Protecting Your Machine Learning Against Drift: An Introduction

Oliver Cobb - Applied Machine Learning Researcher

ALIBI DETECT

SELDON

We will look at

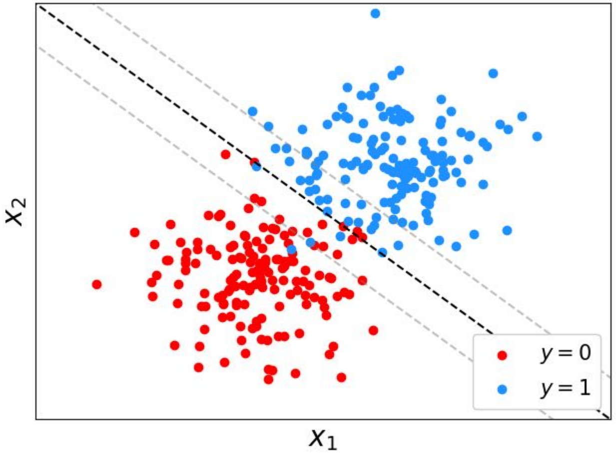
- What drift is and why it pays detect it.
- The different types of drift.
- How drift can be detected in a principled manner.
- The anatomy of a drift detector.
- Demystify concepts such as ‘online detectors’, ‘permutation tests’, ‘MMD test’ etc.
- Practical demonstration with alibi-detect.

Preliminaries

- Wish to use y for some prediction/output.
- Can't observe y, can observe $x=(x_1,\ldots,x_d)$ that are related.
- We fit a model M to predict y from x and use $\hat{y} = M(x)$ as the prediction/output.

- Performance on held out data gives estimate for future performance…

  <u>Assuming the process underlying x and y remains constant.</u>



What is drift?

- When the process underlying x and y during deployment differs from the process that generated the training data.
- Can no longer expect the model's performance during deployment to match that observed on held out training data.

p(x,y)

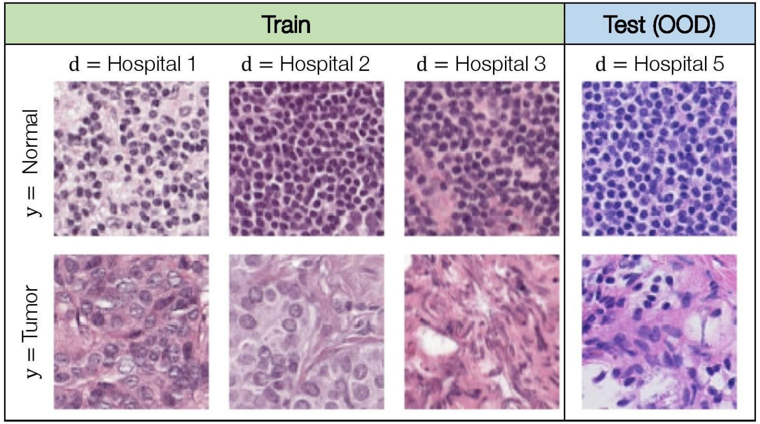
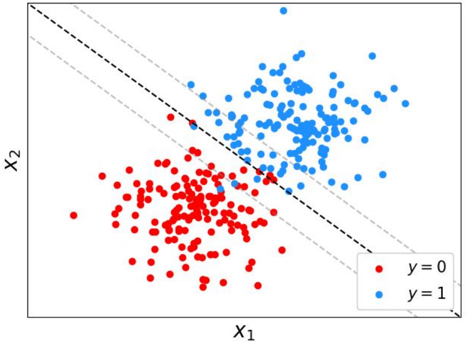


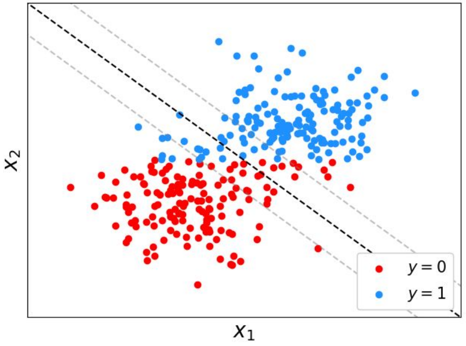
Image credit: https://camelyon17.grand-challenge.org/ & https://wilds.stanford.edu/

What is drift?

$p(x,y) = p(y|x)p(x) = p(x|y)p(y)$

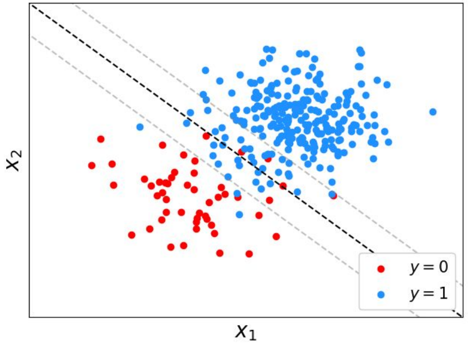


No Drift



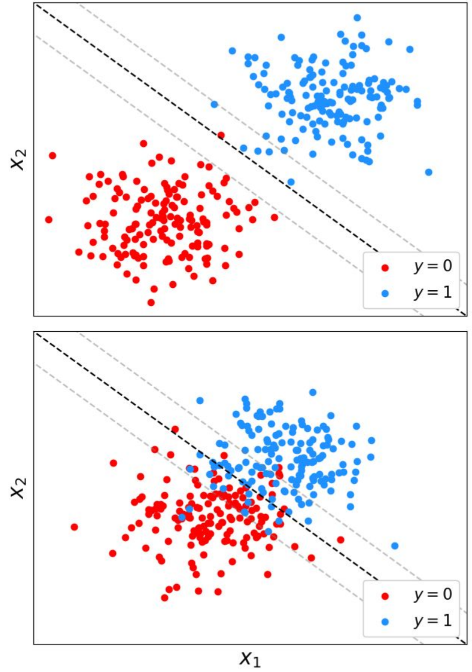
Concept Drift

$\triangle$ p(y|x)



Prior Drift

$\triangle$ p(y)



