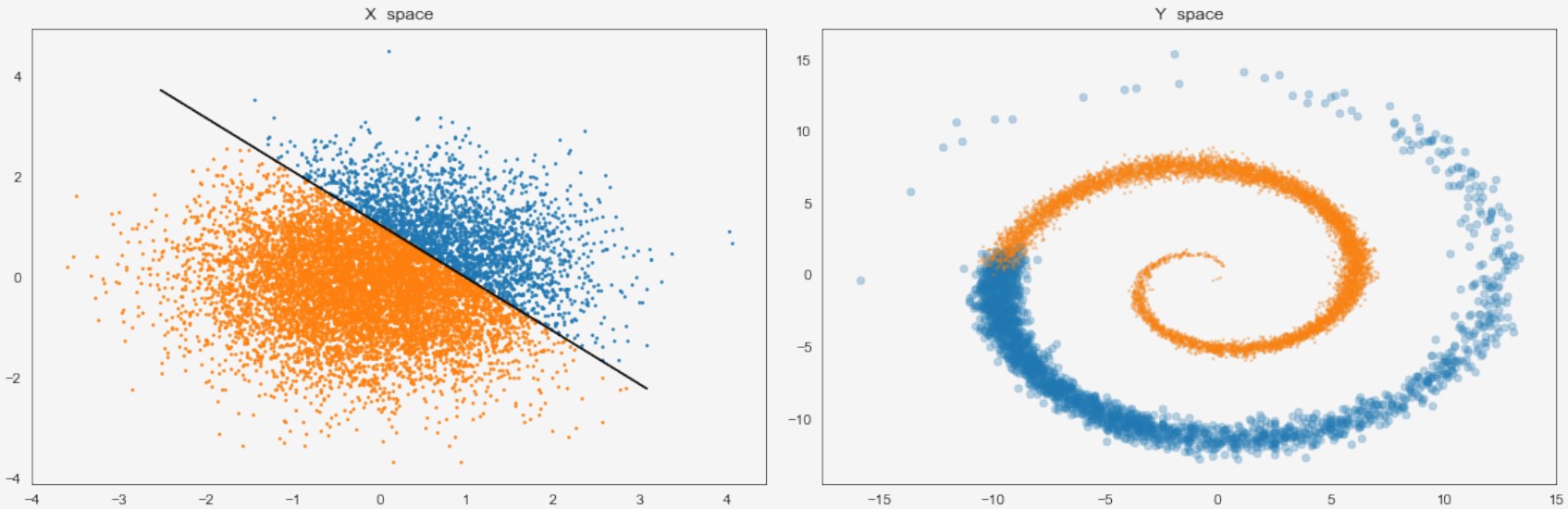

Figure: 2Means split

Toy demo

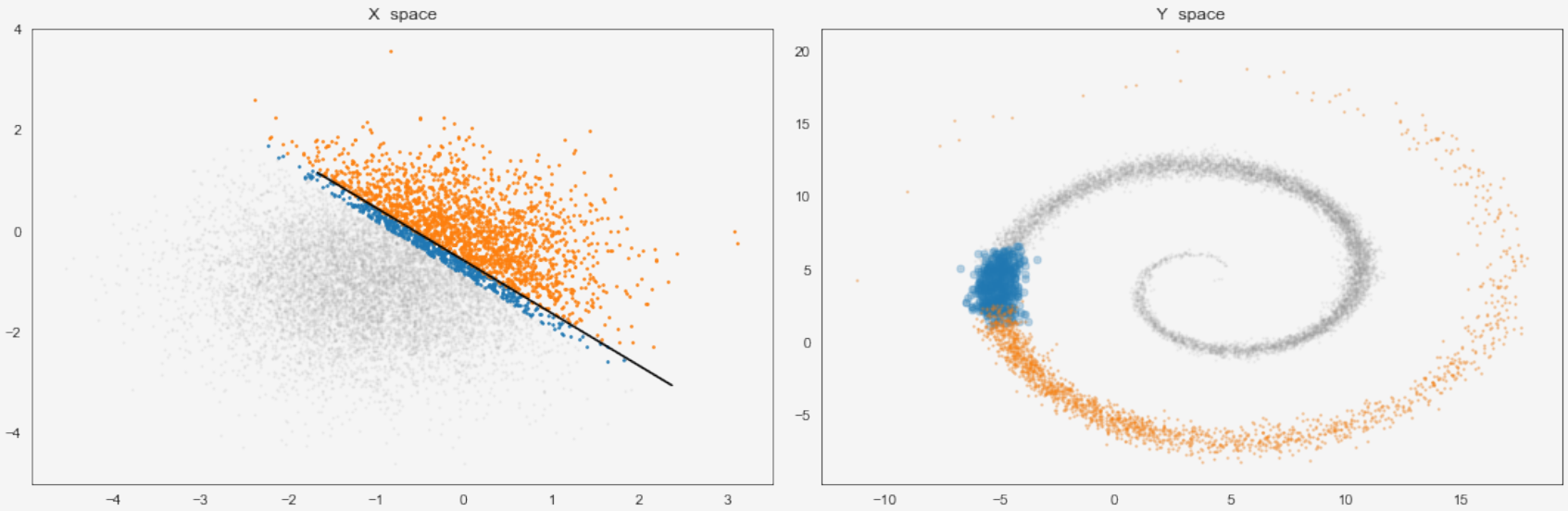

Figure: 2Means split

Toy demo

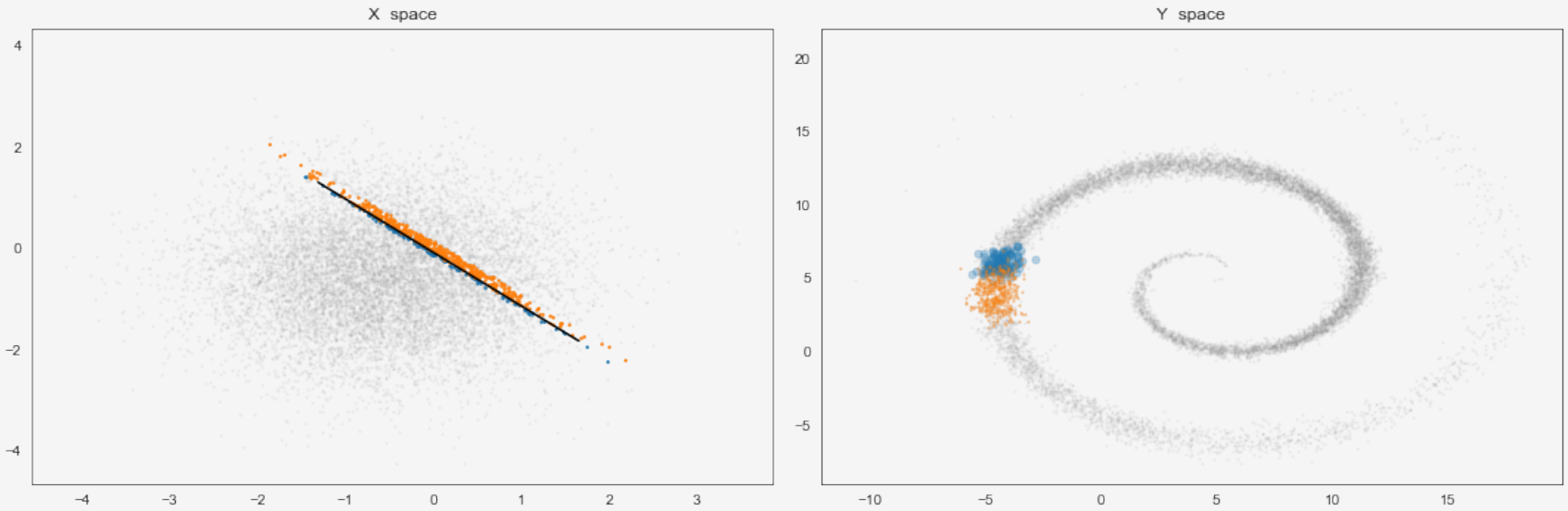

Figure: 2Means split

Toy demo

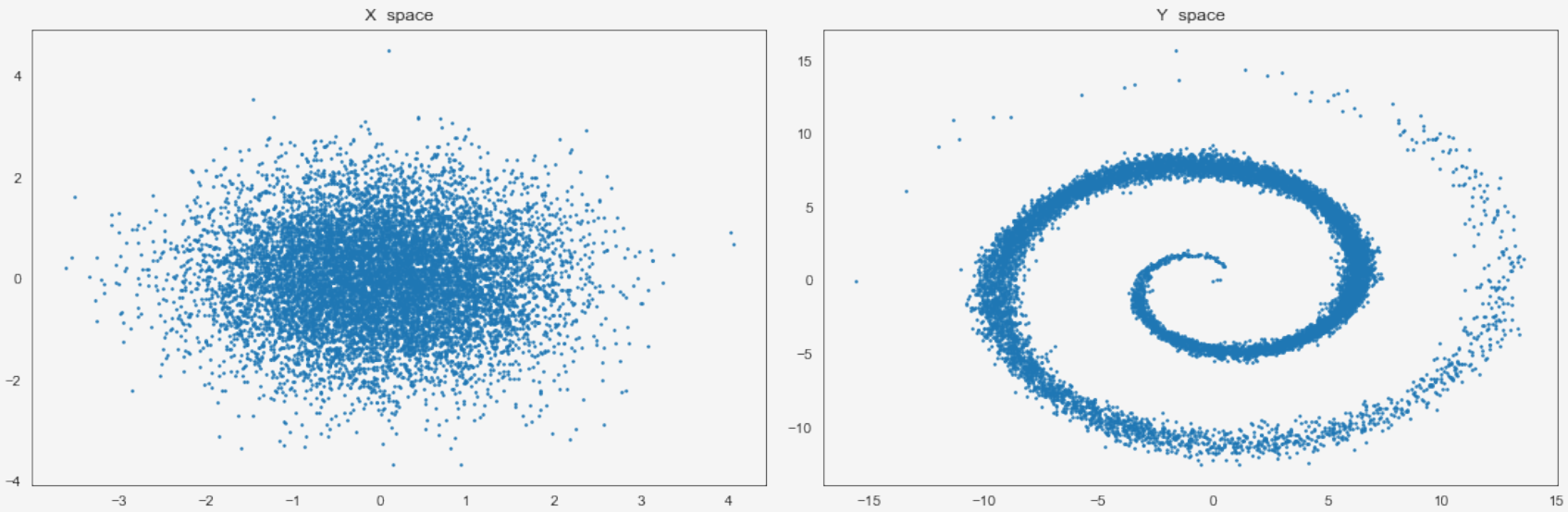

Figure: PCA

Toy demo

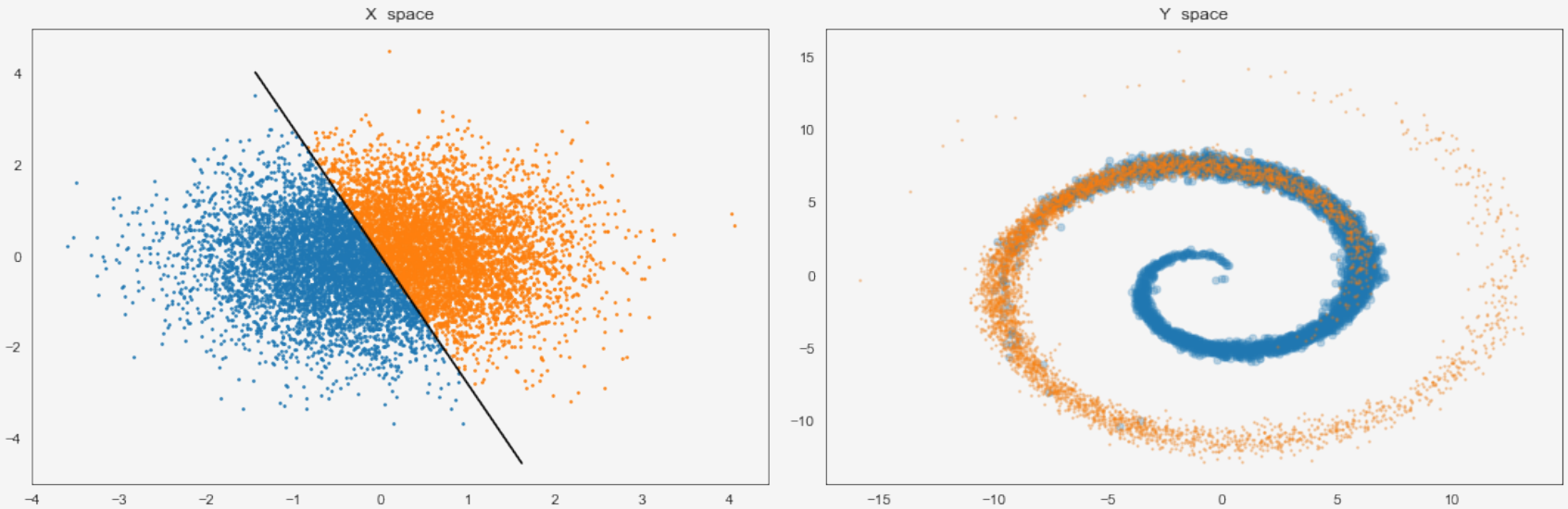

Figure: PCA

Toy demo

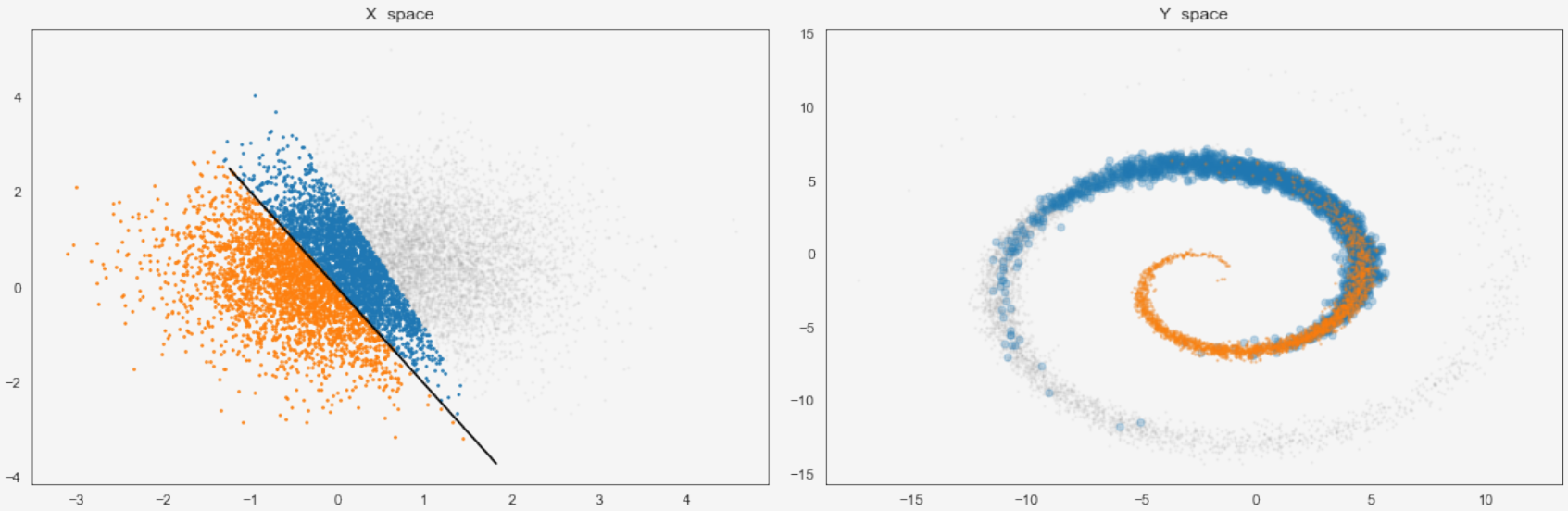

