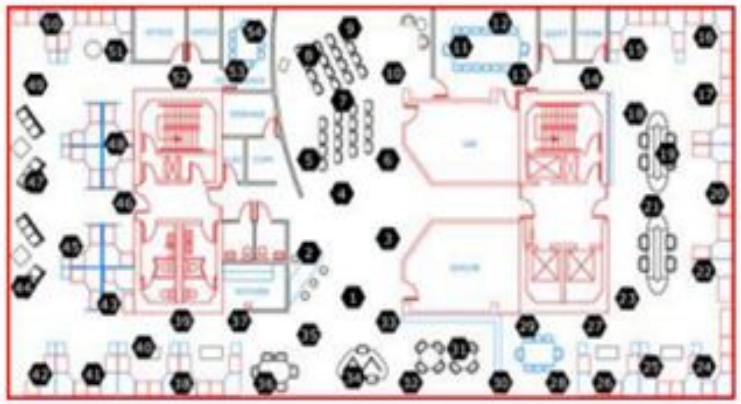
CPP: Concept Drift Detection via Class Posterior Probability

Motivation

Surveillance

Smart Phone



Sensors

Data Stream: (a)Infinite Length (b)Evolving Nature

Challenges:

- ◆Single Pass Handling
- ◆Low Time Complexity
- ◆Memory Limitation
- ◆Concept Drift

Introduction

Concept drift refers to the change of the conditional distribution of the target variable given the input data over time.

$$P(C_i|\mathbf{X}) = \frac{P(C_i)P(\mathbf{X}|C_i)}{P(\mathbf{X})}$$

Therefore, the question is how to well capture the change of conditional distribution of the target variable $P(\mathrm{C}\,|\,\mathrm{X})$

Introduction

Recent works can be summarized in two basic models

Distribution-based detector

- The distribution based methods detect concept drift by monitoring the change of data distributions between two fixed or adaptive windows of data.

- **Hard to determine window size**
- **Learn concept drift slower**
- **Focus on data distribution instead of model(classifier)**

Introduction

Recent works can be summarized in two basic models

Error-rate based detector

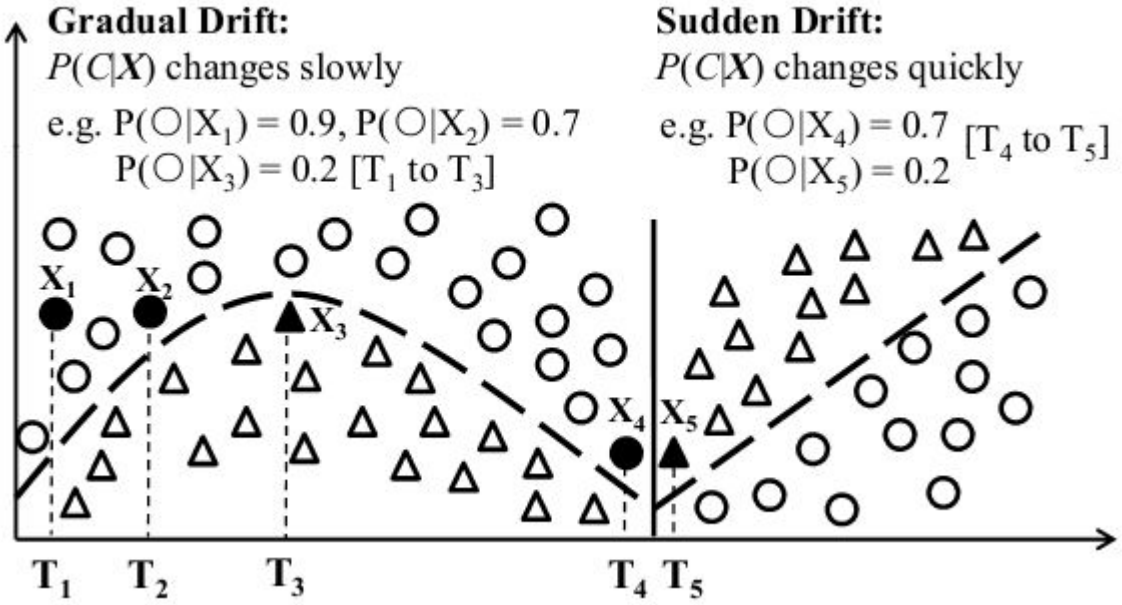
- Error-based methods often capture concept drift based on the change of the classification performance. (e.g. comparing the current classification performance to the average historical error rate with statistical analysis.)