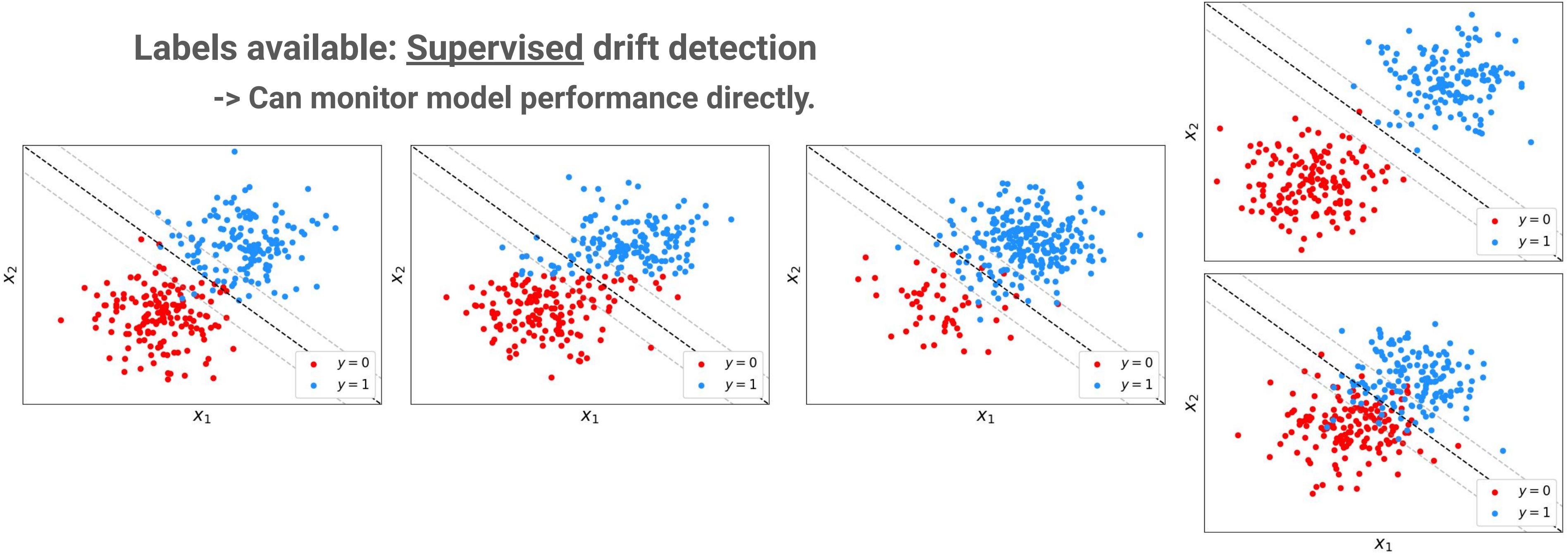
Covariate Drift

$\triangle$ p(x)

What is drift?

Labels available: <u>Supervised</u> drift detection

-> Can monitor model performance directly.



No Drift

Concept Drift

$\triangle$ p(y|x)

Prior Drift

$\triangle$ p(y)

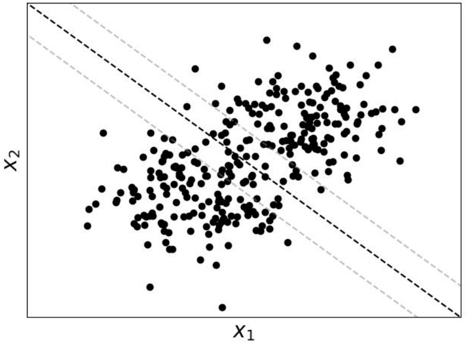
Covariate Drift

$\triangle$ p(x)

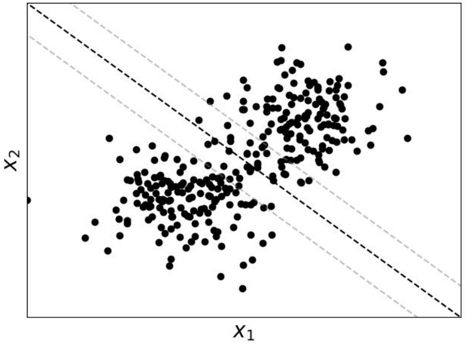
What is drift?

Labels unavailable: <u>Unsupervised</u> drift detection

-> Can't monitor model performance. Must operate in high dimensions.

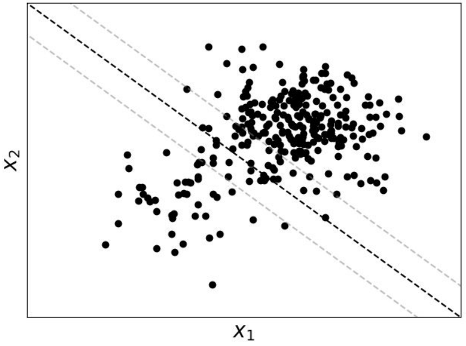


No Drift



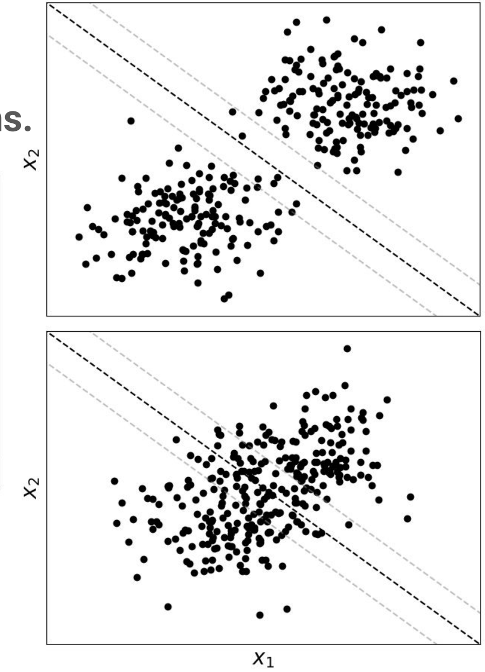
Concept Drift

△ p(y|x)



Prior Drift

△ p(y)