Figure: PCA

Toy demo

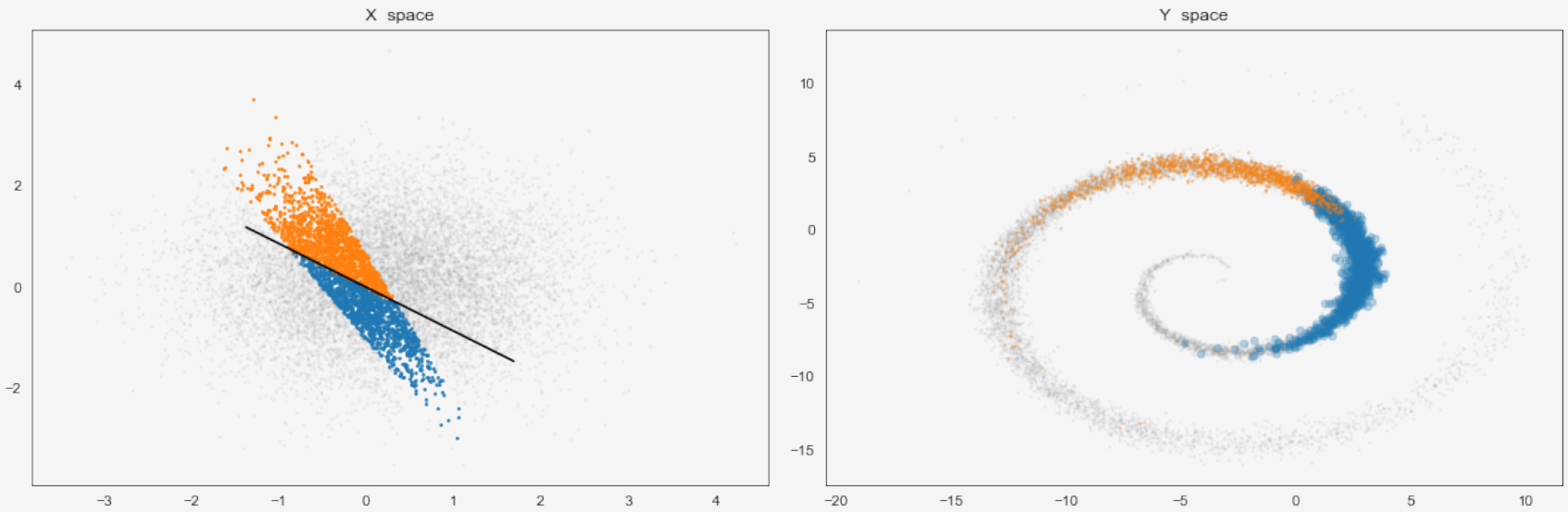

Figure: PCA

Toy demo

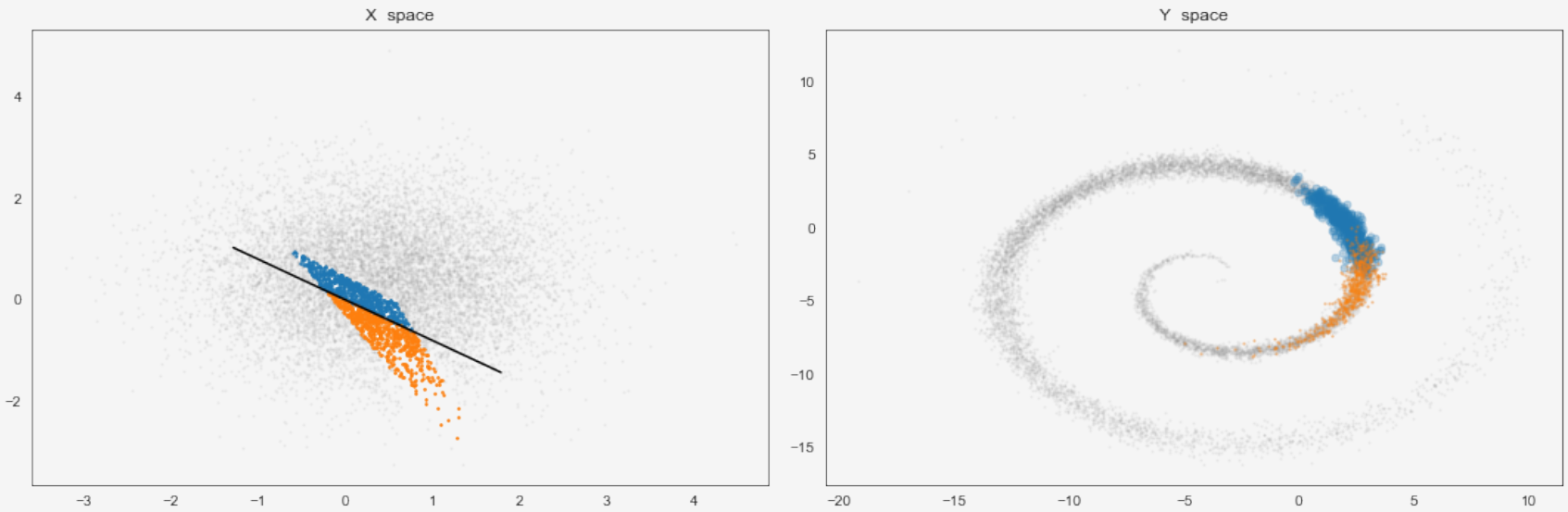

Figure: PCA

Toy demo

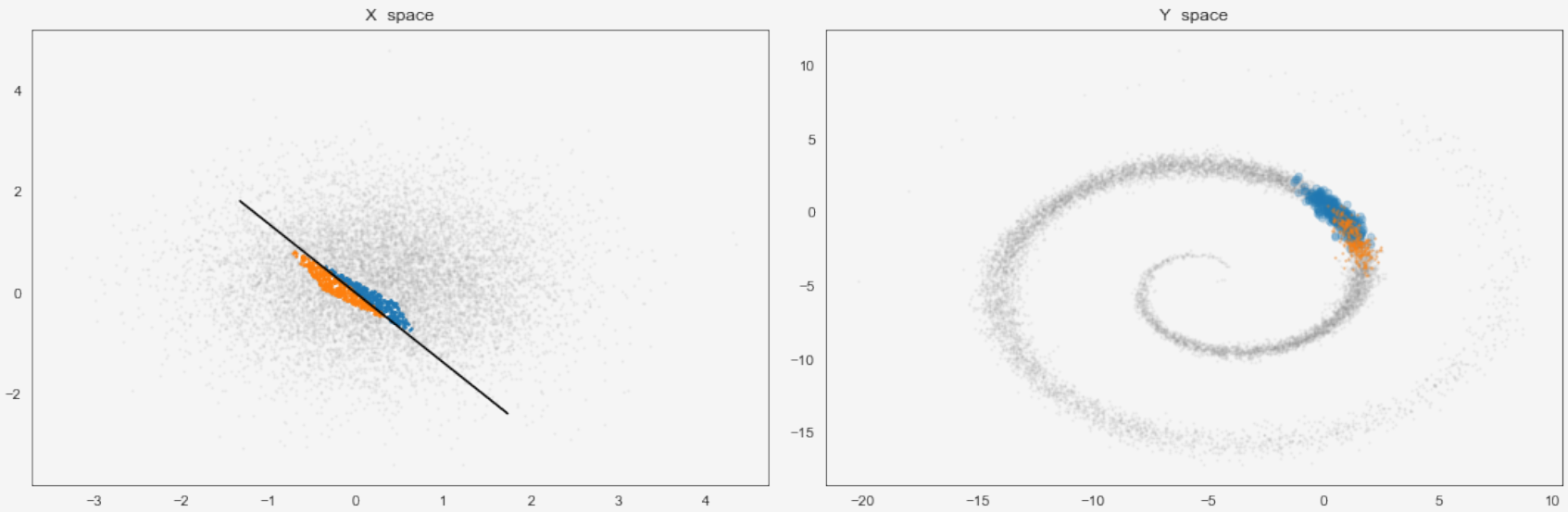

Figure: PCA

Toy demo

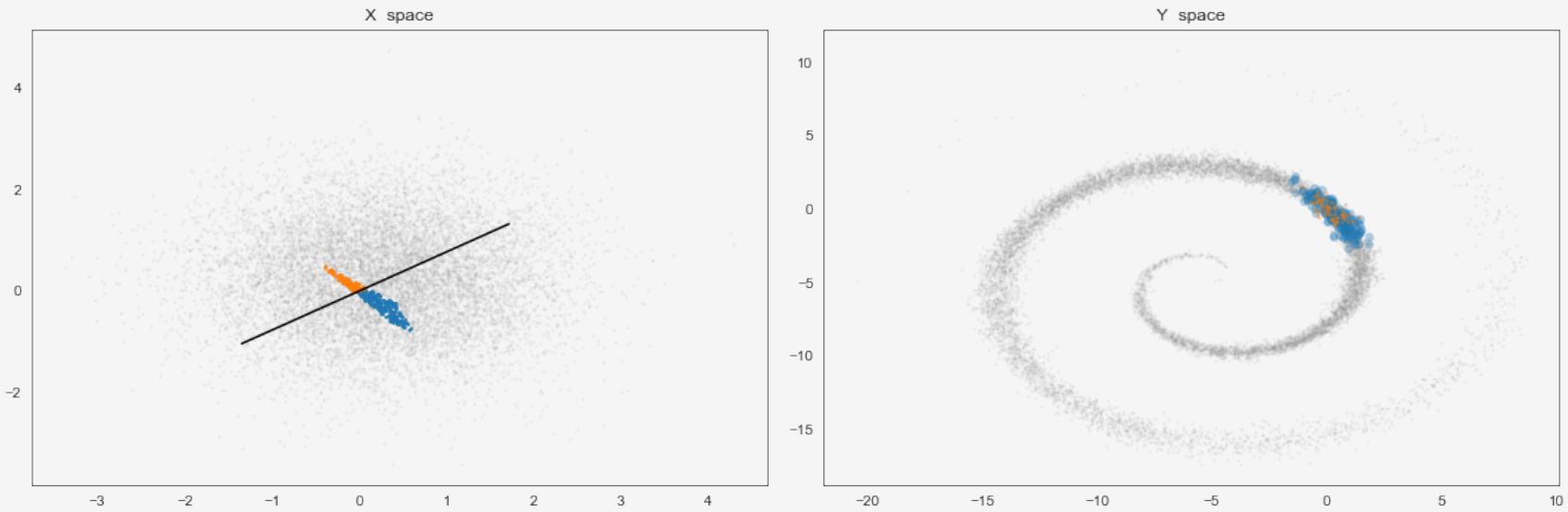

Figure: PCA

Toy demo

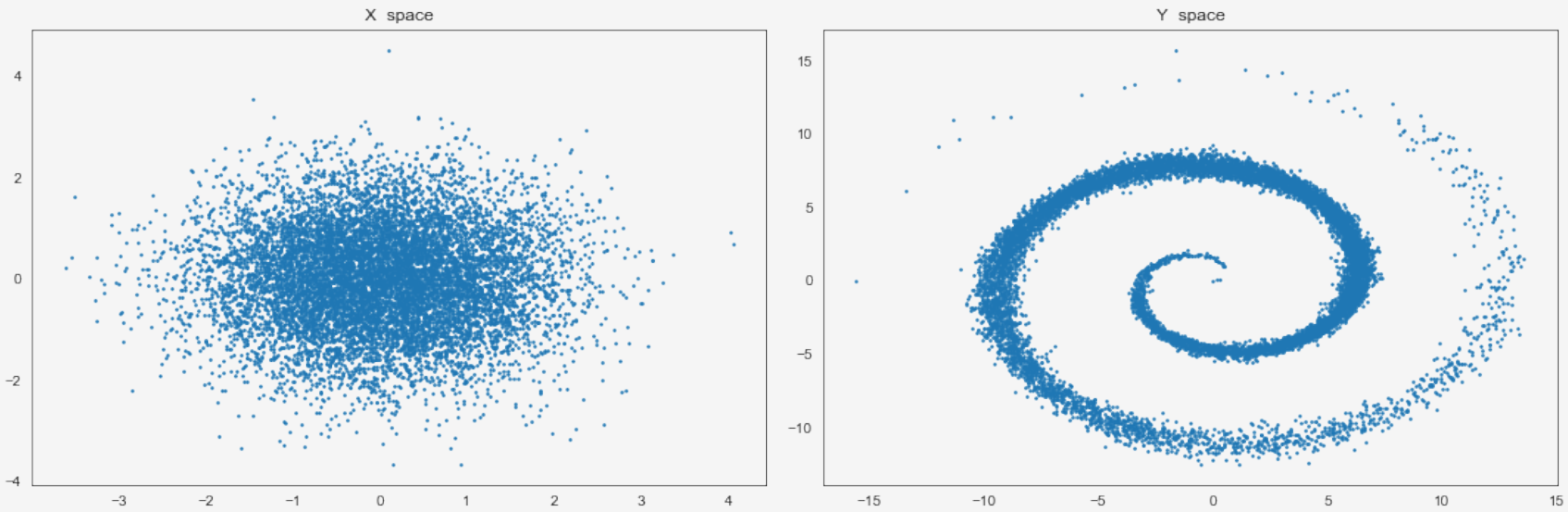

Figure: pls **NotImplementedError**