- **Sensitive to noise**
- **Hard to deal with gradual concept drift**
- **Depend on learning model itself heavily**

Both methods don't model concept drift well

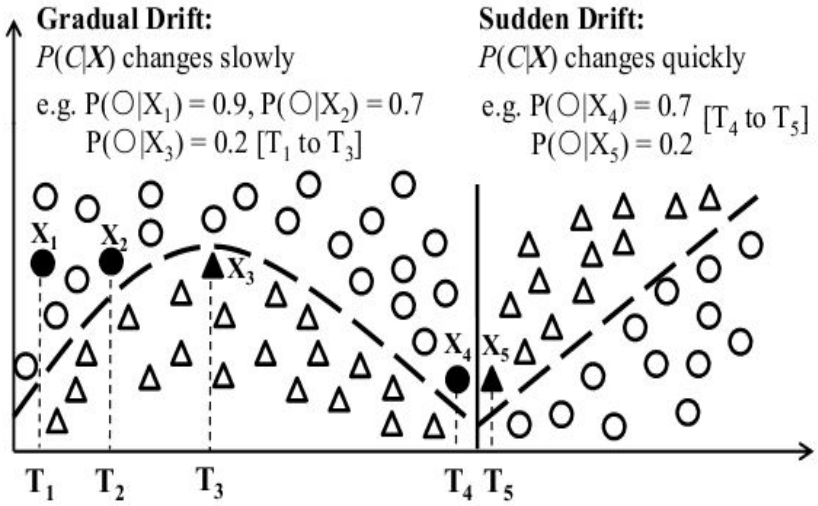
Intuition

Posterior probability: A gradual concept drift leads to a slow change of class posterior probabilities for one or more classes. The quicker the concept evolves, the faster the class posterior probability changes.



Building upon the class posterior probability, the concept drift is well modeled.

Basic idea



CPP: Instead of comparing distributions of two adaptive windows of data or using prediction performance to infer concept drift, we detect concept drift via Class Posterior Probability.

- Class Posterior Probability : Classifier Chain
- Gradual Concept Drift : act in a feedback way
- Sudden Concept Drift : potential drifting point distinguish and early-warning-late-confirmation

Classifier Chain

- **Here we use it to estimate online class posterior probability as it allows learning the dependencies among features effectively.**

Applying the product rule to $f(X_1, X_2, \ldots, X_n)$ yields

Label

$$f_1(X_1) \cdot f_2(X_2 \mid X_1) \cdot \ldots \cdot f_n(X_n \mid X_1, X_2, \ldots, X_{n-1})$$

Classifiers

Simply count it

Naïve Bayes

for $f_1(X_1)$

for $f_i(X_i \mid X_1, X_2, \ldots, X_{i-1})$, $i \in [2; n]$

Both allow us to estimate the density in an online fashion.

Gradual Concept Drift

How to dynamically model and calculate concept drift speed via class posterior probability?

Basic idea: The change rate of posterior probability for a given class C can reflect the corresponding concept drift speed. And we estimate the change rate of posterior probability in a instance-aware manner.

$$\omega_{C_i} = \frac{\mu^R_{C_i} - \mu_{C_i}}{\mu^R_{C_i}} \qquad \beta_{C_i}(j) = \beta_{C_i}(j-1) + \Delta\omega_{C_i}(j)$$

- ◆: $\mu^R_{C_i}$ mean of representative class posterior probability
- ◆: μ_{C_i} mean of all class posterior probability
- ◆: $\beta_{C_i}(j)$ hange rate of posterior probability