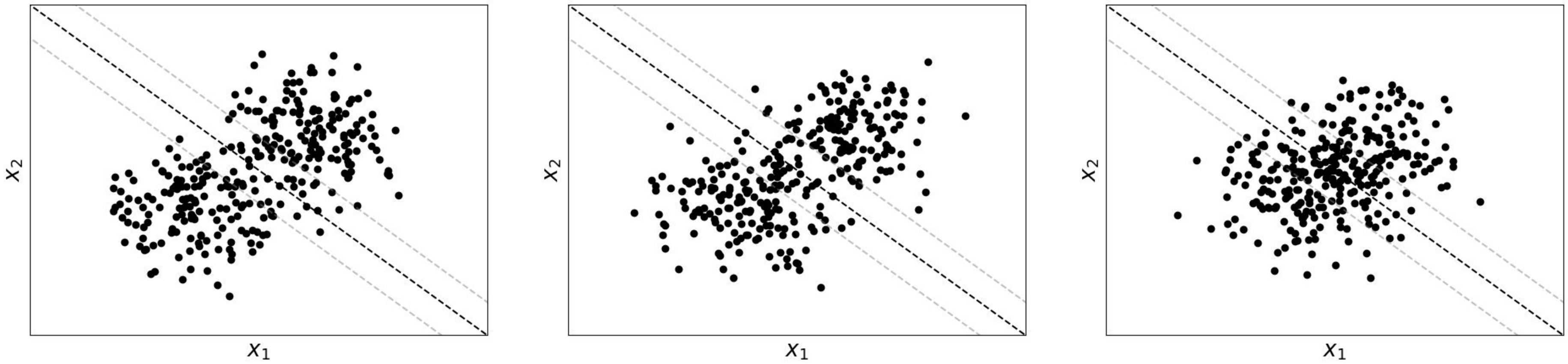


Covariate Drift

△ p(x)

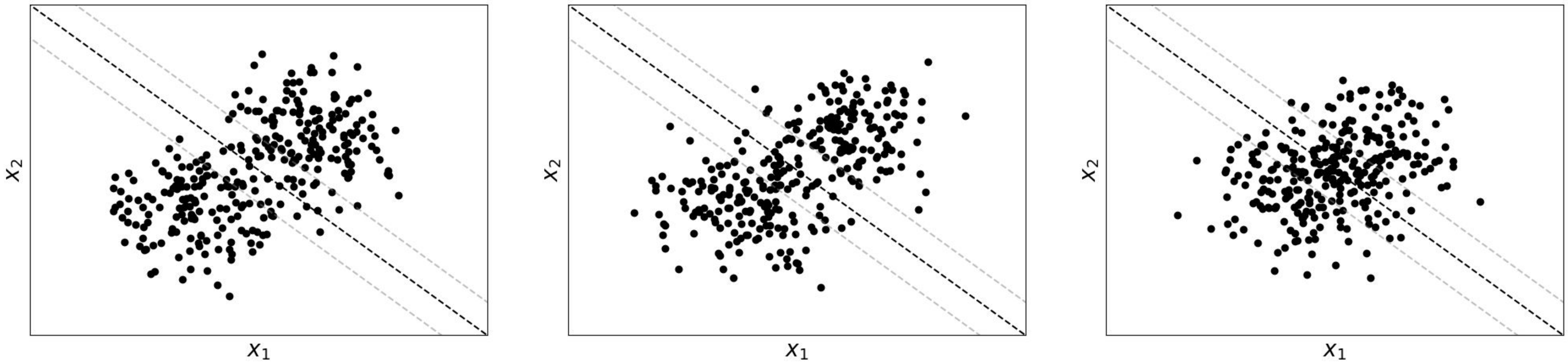
Change or chance?

- We don’t expect new data to look identical to training data.
- So how do we differentiate systemic change from natural fluctuations?
- Statistical hypothesis testing!



Statistical Hypothesis Testing

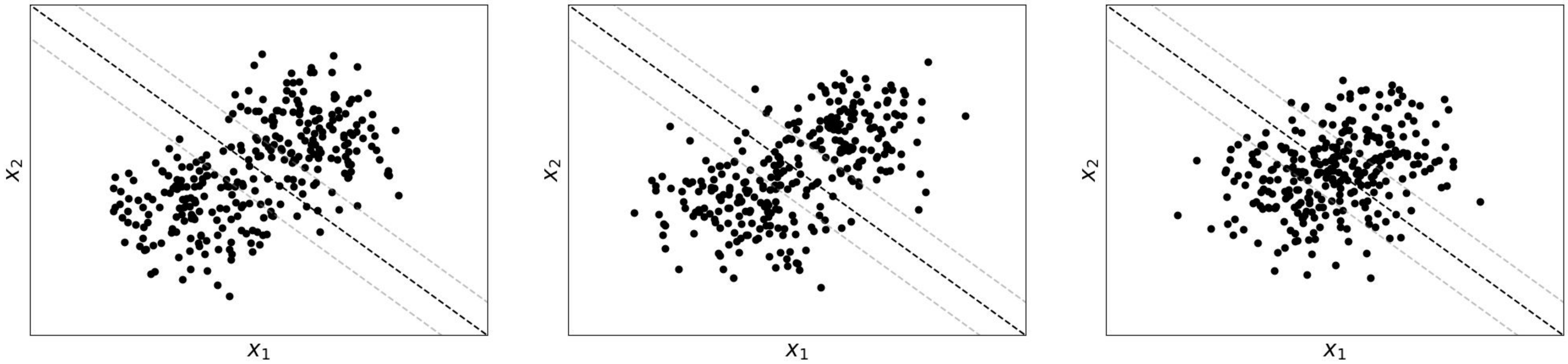
- Before observing data Z, specify null and alternative hypothesis, H_0 and H_1.
- Specify test statistic S(Z) we expect to be small if H_0 and large if H_1.
- Observe data, compute S(Z), compute $\hat{p}$ = P(such an extreme S(Z) | H_0).
- Low p-value discredits H_0.
    - Typically specify a threshold p (FPR) in advance and reject null if $\hat{p} < p$.



Offline Drift detection

- Let q_0 be distribution underlying training data, Z_0.
- Let q_1 be the distribution underlying a batch of new data, Z_1.
- H_0: $q_0=q_1$. H_1: $q_0 \neq q_1$.
- $\boxed{S(Z_0,Z_1) \text{ small if } H_0 \text{ true, large if } H_1 \text{ true.}}$
- $\boxed{\text{Compute } \hat{p}.}$ Flag drift if $\hat{p} < p$ for desire FPR p.

The hard part!

