COMPSCI 361 - Tutorial 12

Ben Halstead

Semester 1, Week 11, 2021

Streaming data vs Static data

- So far in the course we have looked at learning from a *static* training set, then making predictions on a *static* test set.
- In assignment 3, we *trained* a model on a *training set* once, then make predictions on the entire *test* set at once.
- This is the most common format for machine learning, but is it always realistic?

Streaming data vs Static data

- So far in the course we have looked at learning from a *static* training set, then making predictions on a *static* test set.
- In assignment 3, we *trained* a model on a *training set* once, then make predictions on the entire *test* set at once.
- This is the most common format for machine learning, but is it always realistic?
- Often in the real world, we have continuous streams of data, which we want to both learn from and predict.
- Data stream mining adapts the machine learning techniques we have learnt to this streaming scenario.

Examples of data streams

- Machine learning on web data - Predicting behaviour of real time amazon web orders
- Machine learning on IoT sensors - Real time weather sensors etc
- Machine learning for finance - Real time stock price data
- Increasingly many sources of data are not just *datasets*, but continuous streams of data

Change - A motivating example

- A simple (but possibly naive) way to handle streaming data would be:
  1. Collect a training set made up of the first n observations received
  2. Train a model
  3. Use the model to make predictions for the rest of the stream.
- Does this strategy work?

Change - A motivating example

- A simple (but possibly naive) way to handle streaming data would be:
  1. Collect a training set made up of the first n observations received
  2. Train a model
  3. Use the model to make predictions for the rest of the stream.
- Does this strategy work?
- What if the training set isn't enough to get good performance? What if we need to predict the first observations as well? What if the incoming data changes over time?

Change - A motivating example

Lets start with a problem - change in the incoming data can break our static model:

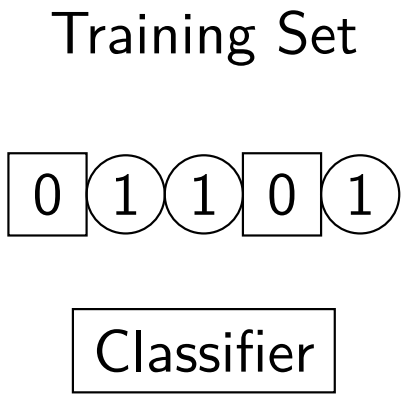


Figure: The first elements in the stream are collected to train on.

Change - A motivating example

Lets start with a problem - change in the incoming data can break our static model:

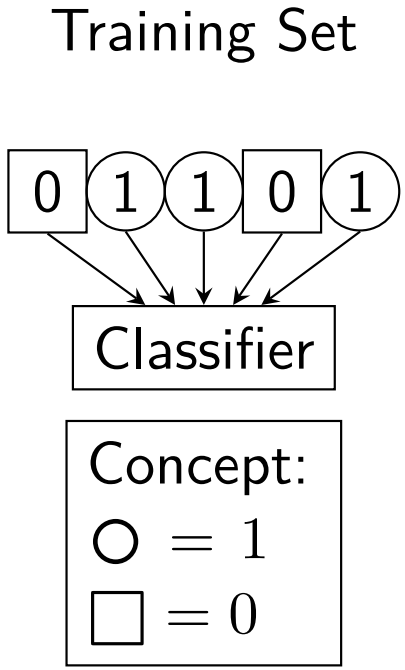


Figure: The classifier learns a concept from the training distribution.

Change - A motivating example

Lets start with a problem - change in the incoming data can break our static model:

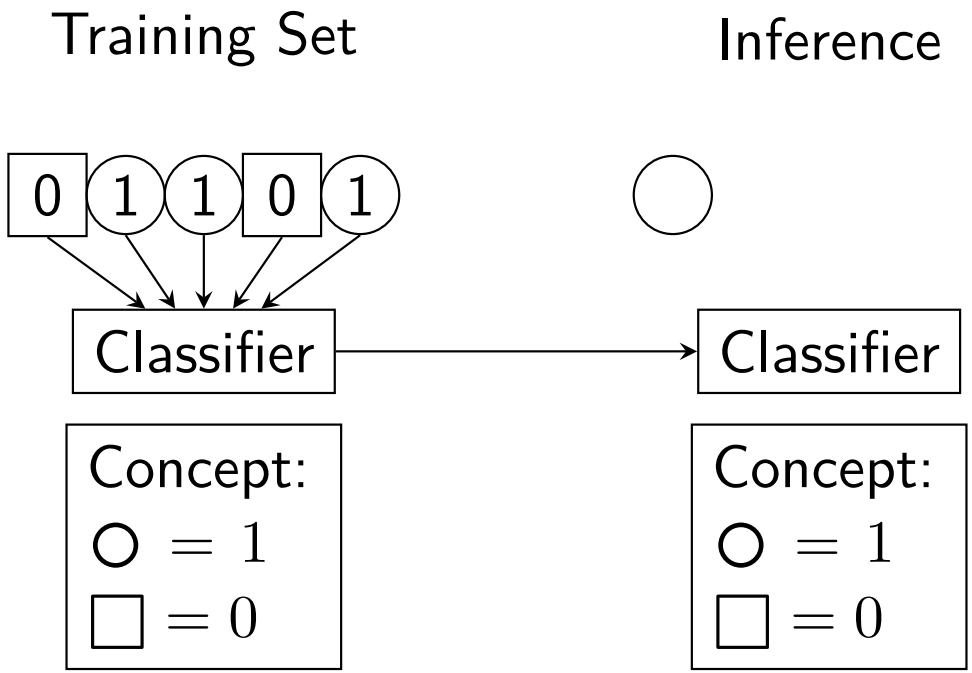


Figure: At inference time, we use the trained classifier to classify new observations.

Change - A motivating example

Lets start with a problem - change in the incoming data can break our static model:

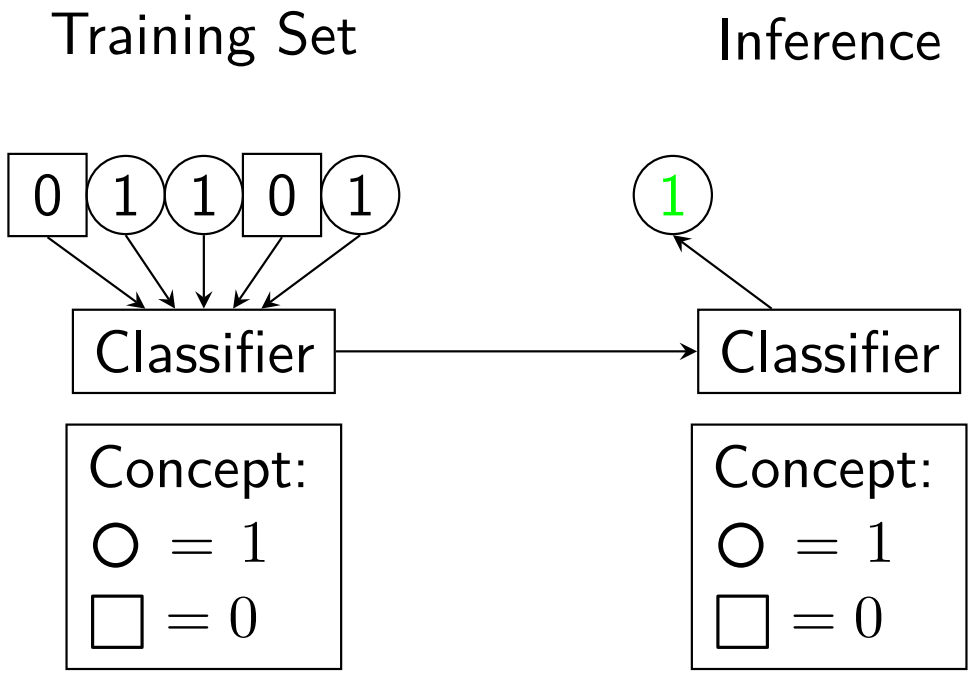


Figure: The classifier works well if the concept it has learned matches the new distribution

Change - A motivating example

Lets start with a problem - change in the incoming data can break our static model:

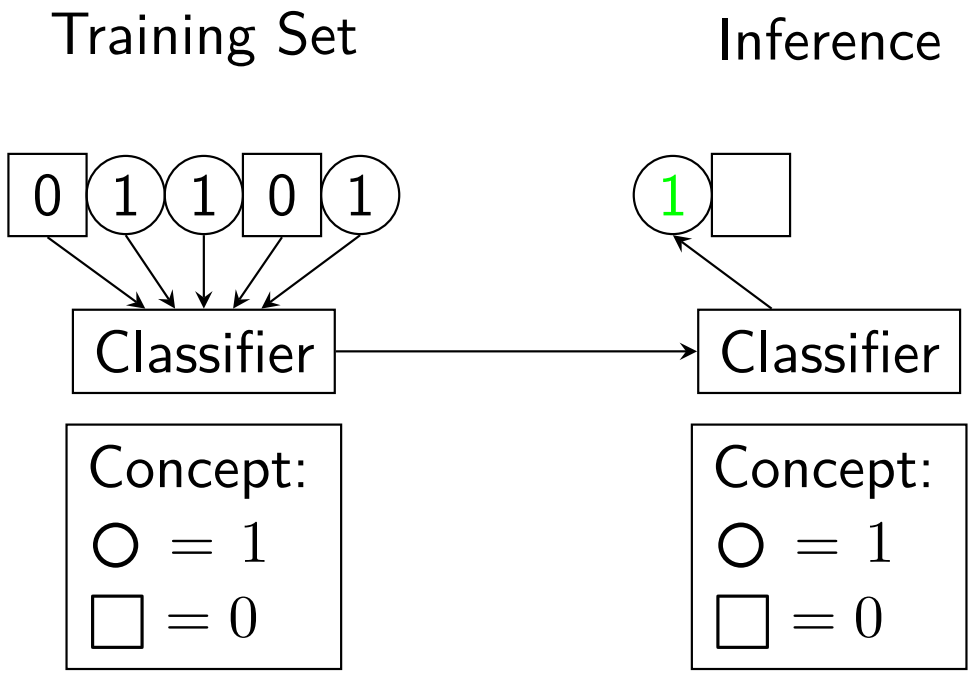


Figure: The classifier works well if the concept it has learned matches the new distribution

Change - A motivating example

Lets start with a problem - change in the incoming data can break our static model:

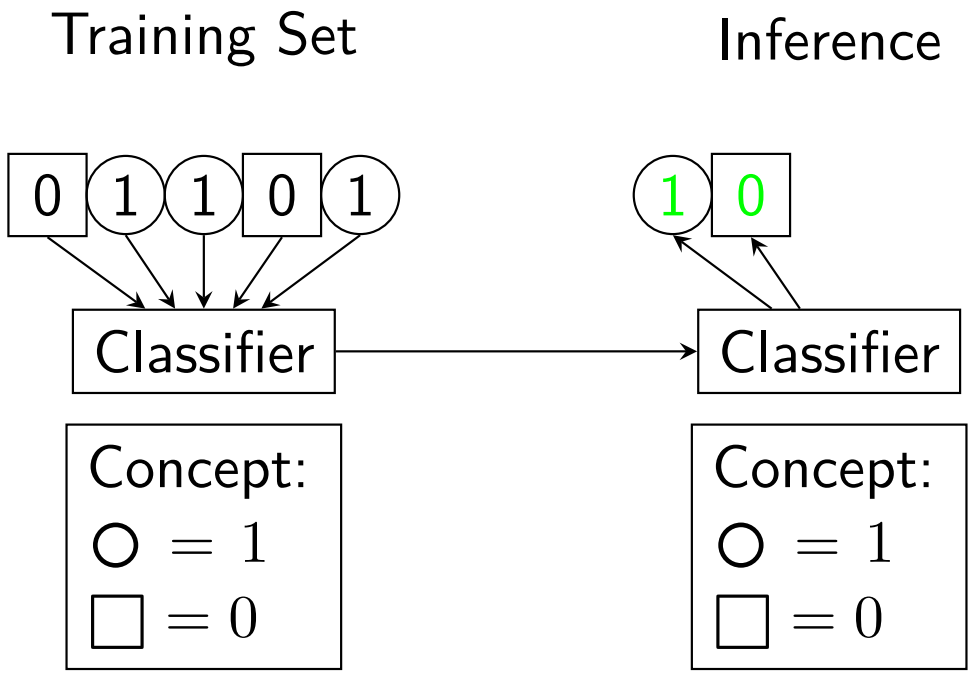


Figure: The classifier works well if the concept it has learned matches the new distribution

Change - A motivating example

Lets start with a problem - change in the incoming data can break our static model:

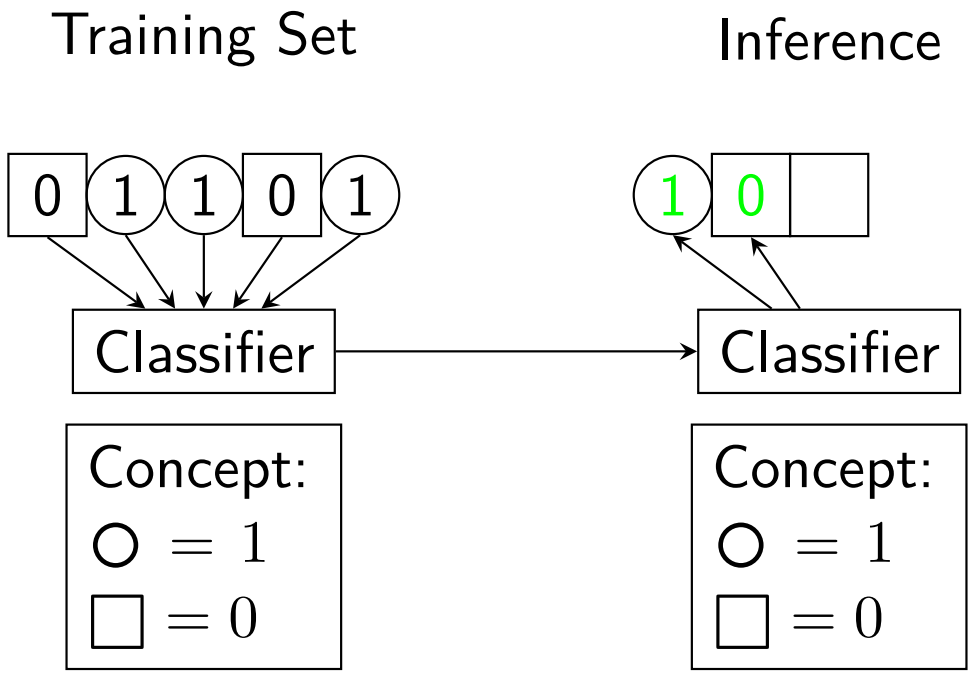


Figure: The classifier works well if the concept it has learned matches the new distribution

Change - A motivating example

Lets start with a problem - change in the incoming data can break our static model:

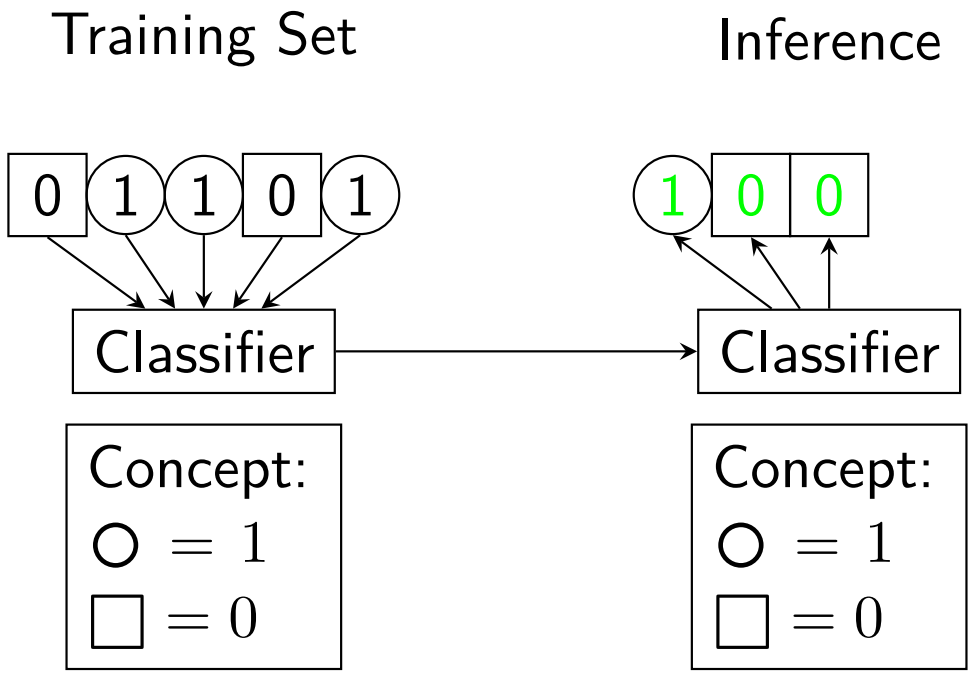


Figure: The classifier works well if the concept it has learned matches the new distribution

Change - A motivating example

Lets start with a problem - change in the incoming data can break our static model:

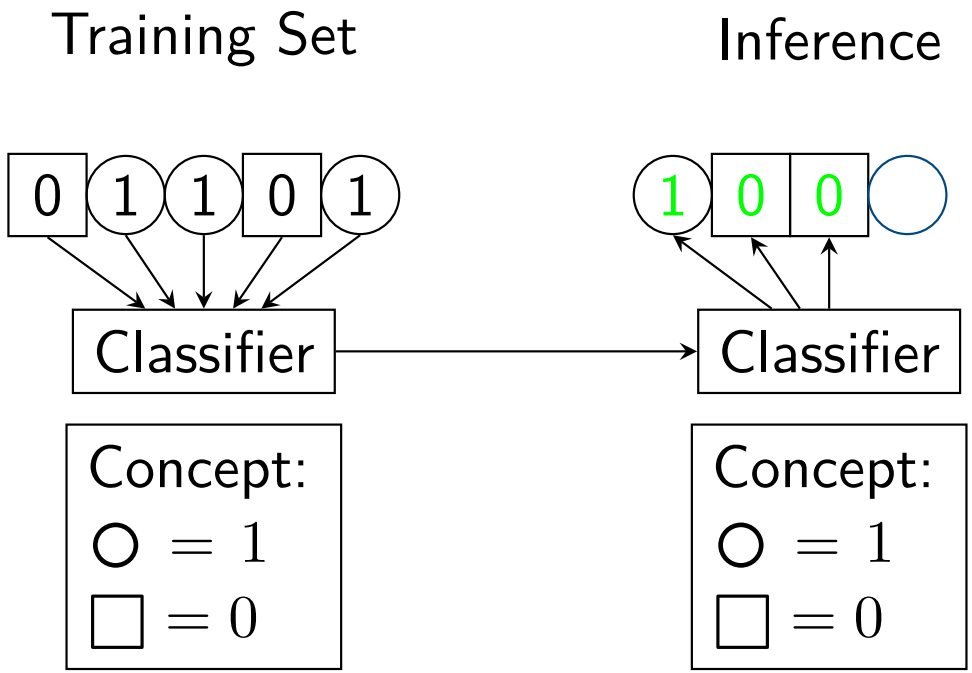


Figure: The classifier works well if the concept it has learned matches the new distribution

Change - A motivating example

Lets start with a problem - change in the incoming data can break our static model:

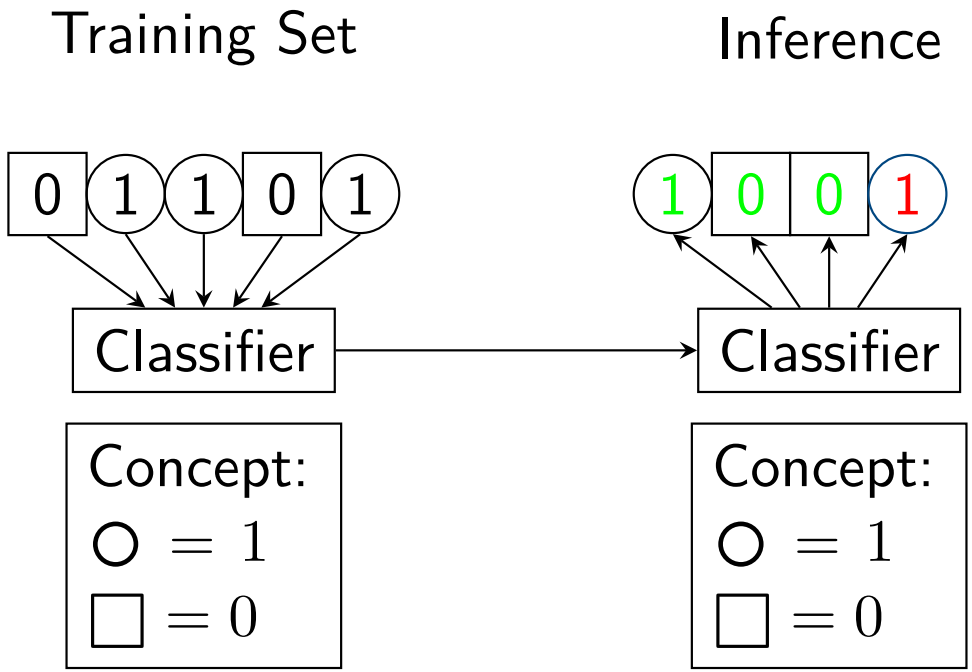


Figure: If the distribution of new observations changes (Shown as the shapes in blue), classification performance may drop

Change - A motivating example

Lets start with a problem - change in the incoming data can break our static model:

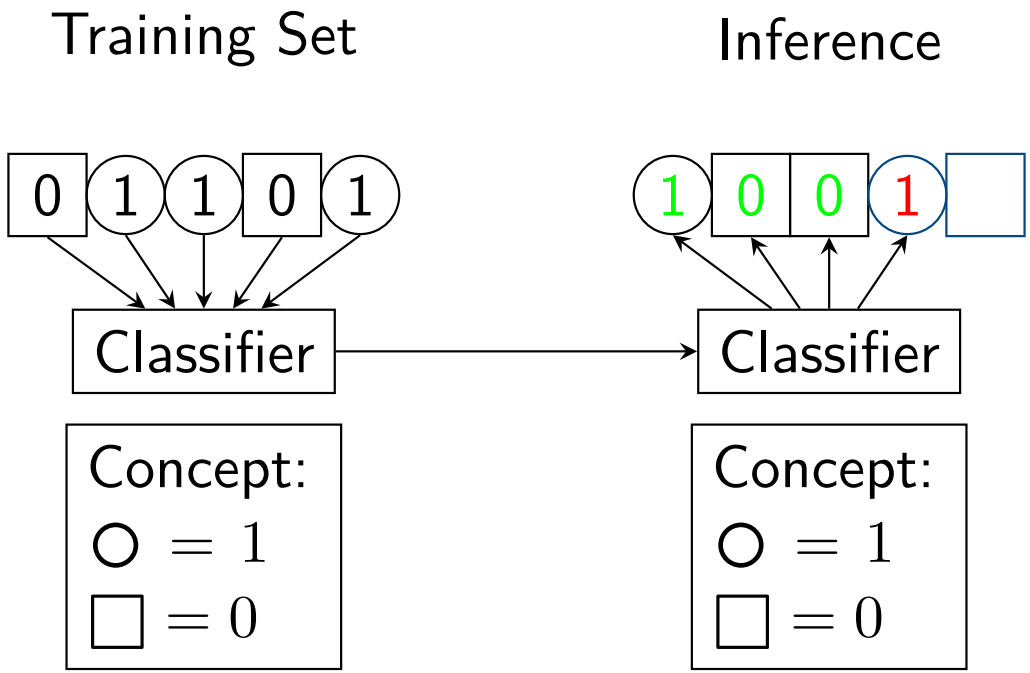


Figure: If the distribution of new observations changes (Shown as the shapes in blue), classification performance may drop

Change - A motivating example

Lets start with a problem - change in the incoming data can break our static model:

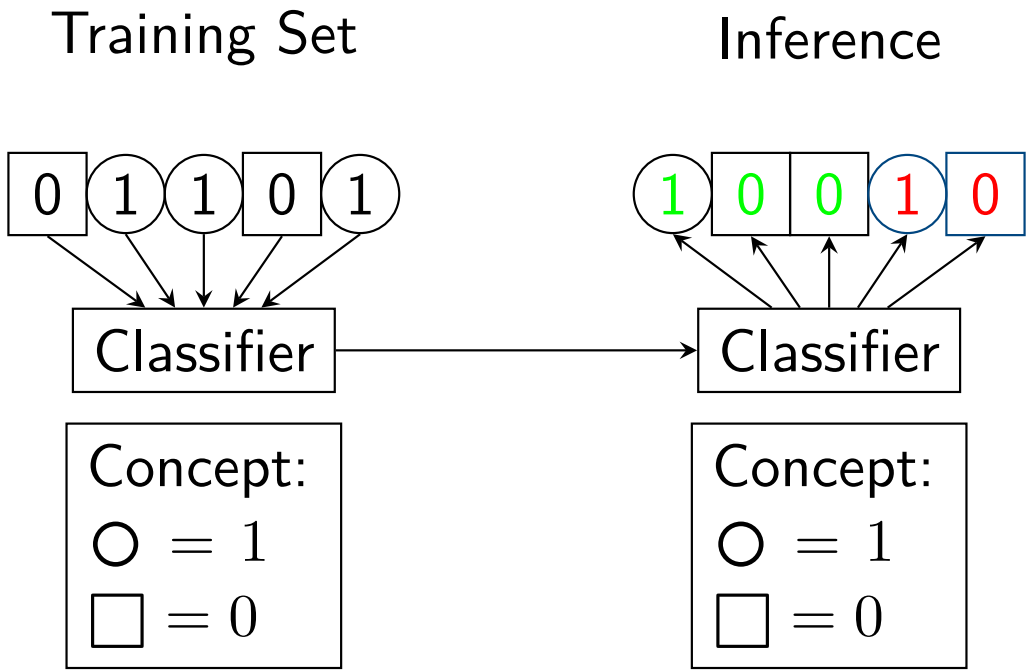


Figure: If the distribution of new observations changes (Shown as the shapes in blue), classification performance may drop

Solution - Data Stream Mining

- Data stream mining aims to properly consider *streams* of data.
- Stream mining techniques are able to handle problems which *static* machine learning methods are not able to.
- Mainly focused on situations with the following 4 properties:
  1. Observations arrive sequentially, over time
  2. Stream is too large to fit in memory, so we can only store a fixed number of observations at once (e.g., last 100). O(1) memory per observation
  3. Need to work in a limited amount of time, before the next observation is received. O(1) time per observation
  4. Can make a prediction at any time. No training downtime.

Example - Weather data

- Weather data is often monitored in real time, with observations up to every minute (e.g., MetService)
  1. Weather observations are sequential, and arrive one at a time
  2. A single computer could not handle a dataset of minute by minute weather observations, so must be processed sequentially
  3. A real time ML system would need to train on and predict each observation in less than a minute, before the next observation arrives
  4. Must be ready to predict at all times

Problems

- We will look at three important problems for streaming data:
- How can we get a representation of an infinite stream in a finite amount of memory? (Approximation)
- How can we handle data which changes over time? (Concept Drift)
- How can we make ML learners which can learn one observation at a time? (Incremental Learning)

Tutorial Question 6

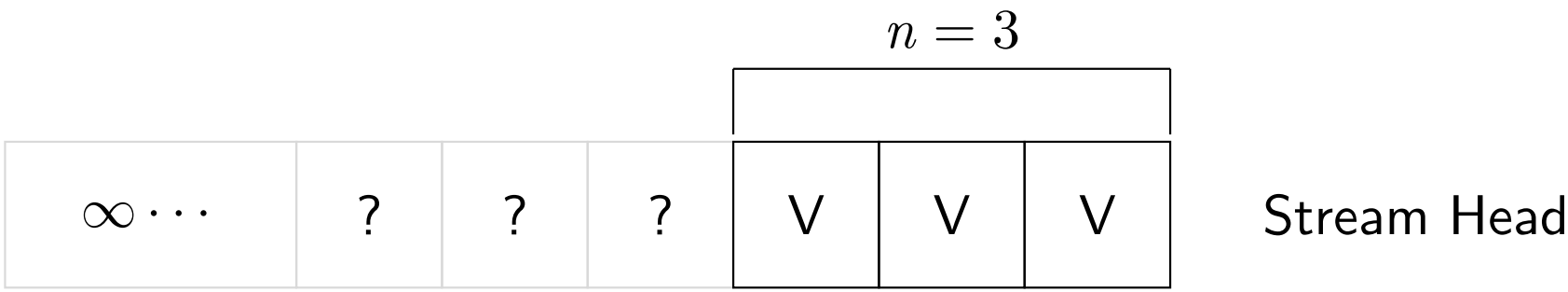
Give examples of three requirements for data stream mining that makes it different from regular data mining.

Infinite Data Vs Finite RAM

- A data stream may contain an infinite number of observations, but our computers only have a limited amount of memory - We can't possibly read in all available data at once
- How can we properly learn if we cannot look at all of the data?
- We have to *approximate* the stream
  1. Capture all information relevant to the learning task,
  2. In a finite amount of space able to be handled by the computer

Sliding Window

- A simple approximation strategy is to only remember *recent* data.
- We could capture the most recent n observations and forget previous data. This is called a *sliding window* of length n
- This gives perfect information on recent data, but zero information on old data.



Sampling

- Sampling attempts to build a representation of *all* previously seen data
- Ideally, at a given point in time each prior observation should have an equal chance to be in the sample.
- Reservoir sampling - Idea is that each new observation has a chance at replacing items in the sample. This chance *decreases* over time, so that a new element is only as likely to enter the sample as the first item has of still being in the sample.
- We have *approximate* information on *all* observations, no matter the age.
- Do we always *want* older data?

Tutorial Question 3

When would you use a sliding window technique as compared to a reservoir sampling technique.

Tutorial Question 3

When would you use a sliding window technique as compared to a reservoir sampling technique.

- What observations does reservoir sampling capture that a sliding window does not?
- When would these observations not be wanted?

Changing Data

- In a data stream, observations arrive over time
- Incoming observations may *change* over time
- E.G., The patterns found in weather sensor readings change over time, as seasons change
- This *change in distribution* is known as *Concept Drift*

Data Stream

$t \rightarrow$

Figure: Distribution changes are common in *Data Streams*, sequences of observations received over time.

Data Stream

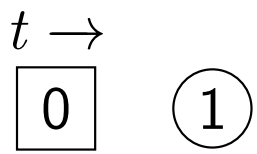


Figure: Distribution changes are common in *Data Streams*, sequences of observations received over time.

Data Stream

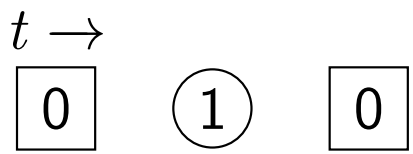


Figure: Distribution changes are common in *Data Streams*, sequences of observations received over time.

Data Stream

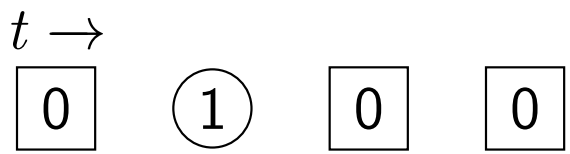


Figure: Distribution changes are common in *Data Streams*, sequences of observations received over time.

Data Stream

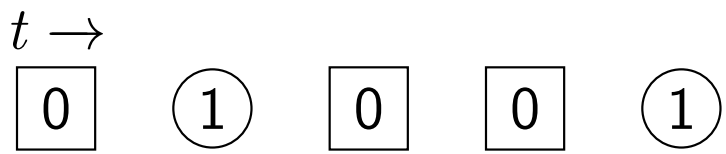


Figure: Distribution changes are common in *Data Streams*, sequences of observations received over time.

Data Stream

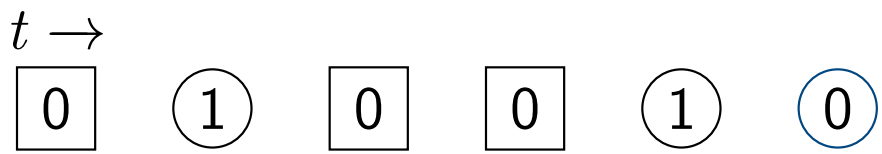


Figure: Distribution changes are common in *Data Streams*, sequences of observations received over time.