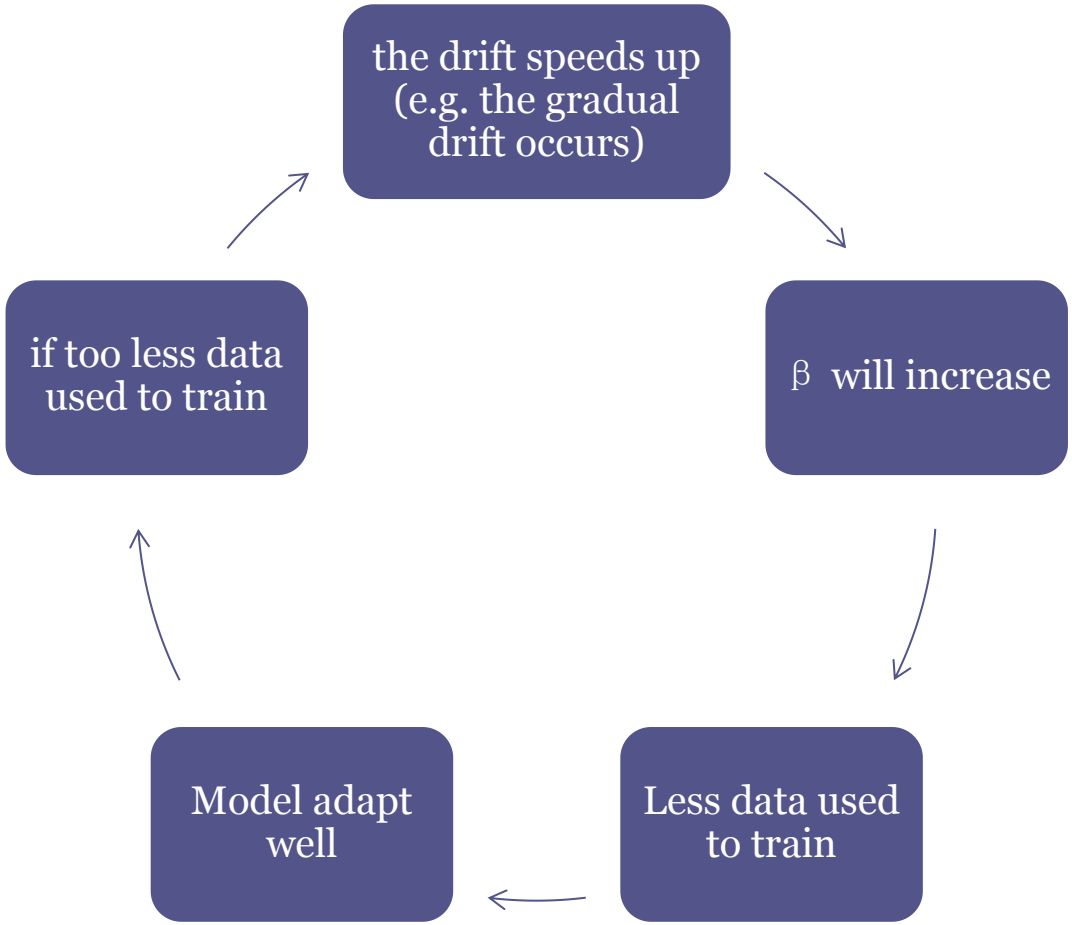
Gradual Concept Drift

How to adapt to concept drift better and faster when we obtain the concept drift speed?

Basic idea: The prediction model needs to be adaptively updated by learning recent relevant instances .Thus we use a forgetting-like mechanism such as decay function to dynamically update model by weighting.

$$w(j) = e^{\lambda\beta(j)} \cdot w(j-1)$$

Gradual Concept Drift



The system acts in a feedback way

Abrupt Concept Drift

If the drift is large enough (abrupt drift), we need to explicitly detect it for better adaptation though we have captured the gradual concept drift well.

The main challenge in abrupt concept drift detection:

Most existing algorithms face a common problem: the tradeoff between stability and sensitivity. An ideal concept drift detector should be sensitive to real concept drift, but robust to noise and virtual concept drift.

Potential drifting point distinguish

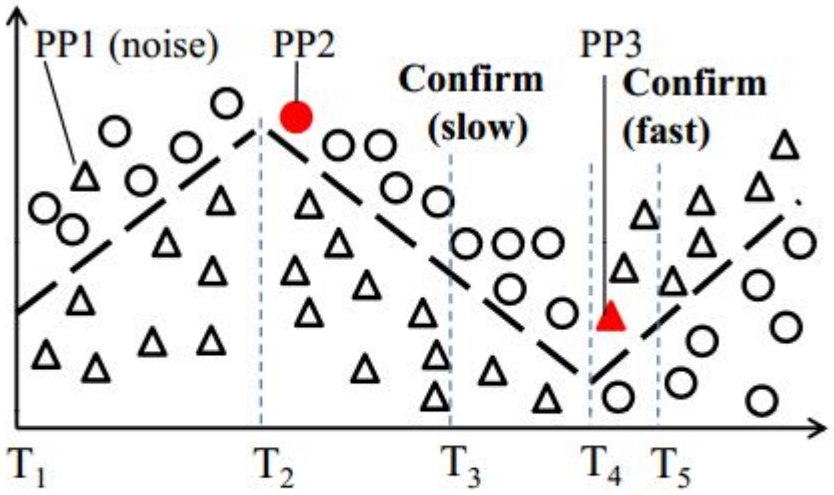


Figure 3: Illustration of abrupt concept drift detection and learning.