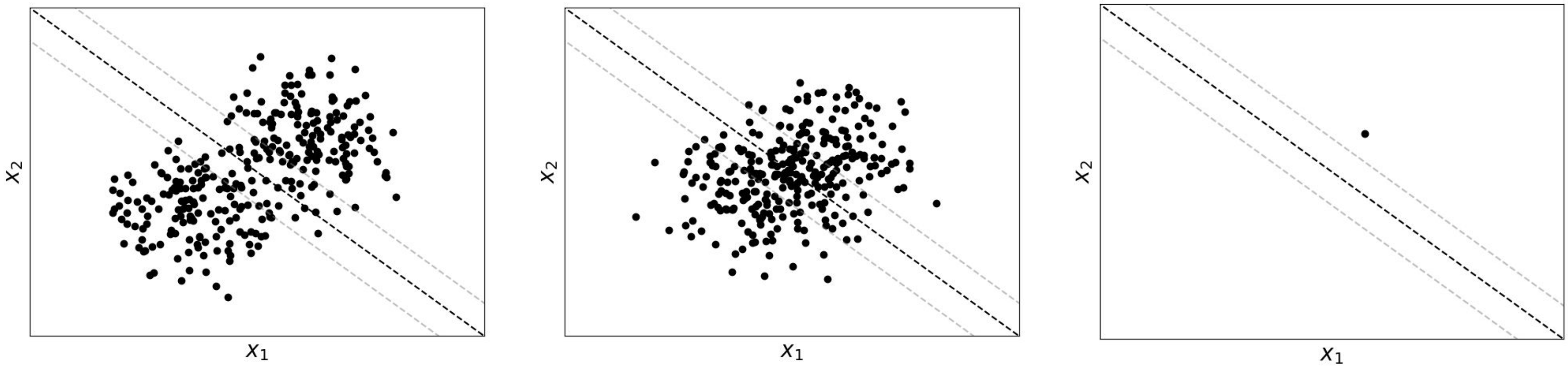


Online Drift detection

- Data points $z=(x,y)$ arrive in sequence, $z_1, z_2, \ldots$ and we'd like to detect drift ASAP.
- Assumption:

$$z_i \sim \begin{cases} q_0 & \text{for } i < T^* \\ q_1 & \text{for } i \geq T^* \end{cases}$$

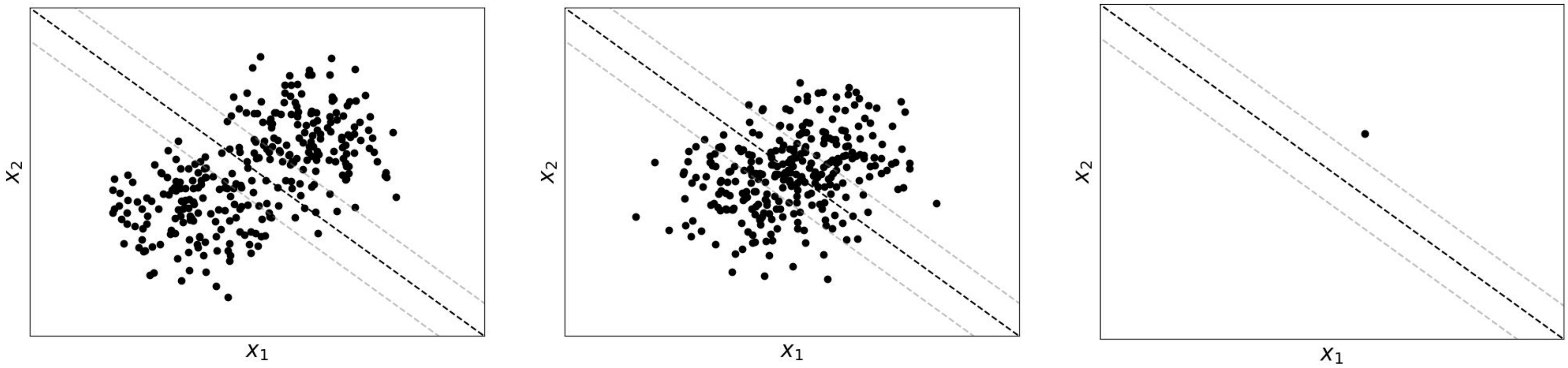
- At each time t we perform a hypothesis test of $\{H_0: T^* > t\}$ vs $\{H_1: T^* \leq t\}$.
  - i.e. "has drift occurred yet"?



Online Drift detectors - desired properties

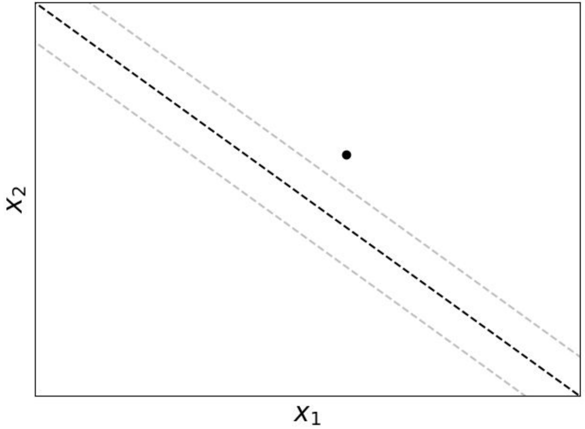
- When a change occurs the detector is fast to respond.
  - i.e. Expected Detection Delay, EDD = E[T' - T*], is small.
- Ability to specify the frequency of false detections in the absence of change.
  - i.e. We can specify Expected Run Time, ERT = E[T' | T*= ∞]
- There's an ERT vs EDD tradeoff.

Often overlooked



Windowing Strategies

- How do we apply SHT to data arriving sequentially?
- By collecting instances into “test” windows.
- These can then compared to the fixed “reference” window.
- Windows can be fixed sized and disjoint, fixed size and overlapping or adaptive.