Data Stream

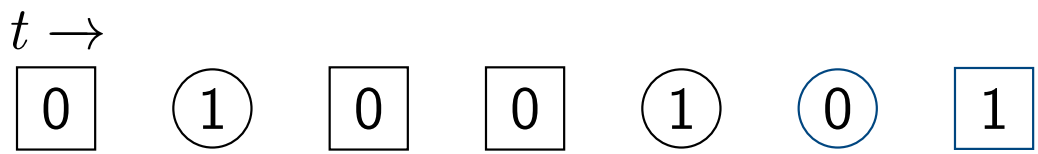


Figure: Distribution changes are common in *Data Streams*, sequences of observations received over time.

Data Stream

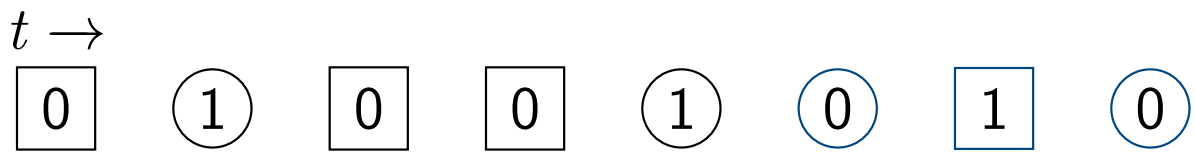


Figure: Distribution changes are common in *Data Streams*, sequences of observations received over time.

Data Stream

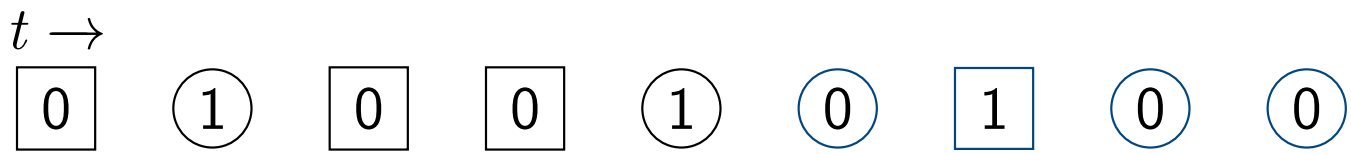


Figure: Distribution changes are common in *Data Streams*, sequences of observations received over time.

Data Stream

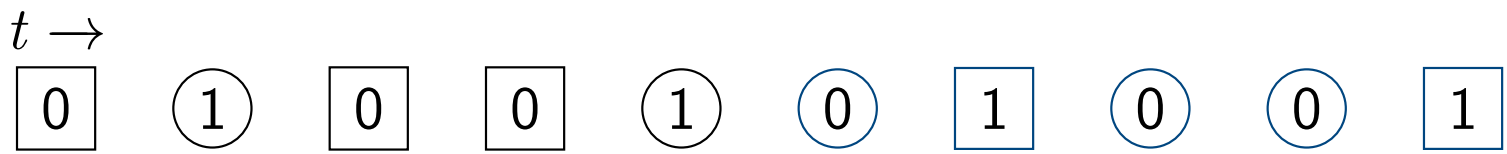


Figure: Distribution changes are common in *Data Streams*, sequences of observations received over time.

Data Stream

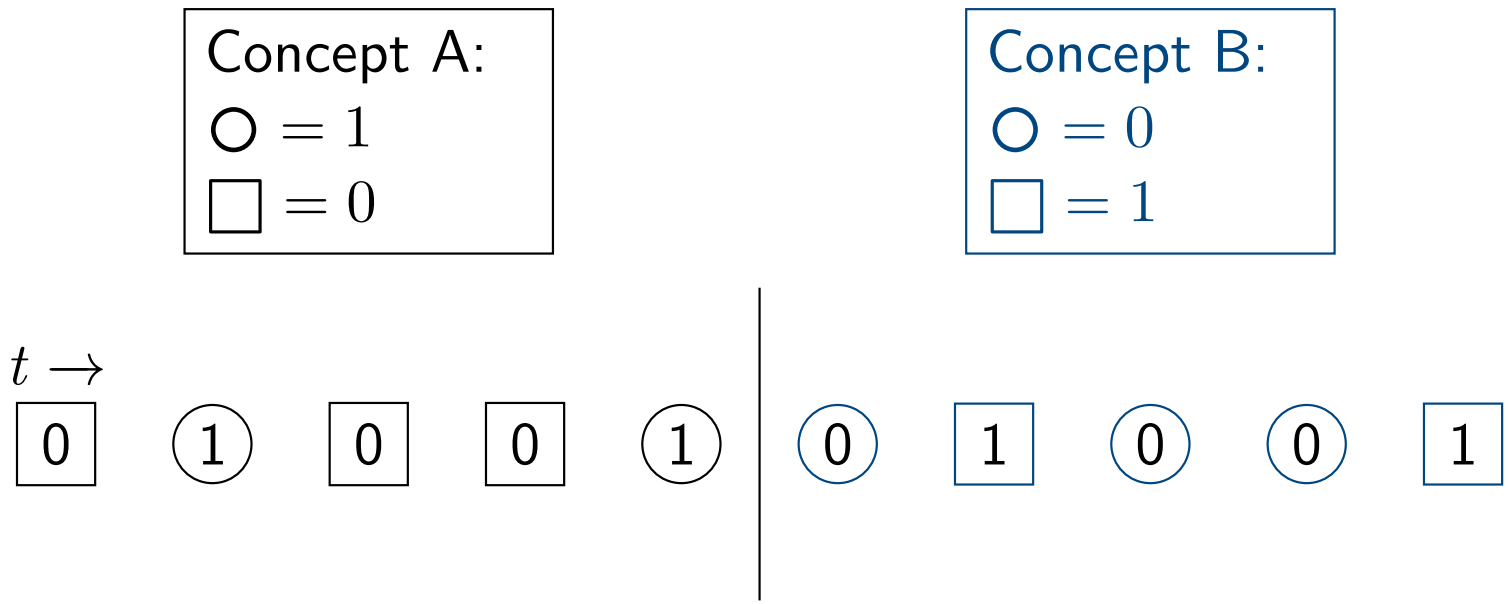


Figure: Concepts are related to some **hidden** underlying context in the stream. When the context changes, the distribution changes too. Color here is hidden, so the model cannot learn this change itself

Data Stream

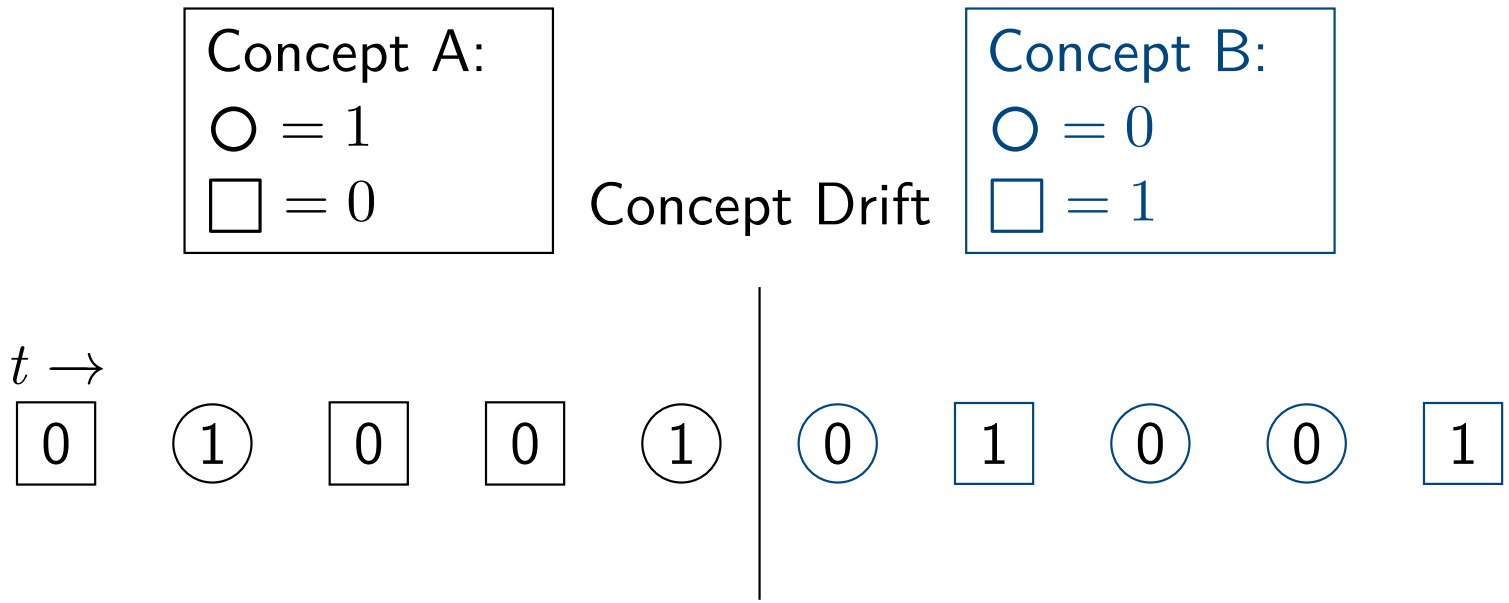


Figure: We assume that context is temporally stable, *i.e.*, we receive groups of observations drawn from the same concept