It is regarded as a potential drifting point if

$$P(X_j) < \mu_{C_i}^{R} - 2\sigma_{C_i}$$

where δ is the standard deviation of the class posterior probabilities.

Early-warning-late-confirmation

Once a potential drifting point is marked, we continuously monitor the following incoming data using a new tentative estimator.

	#regular inst.	#potential drifting inst.	Total
Current	a	b	$a+b$
Tentative	c	d	$c+d$
Total	$a+c$	$b+d$	N

If the estimates between two estimators are different significantly, the abrupt drift is confirmed, and the tentative estimator replaces the current estimator.

$$\chi^2_{Yates} = \frac{N(|ad-bc|-N/2)^2}{(a+b)(a+c)(b+d)(c+d)}$$

Otherwise, the potential drifting points are regarded as noise.

Experiments & Results

Experiment for abrupt concept drift

Data sets
- Synthetic data

Comparison methods
- ADWIN
- PL
- PHT
- STEPD
- DDM
- EDDM
- ECDD

Evaluation Metrics
- False Alarm Rate
- Miss Detection Rate
- Warning Delay
- Confirmation Delay
- Number Of Drift Detected

Table 2: The abrupt concept drift detection performance of different algorithms.

	FA	MD	WD	CD	DD
CPP	0.037	0.000	0.000	17.170	9.333
ADWIN	0.062	0.011	–	16.178	9.533
PL	0.931	0.000	–	8.670	130.800
PHT	0.988	0.985	255.281	309.622	10.567
STEPD	0.155	0.000	6.851	8.115	10.793
DDM	0.805	0.637	23.211	56.319	17.467
EDDM	0.930	0.507	28.552	85.148	64.267
ECDD	0.897	0.007	5.581	25.696	86.867

Experiment for abrupt concept drift

	Noise Level (1%)					Noise Level (5%)					Noise Level (10%)				
	FA	MD	WD	CD	DD	FA	MD	WD	CD	DD	FA	MD	WD	CD	DD
CPP	0.007	0.000	0.000	18.289	9.067	0.000	0.000	0.000	18.370	9.000	0.004	0.000	0.904	20.904	9.033
ADWIN	0.058	0.019	–	16.296	9.433	0.177	0.156	–	18.489	9.267	0.596	0.593	0.000	25.481	9.133
PL	0.935	0.000	–	9.033	139.933	0.953	0.007	–	9.437	191.533	0.974	0.007	0.000	10.889	350.433
PHT	0.997	0.996	263.337	317.737	10.567	0.997	0.996	290.878	346.693	11.333	0.996	0.996	309.407	368.704	11.300
STEPD	0.142	0.000	7.027	8.364	10.655	0.108	0.000	8.153	10.195	10.172	0.043	0.000	9.981	12.494	9.448
DDM	0.825	0.663	25.074	62.463	17.933	0.843	0.737	73.915	77.822	17.000	0.907	0.841	39.400	98.285	16.033
EDDM	0.929	0.478	31.320	92.110	68.533	0.944	0.585	52.289	154.230	68.000	0.951	0.704	95.667	278.593	58.433
ECDD	0.900	0.030	14.641	72.874	87.600	0.894	0.019	9.752	61.259	83.933	0.899	0.011	9.663	25.974	88.900

1、CPP's confirmation delay is (sub)optimal time to exclude the noise effect.
2、CPP allow identifying all the drifting concepts nearly without false alarm and miss detection.