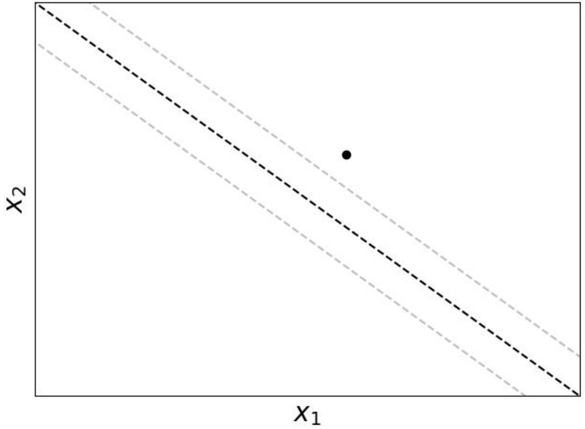
Disjoint

t	1	2	3	4	5	6	7	8	9	10
x_1	0.21	1.51	-1.03	-0.08	1.46	0.80	1.13	0.17	-0.35	-0.12
x_2	-0.41	0.05	0.59	-0.94	0.73	0.74	-1.52	0.66	-1.19	-0.72

TEST! TEST!

Windowing Strategies

- How do we apply SHT to data arriving sequentially?
- By collecting instances into “test” windows.
- These can then compared to the fixed “reference” window.
- Windows can be fixed sized and disjoint, fixed size and overlapping or adaptive.



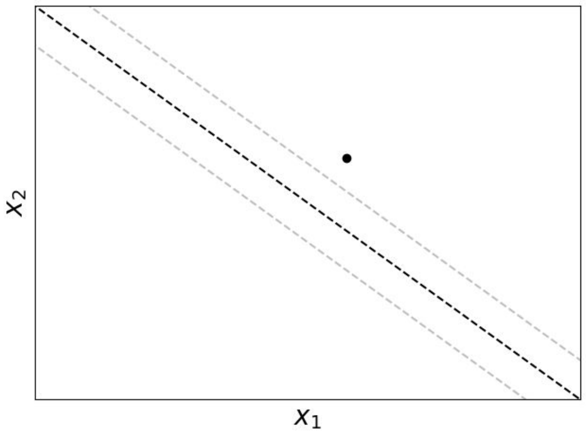
Overlapping

t	1	2	3	4	5	6	7	8	9	10
x_1	0.21	1.51	-1.03	-0.08	1.46	0.80	1.13	0.17	-0.35	-0.12
x_2	-0.41	0.05	0.59	-0.94	0.73	0.74	-1.52	0.66	-1.19	-0.72

TEST!TEST!TEST!

Windowing Strategies

- How do we apply SHT to data arriving sequentially?
- By collecting instances into “test” windows.
- These can then compared to the fixed “reference” window.
- Windows can be fixed sized and disjoint, fixed size and overlapping or adaptive.



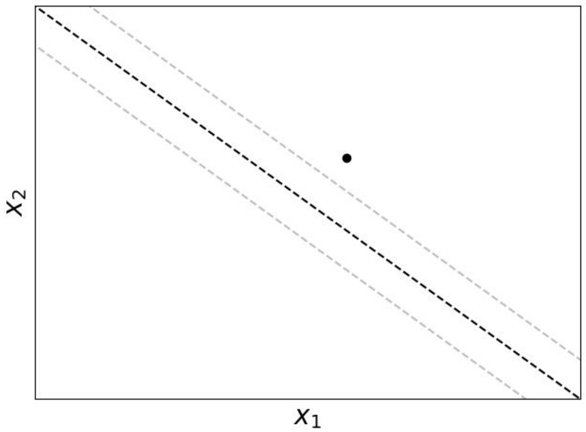
Adaptive

t	1	2	3	4	5	6	7	8	9	10
x_1	0.21	1.51	-1.03	-0.08	1.46	0.80	1.13	0.17	-0.35	-0.12
x_2	-0.41	0.05	0.59	-0.94	0.73	0.74	-1.52	0.66	-1.19	-0.72

TEST! TEST! TEST! TEST!
KEEP? KEEP? KEEP? KEEP?

Windowing Strategies

- Different windowing strategies call for different test statistics and threshold determination processes.
- They also each have their own pros and cons.



t	1	2	3	4	5	6	7	8	9	10
x_1	0.21	1.51	-1.03	-0.08	1.46	0.80	1.13	0.17	-0.35	-0.12
x_2	-0.41	0.05	0.59	-0.94	0.73	0.74	-1.52	0.66	-1.19	-0.72

Disjoint Window Detectors

- **Tests are independent and performed infrequently.**

- **Can compute p-value corresponding to <u>any test statistic</u> $S(Z_0,Z_1)$!**
  Achieved via a permutation test:
  - shuffle(Z_0, Z_1) : $(Z_0, Z_1) \mapsto (Z^*_0, Z^*_1)$
  - alt_stats = [S(shuffle(Z_0,Z_1)) for _ in range(B)]
  - $\hat{p}$ = (alt_stats > S(Z_0,Z_1)).mean()
- **<u>Can learn</u> new <u>test statistic</u> S for each test.**
- **Sensitive to choice of window size.**
- **Slow to respond to severe drift.**

Overlapping Window Detectors

- Test statistics are correlated.
- Makes controlling ERT very difficult.

- Test statistic must be:
    - Incremental
    - Pre-specified
- Computationally light.
- Fast to respond to severe drift.

Adaptive Window Detectors

- Typically accumulate some notion out ‘outlierness’.
- Hard to control ERT.

- Adaptive window size.
- Accumulating ‘outlierness’ not good for EDD!
- Why not?