Data Stream

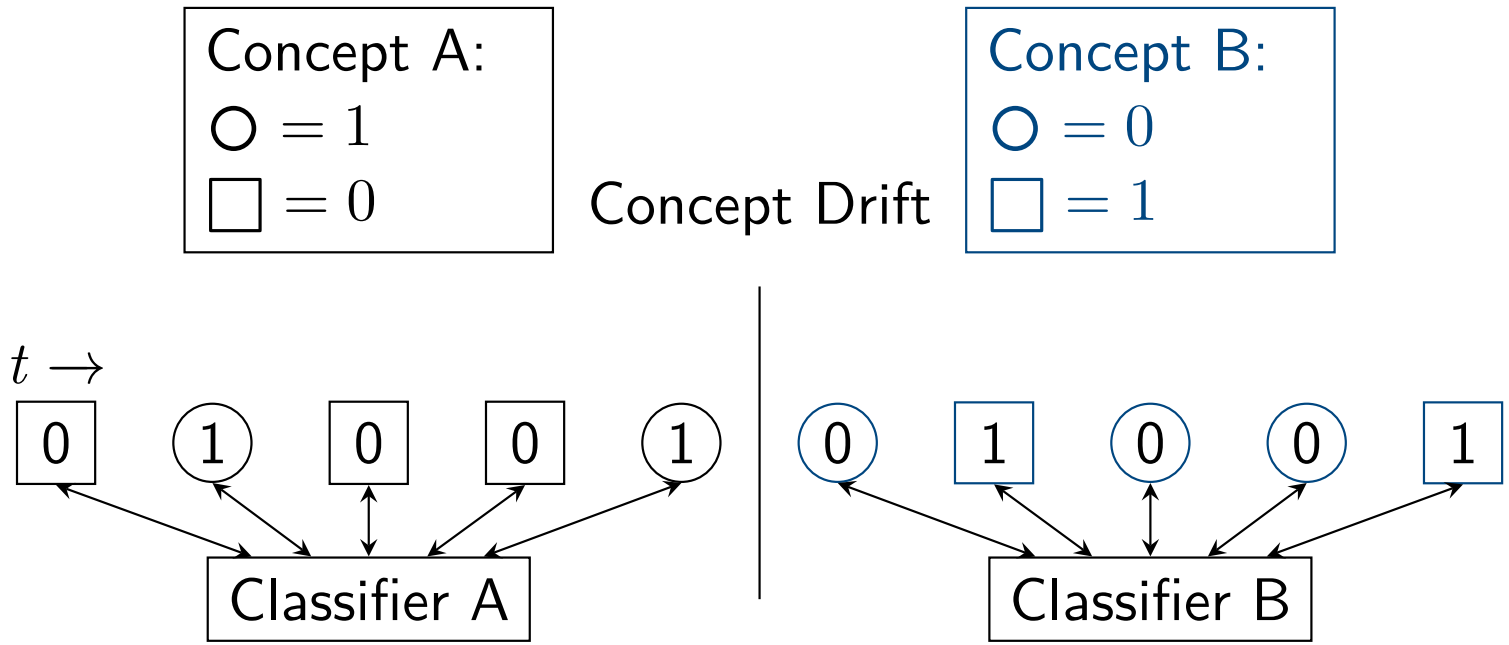


Figure: We want to *adapt* to concept drift

Concept Drift

- **Concept Drift** describes a change in the distribution of input data and labels over time
  - **Real** concept drift: The relationship between the input data and label has changed
  - **Virtual** concept drift: The distribution of input data has changed, but not the decision boundary
- It may cause prediction accuracy to deteriorate over time
  - The distribution of new data is no longer the same as the data the model was trained on
- We need methods which can account for concept drift when learning from a stream

Real Concept Drift

- Air Pollution Prediction: If we have readings from sensors around a city, we can predict what air pollution is like between sensors.
- But the relationship depends on other, unknown, factors, like wind direction.
- A change in wind direction would cause **Concept Drift**, we would need to learn to use a different set of sensors to make predictions.

Example

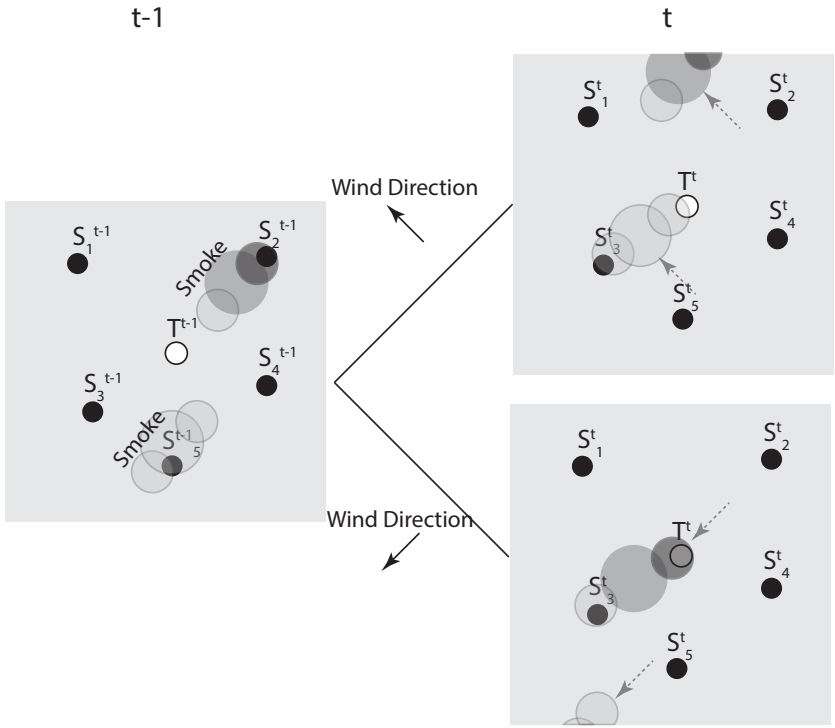


Figure: If wind is blowing from the south east, we should use the value of sensor S_5 to predict T. If from the south west, we should use sensor S_2.

Virtual Concept Drift

- If only the distribution of input features changes, it is known as *virtual* concept drift
- Virtual concept drift doesn't require different relationships to be learnt (the decision boundary stays the same)
- But the new data may be less well known by the model
- For example - A YouTube model may learn that the average age of recent viewers is inversely correlated with the views on Fortnite videos.
- In the school holidays, the distribution of viewer age may change (decrease), but the pattern still holds.
- This is a **virtual** concept drift, there was a change in the distribution of the input, but the relationship to the label is still valid.

Concept Drift Effect on performance

Real concept drift reduces performance, as the classifier has not learned the distribution of incoming data.

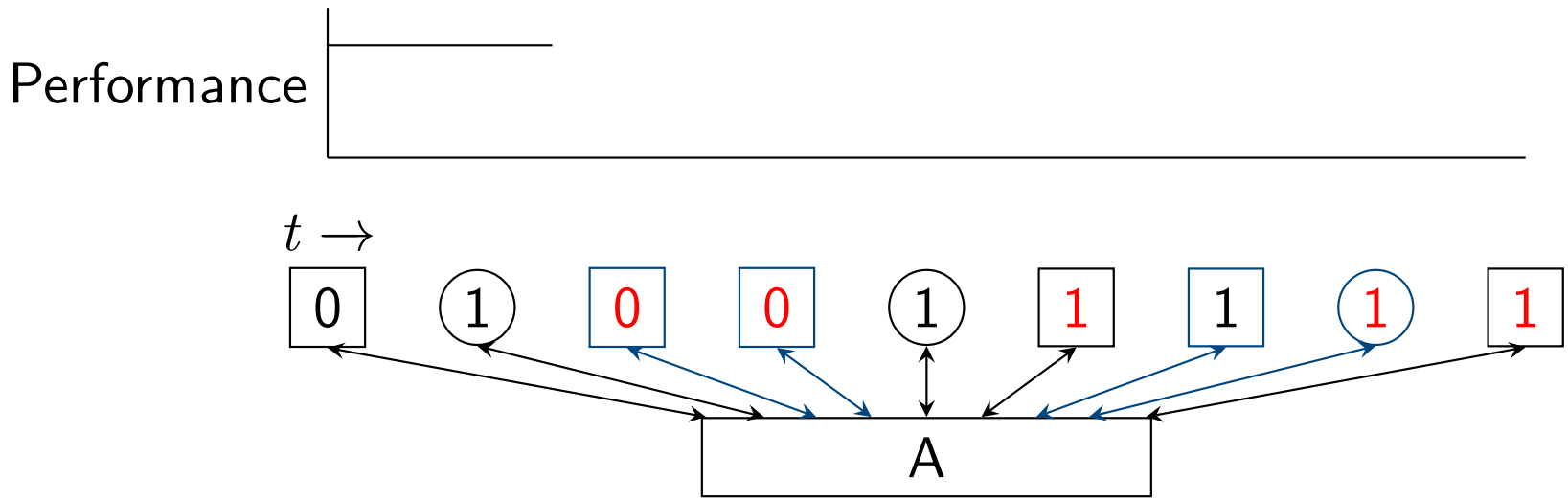


Figure: Performance without adaption

Concept Drift Effect on performance

Real concept drift reduces performance, as the classifier has not learned the distribution of incoming data.

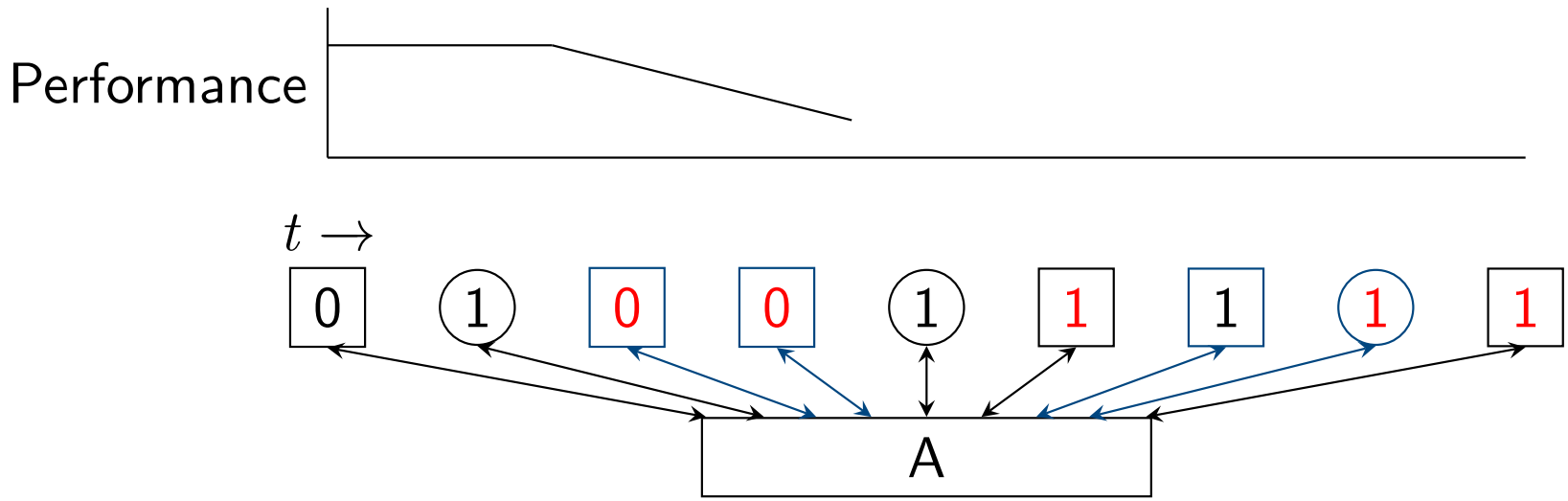


Figure: Performance without adaption

Concept Drift Effect on performance

Real concept drift reduces performance, as the classifier has not learned the distribution of incoming data.