Experiment for gradual concept drift

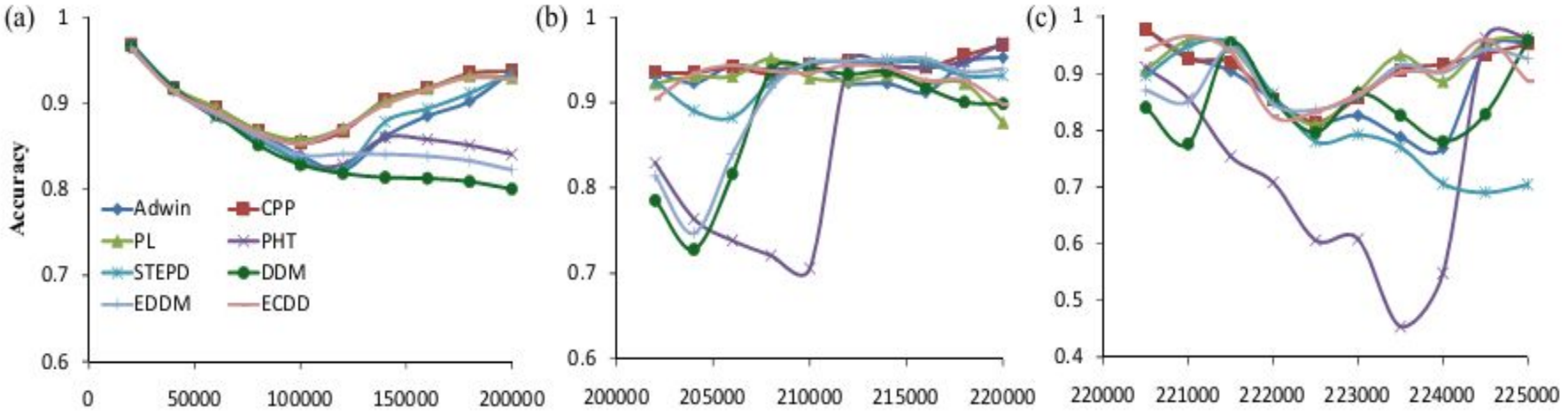


Figure 4: Performance of distinct algorithms on evolving synthetic data streams with different speeds of gradual concept drift. (a) Drift speed: 0.1% (slow), (b) Drift speed: 1% (medium), (c) Drift speed: 10% (fast).

1、CPP allows well capturing gradual concept drift and learning adaptively
2、CPP can capture subtle change of concept drift at an instance level
3、The performance of CPP gradually increases as the estimator is more stable and robust.

Experiment in real world datasets

Data	#Obj	#Dim	#Class	Methods	Acc.	Prec.	Rec.	F_1	Time (ms)
NEweather	18,159	8	2	CPP	**0.757**	0.716	**0.708**	**0.712**	747
				SingleClassifier (DDM)	0.707	0.675	0.694	0.684	643
				SingleClassifier (EDDM)	0.730	0.685	0.679	0.682	**250**
				NaïveBayes (PL)	0.731	0.686	0.677	0.681	1030
				HoeffdingAdaptiveTree	0.735	0.691	0.684	0.687	742
				OzaBagAdwin	0.750	**0.719**	0.657	0.686	1077
				WeightedEnsemble	0.703	0.672	0.691	0.681	2185
				PASC	0.705	0.664	0.672	0.668	1221
Shuttle	43,500	9	9	CPP	**0.993**	**0.656**	0.441	**0.527**	9731
				SingleClassifier (DDM)	0.925	0.332	**0.463**	0.387	938
				SingleClassifier (EDDM)	0.919	0.334	0.445	0.381	1182
				NaïveBayes (PL)	0.929	0.337	0.341	0.339	922
				HoeffdingAdaptiveTree	0.970	0.389	0.430	0.408	**731**
				OzaBagAdwin	0.976	0.410	0.407	0.408	4291
				WeightedEnsemble	0.970	0.470	0.338	0.393	11670
				PASC	0.932	0.321	0.335	0.328	3712
Electricity	45,312	8	2	CPP	**0.882**	**0.877**	**0.877**	**0.877**	2481
				SingleClassifier (DDM)	0.812	0.809	0.803	0.806	**949**
				SingleClassifier (EDDM)	0.848	0.845	0.845	0.381	1037
				NaïveBayes (PL)	0.871	0.868	0.869	0.869	2043
				HoeffdingAdaptiveTree	0.834	0.832	0.826	0.829	2264
				OzaBagAdwin	0.843	0.845	0.833	0.839	4305
				WeightedEnsemble	0.709	0.702	0.702	0.702	5914
				PASC	0.786	0.782	0.777	0.779	1981
US_Census	32,561	14	2	CPP	**0.855**	0.804	**0.788**	**0.796**	6173
				SingleClassifier (DDM)	0.834	0.787	0.728	0.756	**660**
				SingleClassifier (EDDM)	0.832	0.781	0.726	0.753	1227
				NaïveBayes (PL)	0.762	0.667	0.648	0.657	905
				HoeffdingAdaptiveTree	0.828	0.787	0.703	0.742	976
				OzaBagAdwin	0.845	**0.815**	0.728	0.769	2512
				WeightedEnsemble	0.829	0.775	0.728	0.751	6132
				PASC	0.775	0.687	0.666	0.676	2466

Conclusion

- ✓ Our method successfully model the concept drift via class posterior probability.
- ✓ Our method successfully detect the abrupt concept drift with noise via pinpointing the noisy examples.
- ✓ Our method successfully model the speed of concept drift and adapt to the gradual concept drift dynamically in a feedback way.

Thanks for your attention !

Q & A