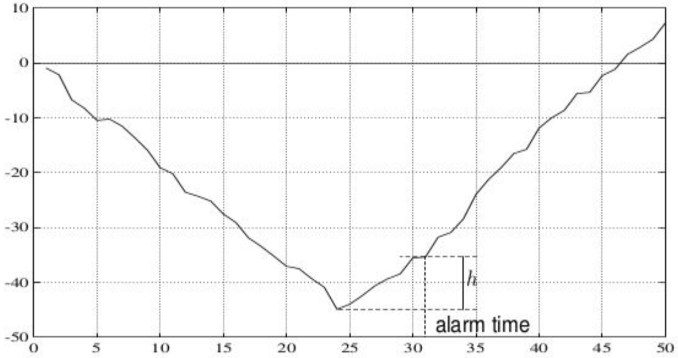
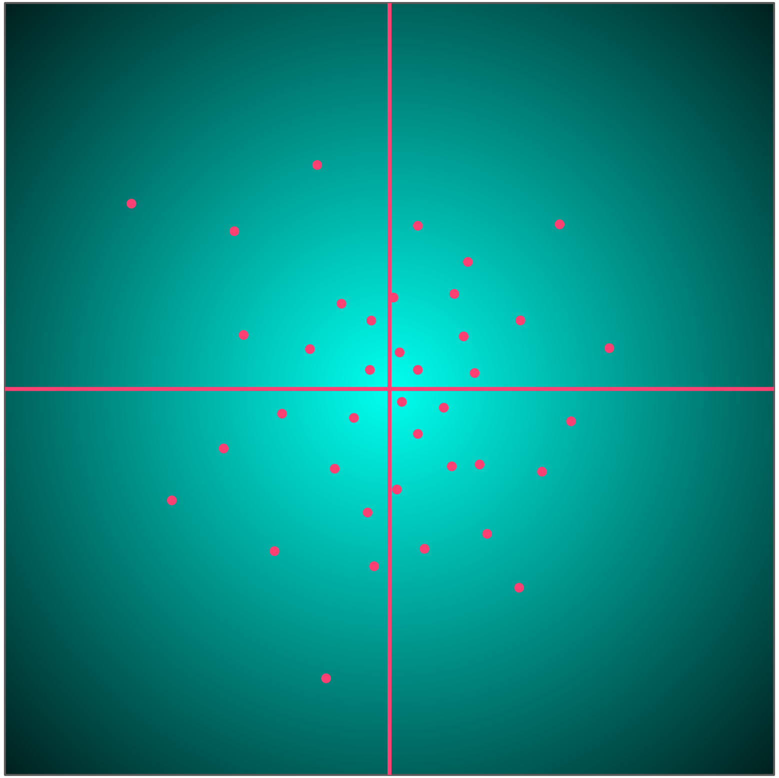


Image credit: Basseville and Nikiforov (1993) - Detection of Abrupt Changes: Theory and Application

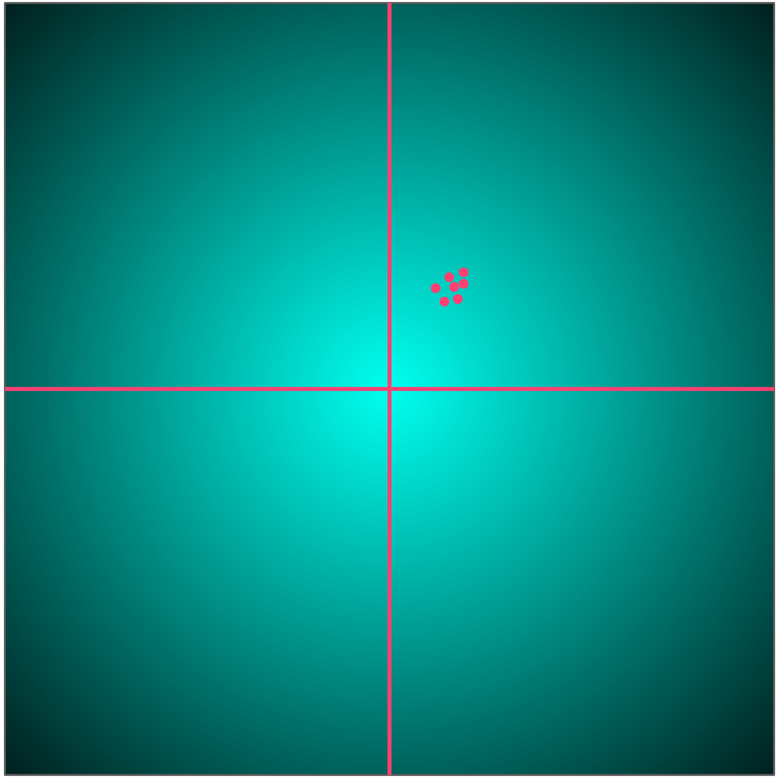
Drift = persistent outliers?

$q_0 = N(0,I_2)$



Drift = persistent outliers?

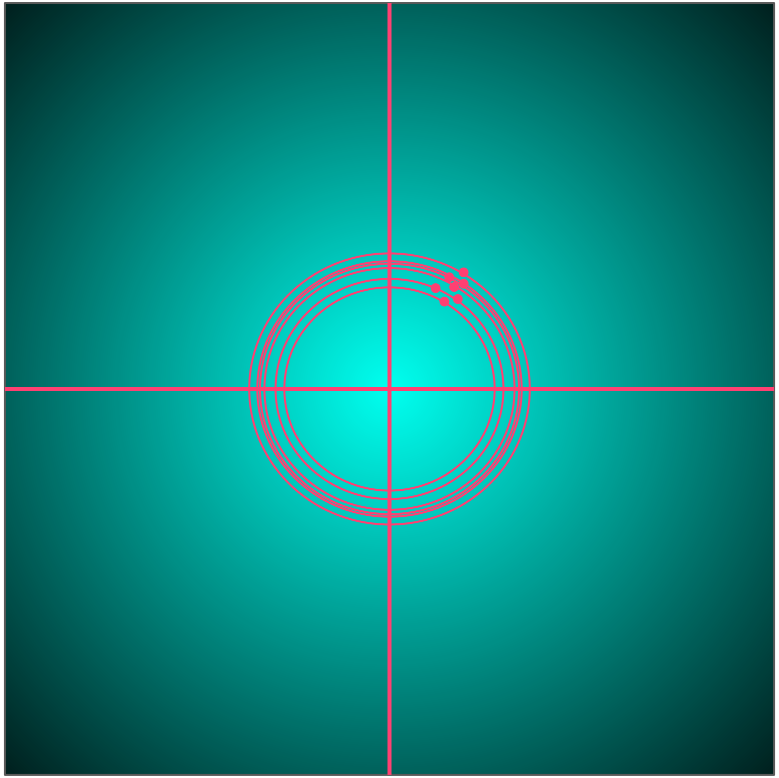
$q_0 = N(0, I_2)$



$q_1 = N(\mu, \sigma I_2)$

Drift = persistent outliers?