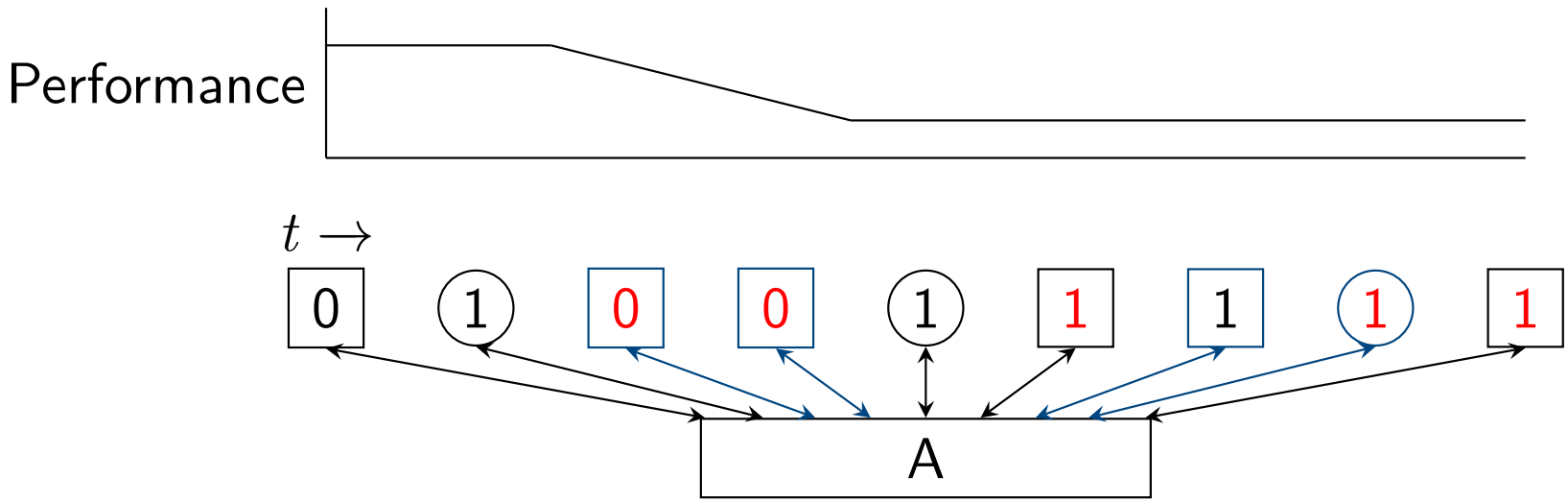


Figure: Performance without adaption

Concept Drift Detection

- Concept drift causes our current model to become out of date
- When concept drift occurs, we need to *adapt* our model to the new data, in order to retain performance
- This requires that we **Detect** concept drift
- We can do this with *Concept Drift Detectors*

Concept Drift Detection

- Intuition - A concept drift causes a drop in performance
- If we see a drop in performance, it could be concept drift or it could just be noise
- We can detect concept drift by looking for *significant* drops in performance

Concept Drift Effect on performance

Real concept drift reduces performance, as the classifier has not learned the distribution of incoming data.

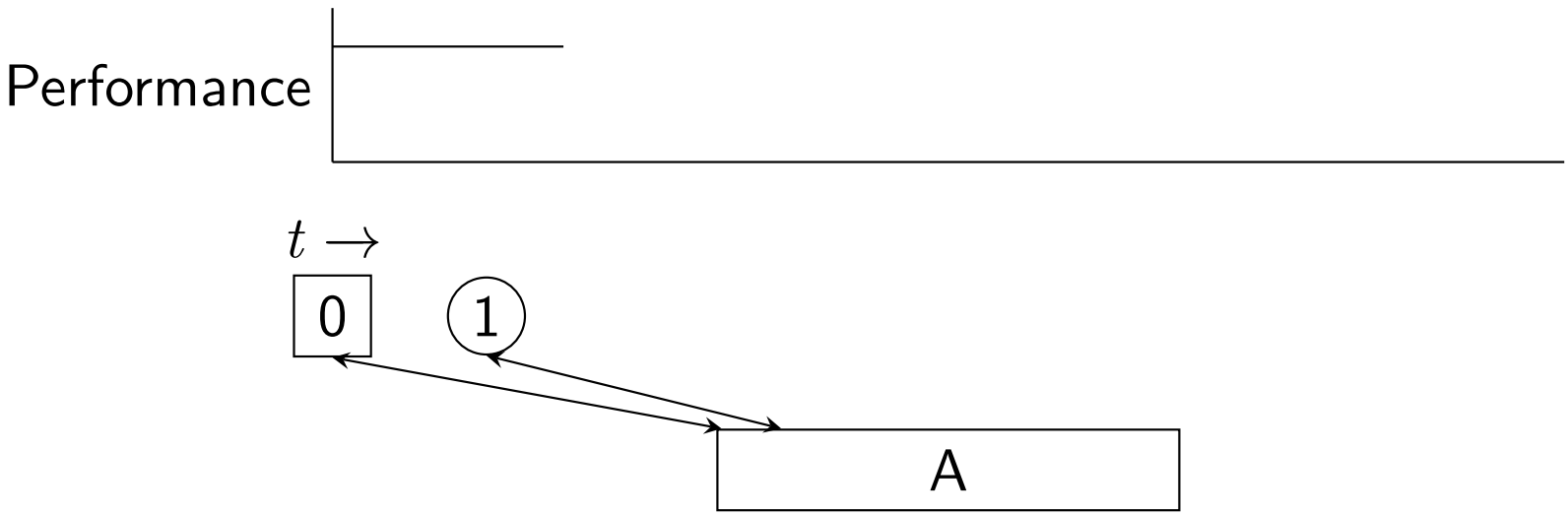


Figure: Good performance before drift

Concept Drift Effect on performance

Real concept drift reduces performance, as the classifier has not learned the distribution of incoming data.

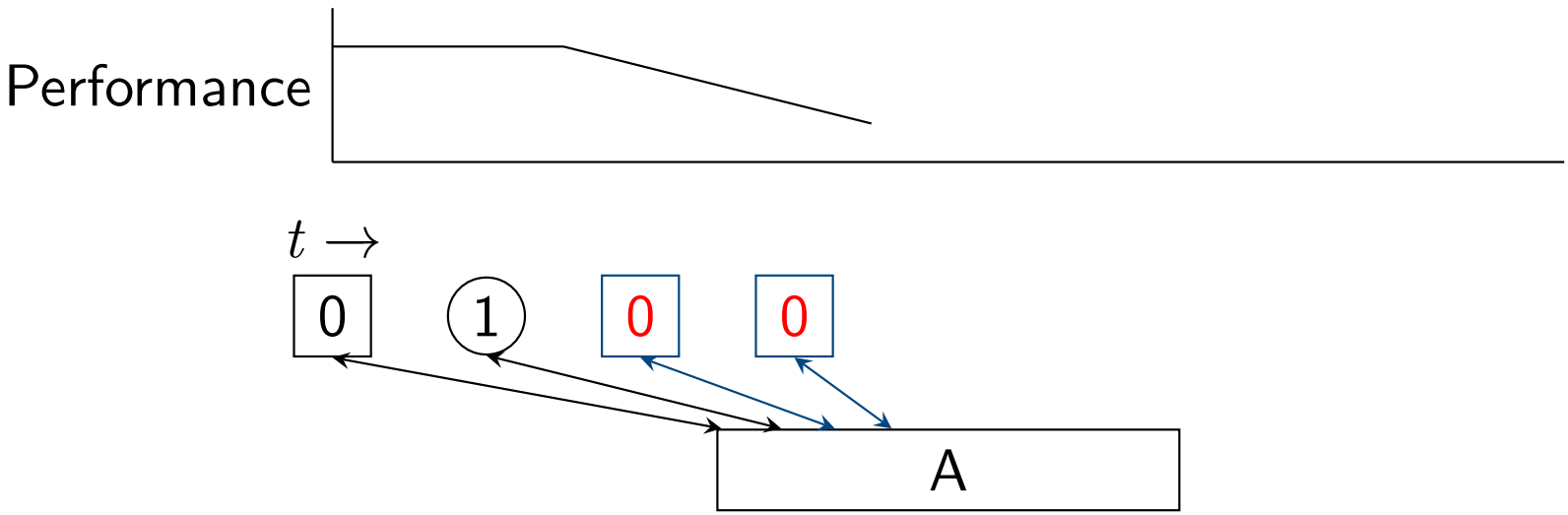


Figure: Performance drops due to drift

Concept Drift Effect on performance

Real concept drift reduces performance, as the classifier has not learned the distribution of incoming data.

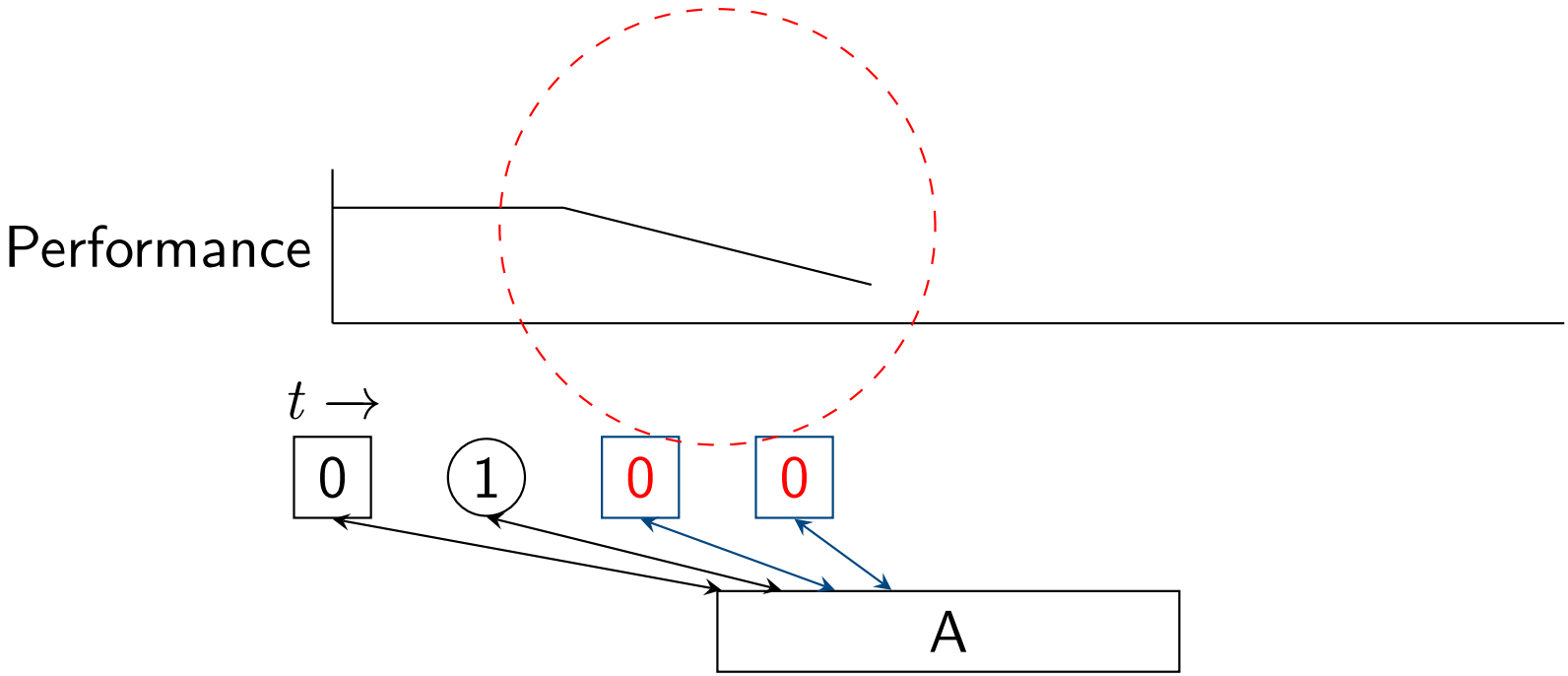


Figure: Performance Drop is detected as concept drift

Concept Drift Detectors

- CUSUM - Mean of input differs from 0
- DDM - Error rate more than threshold above minimum
- ADWIN - Difference in mean error rate between recent and older observations. Uses exponential histogram to store different windows to compare.