Toy demo

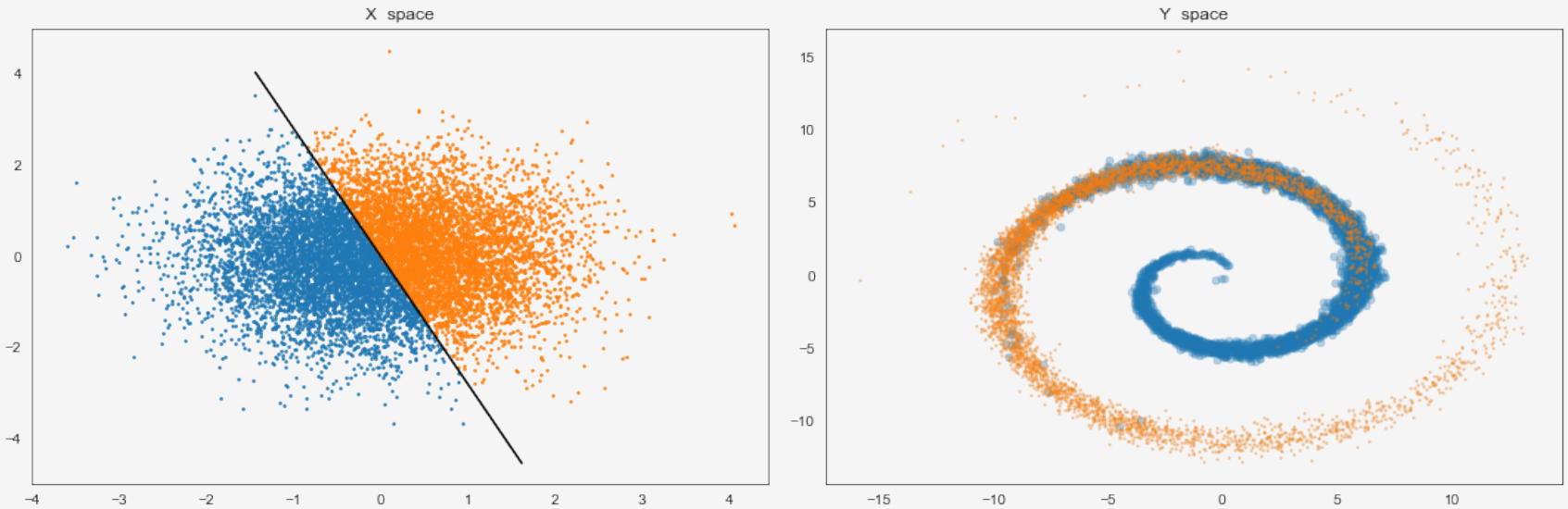

Figure: pls **NotImplementedError**

Toy demo

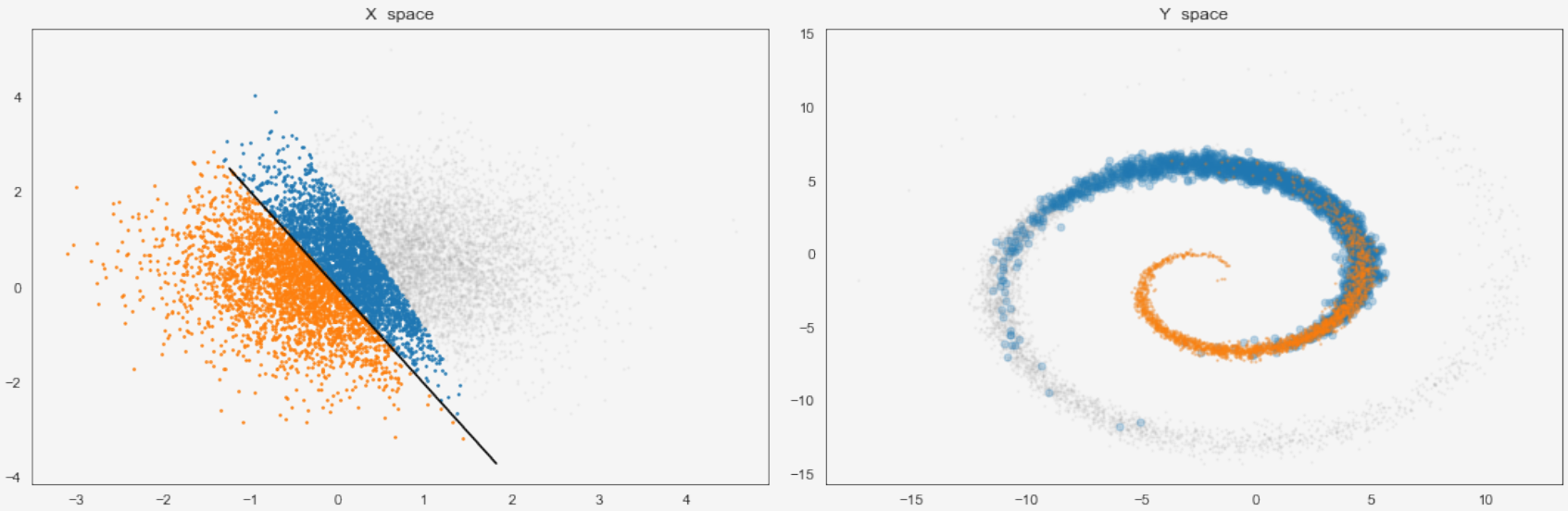

Figure: pls **NotImplementedError**

Toy demo

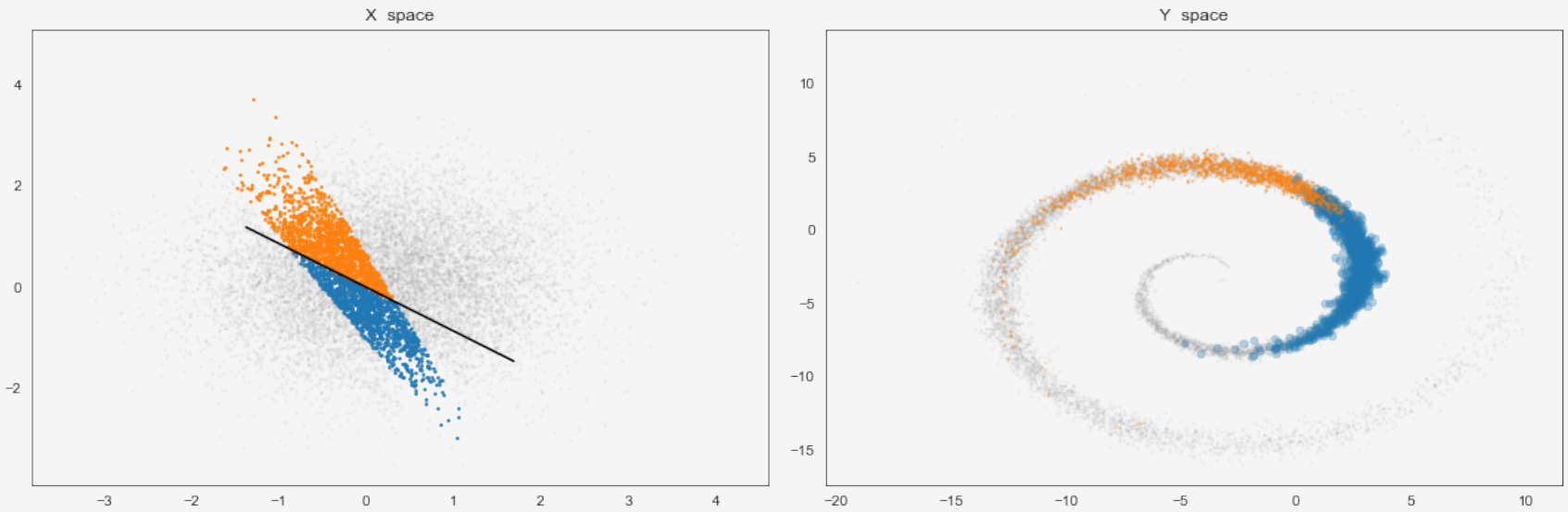

Figure: pls **NotImplementedError**

Toy demo

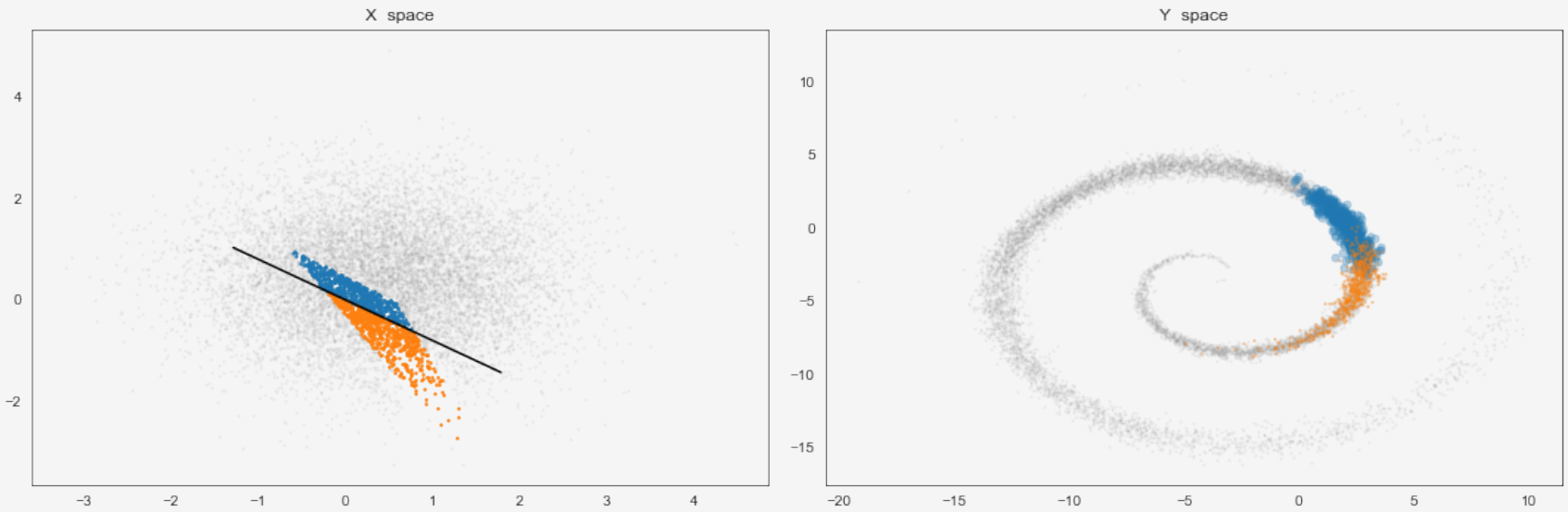

Figure: pls **NotImplementedError**

Real world experiments - error