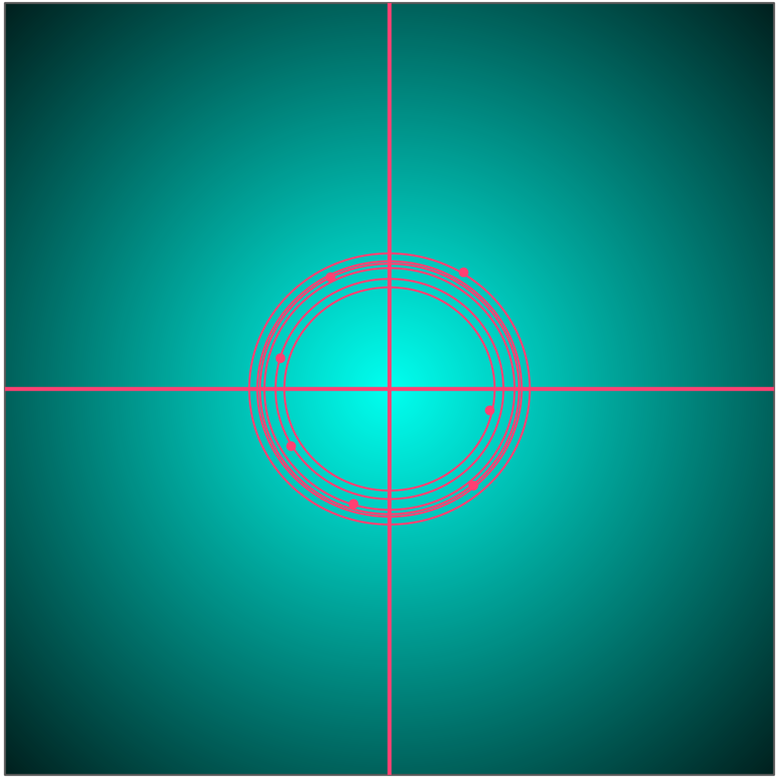
$q_0 = N(0, I_2)$



$q_1 = N(\mu, \sigma I_2)$

Drift = persistent outliers?

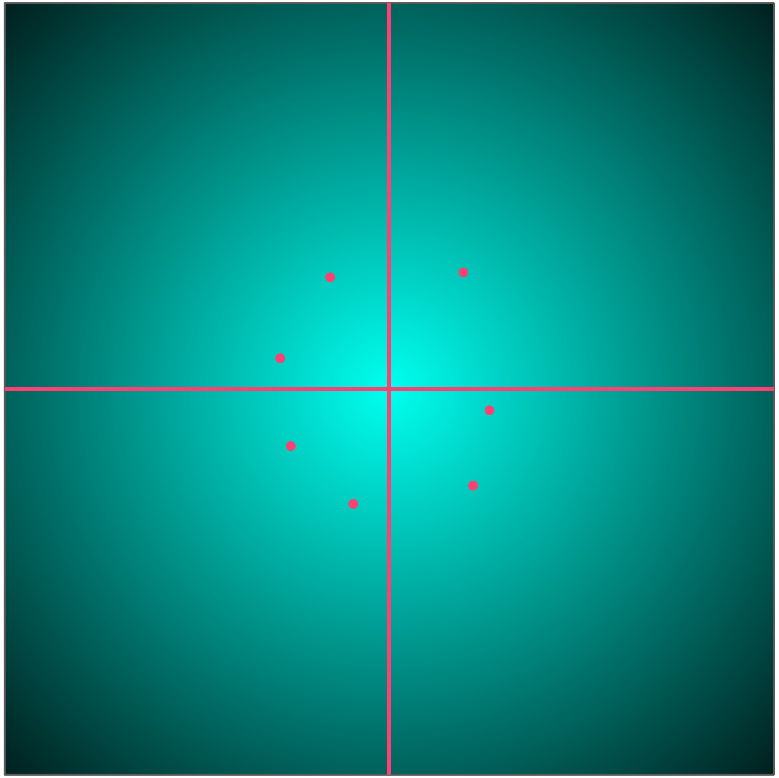
$q_0 = N(0, I_2)$



$q_1 = N(\mu, \sigma I_2)$

Drift = persistent outliers?

$q_0 = N(0, I_2)$



$q_1 = N(\mu, \sigma I_2)$

Test Statistics

- Outlierness-based test statistics are not sufficient - so what is?
- Test statistics which estimate distance between q_0 and q_1.
- Could first estimate $\hat{q}_0$ and $\hat{q}_1$ and evaluate $d(\hat{q}_0,\hat{q}_1)$.
- More efficient to directly estimate $d(q_0,q_1)$.
- $d(q_0,q_1) = MMD_k(q_0,q_1)$ features prominently in alibi-detect.

Maximum Mean Discrepancy (MMD_k)

- Transforms problem from specifying a distance $d(q_0,q_1)$ between distributions to specifying a similarity (kernel) $k(z_0,z_1)$ between data points.
- Typically $k(z_0,z_1)=\Phi(z_0)^T\Phi(z_1)$ for some projection Φ.

- $MMD_k(q_0,q_1) = E[k(z_0,z'_0) + k(z_1,z'_1) - 2k(z_0,z_1)]$

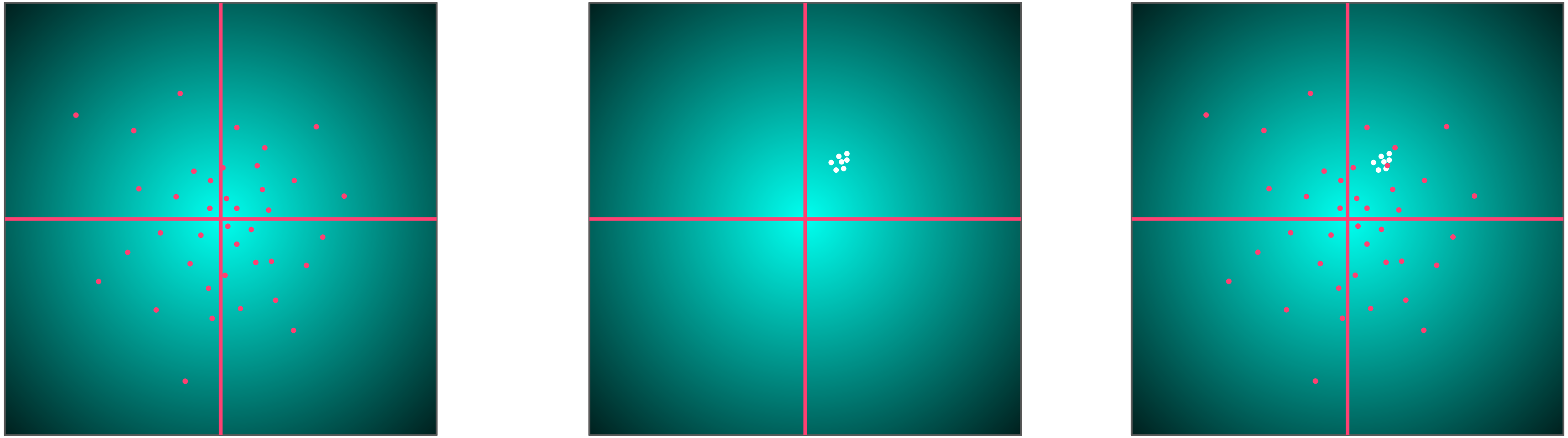
- $\widehat{MMD}_k(q_0,q_1)$ = Avg. similarity between reference instances
  + Avg. similarity between test instances
  - 2*Avg. similarity between reference and test instances