DDM - Question

DDM has the drawback that it may take a long time to react to changes after a long period without change. Suggest a couple of ways to fix this, possibly at the cost of introducing some parameters.

DDM - Question

DDM has the drawback that it may take a long time to react to changes after a long period without change. Suggest a couple of ways to fix this, possibly at the cost of introducing some parameters.

- How can we only consider recent data, rather than all data?

ADWIN - Question

Consider the implementation of ADWIN with the Hoeffding-based test. Discuss how ADWIN will react to: (1) A stream with an abrupt change. After a long stream of bits with average μ_0, at some time T the average suddenly changes to $\mu_1 \neq \mu_0$.
(2) Gradual after a long stream of bits with average μ_0, at time T the average starts increasing linearly so that the average at time $t > T$ is $\mu_{t+1} = \mu_t + \delta$, with δ a small value.

ADWIN - Question

Consider the implementation of ADWIN with the Hoeffding-based test. Discuss how ADWIN will react to: (1) A stream with an abrupt change. After a long stream of bits with average μ_0, at some time T the average suddenly changes to $\mu_1 \neq \mu_0$.
(2) Gradual after a long stream of bits with average μ_0, at time T the average starts increasing linearly so that the average at time $t > T$ is $\mu_{t+1} = \mu_t + \delta$, with δ a small value.

- How does ADWIN calculate change?
- ADWIN detects a change if there are two windows $W_0\, W_1$ such that if the mean of W_0 (δ_0) is further away from the mean of W_1 (δ_1) than a threshold 2ϵ
- ϵ is given by the Hoeffding bound.

Learning Algorithm Requirements

Learning algorithms which process streams must be able to:

- Update parameters by learning from one observation at a time
- Only store a fixed number of observations
- Be able to update before the next observation arrives (be fast)
-
- Does a standard decision tree meet these requirements?

Learning Algorithm Requirements

Learning algorithms which process streams must be able to:

- Update parameters by learning from one observation at a time
- Only store a fixed number of observations
- Be able to update before the next observation arrives (be fast)
-
- Does a standard decision tree meet these requirements?
- No, the standard DT algorithm needs to look at all observations to calculate split entropy.

Hoeffding Tree

- Intuition: We can store counts to approximate the dataset
- At each node, we count the number of observations of each class with each possible attribute value.
- We can calculate the quality of a split on a given attribute a_i with a function $G(a_i)$, based on these counts.
- If G is *good enough*, we make the split.
- How do we know if G is *good enough*?
- We use the *Hoeffding Bound*, if $G(a_i)$ of the best attribute is above the second attribute by more than the Hoeffding bound, we can say it is good enough to split on.

Hoeffding Tree

- (+) No need to store any of the previously observed training instances.
- (+) Although tree grows with the number of instances (increasing memory), it grows very slowly and there is a height.
- (+) Classification and training times are fast.
- (+) Able to account for relationships between attributes.
- (-) It does not account for concept drift.
- (-) Greedy algorithm. Training instances which occur earlier have more impact than those which occur later.

VFDT

- Very Fast Decision Tree
- An extension of Hoeffding Trees
- Implements some practical improvements:
  - Tie Breaking
  - Periodic Checks
  - Pruning attributes
  - Early stopping
  - Initialization

Concept drift aware learners

- Neither Hoeffding Tree or VFDT handle concept drift
- Concept-adapting very fast decision tree (CVFDT) algorithm as an extension of VFDT to deal with concept drift, maintaining a model that is consistent with the instances stored in a sliding window.
- Grows *alternative branches*, which swap in if they become more accurate
- Hoeffding Adaptive Tree (HAT) extends CVFDT, uses a drift detector to signal branch swapping. Better Guarantees, and fewer parameters.
- Both allow the model to overwrite splits learned from old data

Tutorial Questions 4

Most data stream techniques are approximation. Explain what this means in terms of Hoeffding Tree in comparison to static decision trees.

Tutorial Questions 4

Most data stream techniques are approximation. Explain what this means in terms of Hoeffding Tree in comparison to static decision trees.

- How does a Hoeffding Tree avoid the requirement of calculating a split criterion over the whole dataset?
- We approximate by storing *sufficient statistics*, in this case counts.

Ensemble Learners

- An ensemble is a collection of *diverse* models
- Ensembles fit naturally with concept drift, as we can train diverse models as the distribution changes. We can adapt to concept drift by incorporating new models into the ensemble.
- We may also need to *forget* old members of the ensemble, with some forgetting strategy.

Tutorial Question 7

What are the main differences in online ensemble techniques compared to static ensemble techniques? Choose an example of an technique and explain in detail what those differences are.

Tutorial Question 7

What are the main differences in online ensemble techniques compared to static ensemble techniques? Choose an example of an technique and explain in detail what those differences are.

- In static ensembles, we train a set of models with diverse results on a static distribution.
- In streaming ensembles, we train a set of models with diverse results in different distributions (and in static distributions).
- We mainly need to handle how to create *new* ensemble members to add diversity
- and how to remove *old* ensemble members which do not perform well any more.

DWM

- Maintains a set of models
- Weights votes based on current accuracy (to handle concept drift making some inaccurate)
- Drop models with consistent low performance
- Create new models when errors are made by the ensemble

Adaptive Random Forest

- Streaming random forest implementation
- For each tree, maintains a drift detector.
- If a warning is signalled, starts building a 'background' tree from incoming data
- If a drift is signalled, replaces the tree with the background tree
- This automatically handles both removing old models and building new models.

Evaluation Techniques

- Evaluation on a data stream has some differences to a static data set.
- Firstly, we have to maintain the sequential nature of the stream. Working with observations out of order does not give a realistic estimate of performance.
  - This means we cannot perform cross validation!
- Secondly, performance may be different at different points in the stream, due to concept drift.
  - Do we want to estimate performance before a drift? After a drift? Over the whole stream?

Holdout

- Like in static scenarios, a holdout evaluation retains a test set that is never used in training.
- In a streaming scenario, this set should come from the end of the stream, the data directly after the data the model was trained on.
- (-) What if there is a concept drift in the holdout set? We are not able to learn the change

Interleaved test then train

- In a streaming scenario we can test on each observation as it arrives, *then* train using it.
- This allows all observations to be used in testing and training
- Allows performance trends over time to be monitored
- *Prequential* evaluation is similar, but adjusts more recent data to be more important to the performance measure.

Tutorial Question 5

In prequential evaluation, you test then train a new incoming data point. Is there a reason for this? Prequential Evaluation is known as a pessimistic estimator. What does a pessimistic estimator mean?

Tutorial Question 5

In prequential evaluation, you test then train a new incoming data point. Is there a reason for this? Prequential Evaluation is known as a pessimistic estimator. What does a pessimistic estimator mean?

- See previous slide
- It is called a *pessimistic* estimator because it estimates a lower performance than would be obtained on future data. Why?

Tutorial Question 5

In prequential evaluation, you test then train a new incoming data point. Is there a reason for this? Prequential Evaluation is known as a pessimistic estimator. What does a pessimistic estimator mean?

- See previous slide
- It is called a *pessimistic* estimator because it estimates a lower performance than would be obtained on future data. Why?
- Because it includes old observations, which were tested using an *older version of the model*. The idea is that on future data, the model will have seen more observations so should reach a higher performance.

Tutorial Question

You are given the following list of 2D data points. $[[1,1],[1,2],[2,2],[2,1],[3,3],[2,5],[2,3]]$ If you had to select one point to be anomalous, which would you pick? Explain your answer. Please link your explanation to an anomaly detection technique.

Tutorial Question

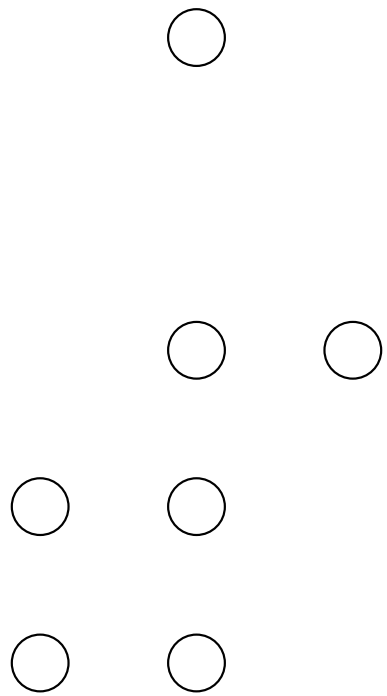


Figure: Data set plotted

Anomaly Detection Methods

- Parametric based approaches
  - Assume drawn from a normal distribution, estimate the maximum likelihood μ and σ.
  - anomaly if further than n standard deviations from the mean, usually $n = 3$
- Distance Based approaches
  - anomaly if further than a radius r from the k^{th} nearest neighbor
- Density Based Approaches
  - anomaly is density in local neighborhood is low relative to neighbors