method	mean	std	dataset
2MeansSVM	7.158	0.650	Concrete
Iso	1.831	0.542	Concrete
PCA	6.472	0.546	Concrete
2MeansSVM	2.611	0.281	AirFoil
Iso	0.564	0.215	AirFoil
PCA	3.724	0.207	AirFoil
2MeansSVM	3.747	0.139	Protein
Iso	1.049	0.317	Protein
PCA	2.860	0.217	Protein
2MeansSVM	118.624	10.641	ToyData
Iso	42.063	12.979	ToyData
PCA	78.573	11.230	ToyData
2MeansSVM	5.760	0.190	Parkinson
Iso	1.056	0.377	Parkinson
PCA	4.213	0.343	Parkinson

Table: Train error

Real world experiments - error

method	mean	std	dataset
2MeansSVM	6.916	0.788	Concrete
Iso	1.587	0.559	Concrete
PCA	6.295	0.528	Concrete
2MeansSVM	2.600	0.345	AirFoil
Iso	0.564	0.252	AirFoil
PCA	3.558	0.249	AirFoil
2MeansSVM	3.718	0.138	Protein
Iso	1.073	0.302	Protein
PCA	2.842	0.213	Protein
2MeansSVM	116.359	10.804	ToyData
Iso	40.518	11.907	ToyData
PCA	77.152	11.444	ToyData
2MeansSVM	5.608	0.200	Parkinson
Iso	1.021	0.393	Parkinson
PCA	4.153	0.355	Parkinson

Table: Validation error

Real world experiments - error

method	mean	std	dataset
2MeansSVM	8.461	0.985	Concrete
Iso	5.198	0.883	Concrete
PCA	8.867	1.046	Concrete
2MeansSVM	3.038	0.363	AirFoil
Iso	1.889	0.284	AirFoil
PCA	4.559	0.336	AirFoil
2MeansSVM	4.290	0.149	Protein
Iso	3.835	0.345	Protein
PCA	4.203	0.224	Protein
2MeansSVM	143.372	11.906	ToyData
Iso	122.088	16.475	ToyData
PCA	125.165	13.071	ToyData
2MeansSVM	6.641	0.210	Parkinson
Iso	3.631	0.493	Parkinson
PCA	6.393	0.368	Parkinson

Table: Test error

Real world experiments - α

method	mean	std	dataset
2MeansSVM	0.119	0.002	Concrete
Iso	0.255	0.009	Concrete
PCA	0.116	0.003	Concrete
2MeansSVM	0.133	0.003	AirFoil
Iso	0.246	0.001	AirFoil
PCA	0.072	0.003	AirFoil
2MeansSVM	0.046	0.000	Protein
Iso	0.139	0.000	Protein
PCA	0.060	0.000	Protein
2MeansSVM	0.070	0.001	ToyData
Iso	0.153	0.013	ToyData
PCA	0.094	0.001	ToyData
2MeansSVM	0.037	0.000	Parkinson
Iso	0.199	0.004	Parkinson
PCA	0.057	0.002	Parkinson

Table: α