$$\frac{1}{n_x(n_x-1)}\sum_{x_1\neq x_2\in\mathcal{X}}k(x_1,x_2)+\frac{1}{n_y(n_y-1)}\sum_{y_1\neq y_2\in\mathcal{Y}}k(y_1,y_2)-\frac{2}{n_xn_y}\sum_{(x,y)\in\mathcal{X}\times\mathcal{Y}}k(x,y)$$

Maximum Mean Discrepancy (MMD_k)

$\widetilde{MMD}_k(q_0,q_1)$ = Avg. similarity between reference instances (small)
+ Avg. similarity between test instances (large)
- 2*Avg. similarity between reference and test instances (small)

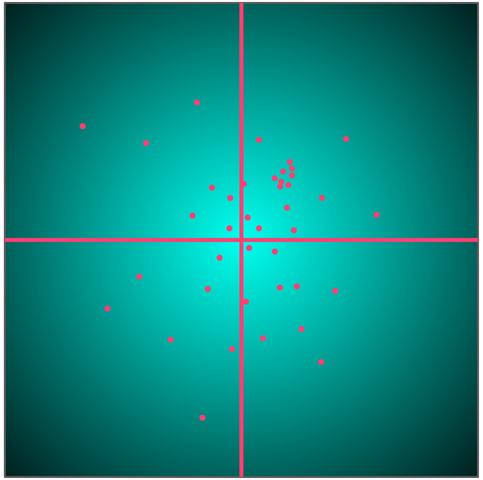
= large (e.g. 0.1 + 0.8 - 2*0.12 = 0.68)

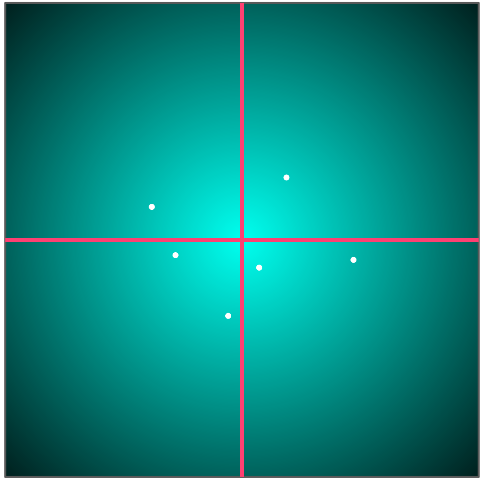


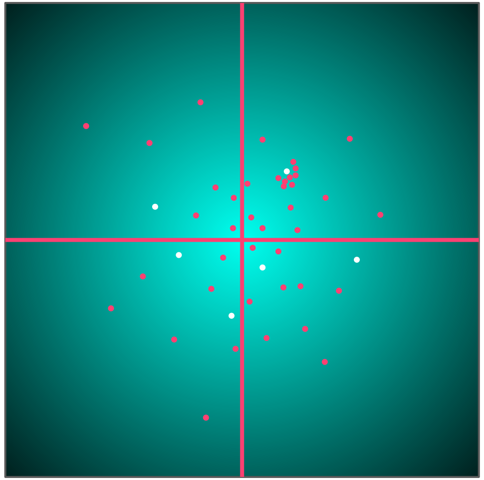
Maximum Mean Discrepancy (MMD_k) - Permuted

$\widehat{MMD}_k(q_0,q_1)$ = Avg. similarity between reference instances (small)
+ Avg. similarity between test instances (small)
- 2*Avg. similarity between reference and test instances (small)

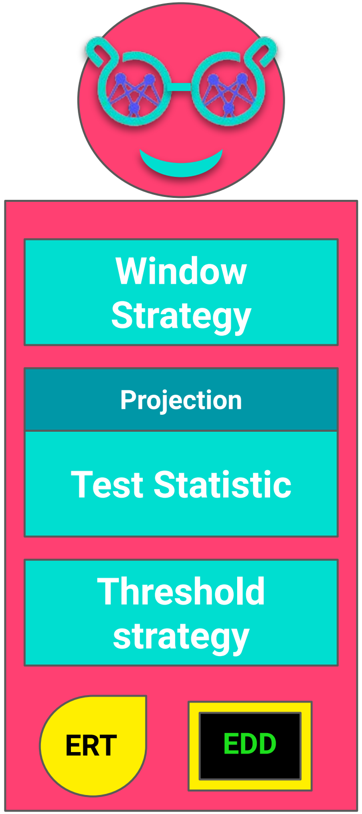
= small (e.g. 0.09 + 0.11 - 2*0.09 = 0.02)







Summary and demo



ALIBI DETECT

Drift Detection

Detector	Tabular	Image	Time Series	Text	Categorical Features	Online	Feature Level
Kolmogorov-Smirnov	✓	✓		✓	✓		✓
Maximum Mean Discrepancy	✓	✓		✓	✓	✓	
Least-Squares Density Difference	✓	✓		✓	✓	✓	
Chi-Squared	✓				✓		✓
Mixed-type tabular data	✓				✓		✓
Classifier	✓	✓	✓	✓	✓		
Classifier Uncertainty	✓	✓	✓	✓	✓		
Regressor Uncertainty	✓	✓	✓	✓	✓		

github.com/SeldonIO/alibi-detect

Thanks for watching!

github.com/SeldonIO/alibi-detect

@SeldonResearch

oc@seldon.io