Tutorial Question

Parametric

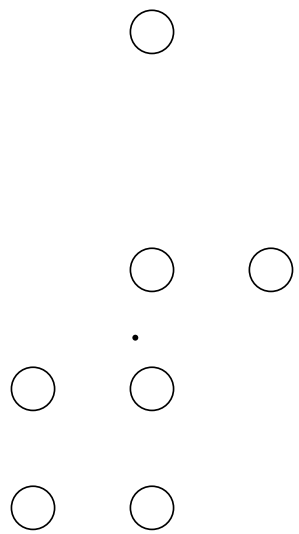


Figure: Estimate μ: Mean X = 1.86, Mean Y = 2.43

Tutorial Question

Parametric

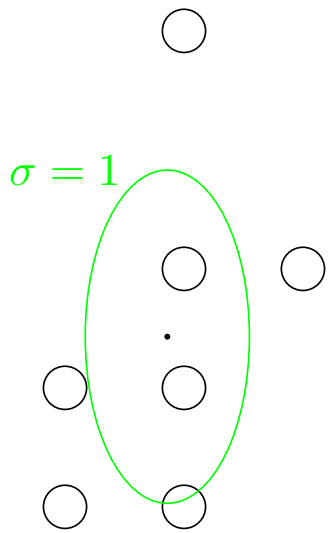


Figure: Estimate σ: X stdev $= 0.69$, Y Stdev $= 1.40$

Tutorial Question

Parametric

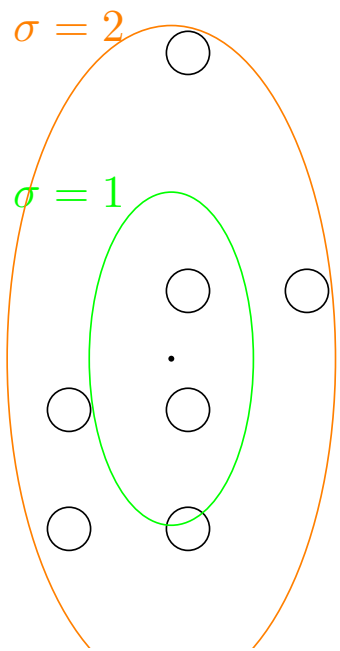


Figure: Check if nodes are within thresholds

Tutorial Question

Parametric

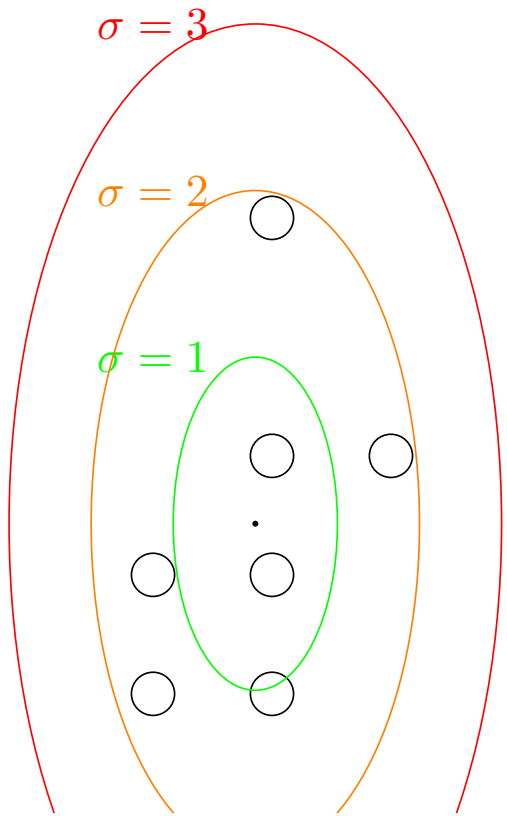


Figure: Check if nodes are within thresholds

Tutorial Question

Parametric - Normalized Standard deviation

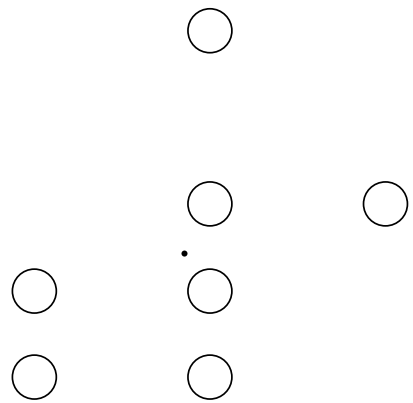


Figure: Re scaled based on σ (Mahalanobis Space)

Tutorial Question

Parametric - Normalized Standard deviation

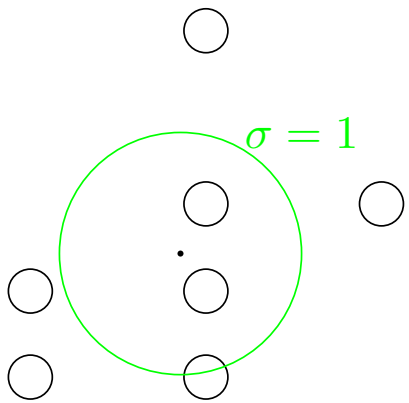


Figure: Notice circular Stdev curves due to normalization

Tutorial Question

Parametric - Normalized Standard deviation

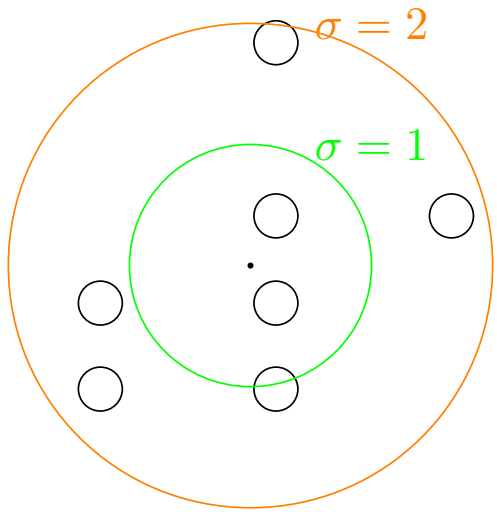


Figure: Notice circular Stdev curves due to normalization

Tutorial Question

Parametric - Normalized Standard deviation

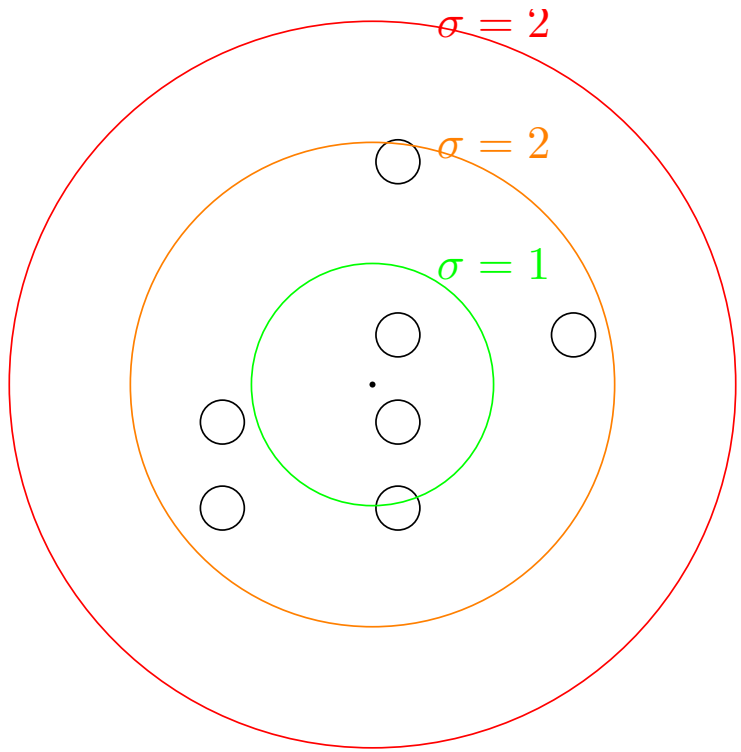


Figure: Notice circular Stdev curves due to normalization

Tutorial Question

Parametric - Normalized Standard deviation

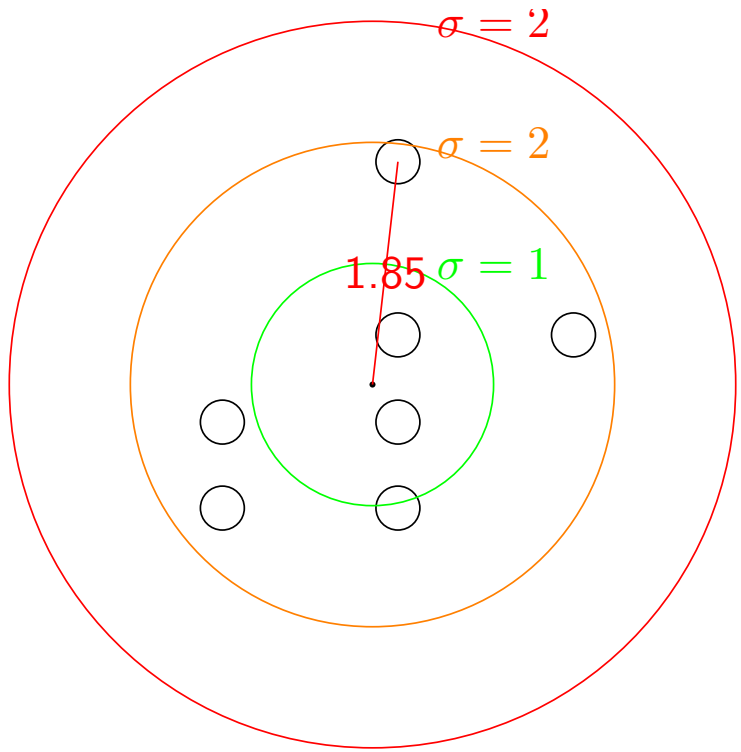


Figure: Calculate distance to μ, take largest as anomaly

Tutorial Question

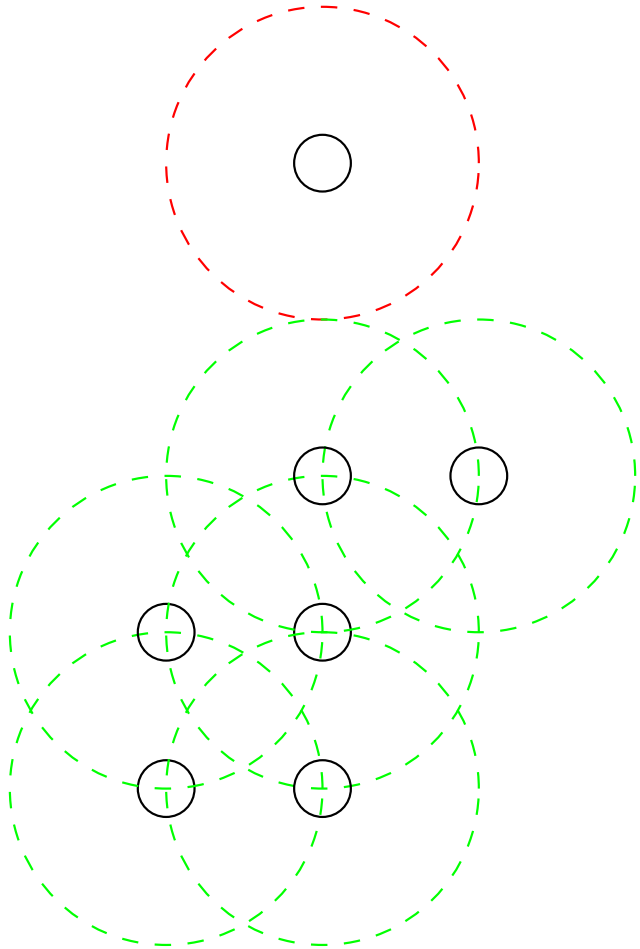


Figure: Distanced Based Detection, $r = 1$