Trend or No Trend: A Novel Nonparametric Method for Classifying Time Series

by

Stanislav Nikolov

S.B., Massachusetts Institute of Technology (2011)

Submitted to the Department of Electrical Engineering and Computer Science

in partial fulfillment of the requirements for the degree of

Master of Engineering in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2012

Author .....
Department of Electrical Engineering and Computer Science
August 15, 2012

Certified by .....
Prof. Devavrat Shah
Jamieson Career Development Associate Professor of Electrical Engineering and Computer Science
Thesis Supervisor
August 15, 2012

Certified by .....
Dr. Satanjeev Banerjee
Engineer, Twitter Inc.
Thesis Co-Supervisor
August 15, 2012

Accepted by .....
Prof. Dennis M. Freeman
Chairman, Masters of Engineering Thesis Committee

Trend or No Trend: A Novel Nonparametric Method for Classifying Time Series

by

Stanislav Nikolov

Submitted to the Department of Electrical Engineering and Computer Science
on August 15, 2012, in partial fulfillment of the
requirements for the degree of
Master of Engineering in Electrical Engineering and Computer Science

Abstract

We propose a nonparametric detection framework for outbreaks of viral topics in social networks.

Thesis Supervisor: Prof. Devavrat Shah
Title: Jamieson Career Development Associate Professor of Electrical Engineering and Computer Science

Thesis Supervisor: Dr. Satanjeev Banerjee
Title: Engineer, Twitter Inc.

Acknowledgments

This is the acknowledgements section. You should replace this with your own acknowledgements.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Get to the point of what this is about and what your contributions are in the first couple of sentences. Then you can elaborate.

Detection, classification, and prediction of events in temporal streams of information are ubiquitous problems in science, engineering and society (TOO BROAD?). From detecting malfunctions in a production plant, to predicting an imminent market crash, to revealing emerging popular topics in a social network, extracting useful information from time-varying data is fundamental for understanding the processes around us and making decisions.

(The method does not require that the indpendent variable is time.)

In recent years, there has been an explosion in the availability of data related to virtually every human endeavor — data that demands to be analyzed and turned into valuable insights. Massive streams of user generated documents, such as blogs and tweets, as well as data from portable electronic devices, provide an amazing opportunity to study the dynamics of human social interaction online and face to face (cite Pentland?). How do people make decisions? Who are they influenced by? How do ideas and behaviors spread and evolve? These are questions that have been impossible to study empirically at scale until recent times. In healthcare, records of over-the-counter medication sales (cite National Retail Data Monitor) as well as

search engine queries can anticipate the outbreak of disease and provide insight into the most effective ways to limit its spread. Particle collision experiments at the Large Hadron Collider generate more than 15 petabytes (SOURCE?) of data every year that promises to reveal the most fundamental physical truths.

(Does the physics example really fit in here? It's not a network example, but it looks like we won't be using anything network specific in the method.)

At the same time, advances in distributed computing have made it easier than ever to exploit the structure in that data to do inference at scale. ELABORATE?

All of the examples mentioned above share a common setting. There is an underlying process whose observable properties generate timeseries. We would like to observe those timeseries to

- detect anomalous events
- classify the current activity of the timeseries into event types
- predict the values of the timeseries at some future point.

This is difficult to do in general. Many real-world processes defy simple models that describe their behavior. A quote from "The Unreasonable Effectiveness of Data" by Halevy, Norvig, and Pereira sums it up:

> "*Eugene Wigner's article 'The Unreasonable Effectiveness of Mathematics in the Natural Sciences' examines why so much of physics can be neatly explained with simple mathematical formulas such as* $f = ma$ *or* $e = mc^2$*. Meanwhile, sciences that involve human beings rather than elementary particles have proven more resistant to elegant mathematics. Economists suffer from physics envy over their inability to neatly model human behavior. An informal, incomplete grammar of the English language runs over 1,700 pages. Perhaps when it comes to natural language processing and related fields, we're doomed to complex theories that will never have the elegance of physics equations. But if that's so, we should stop acting*

as if our goal is to author extremely elegant theories, and instead embrace complexity and make use of the best ally we have: the unreasonable effectiveness of data."

Like language, the behavior of complex systems rarely admits a simple model that works well in practice. Like machine translation and speech recognition, there is an ever growing amount of data "in the wild" about processes like epidemics, rumor-spreading in social networks, financial transactions, and more. The inadequacy of simple models for complex behavior requires an approach that embraces this wealth of data and it highlights the need for a unified framework that efficiently exploits the structure in that data to do detection, classification, and prediction in timeseries.

1.2 Previous Work

Maybe start from more specific things (Twitter event detection, etc) and move to more general/abstract things and conclude with nonparametric timeseries methods.

Event detection in timeseries.

Classification in timeseries.

Prediction in timeseries.

Models with theoretical justification for the underlying model of the process. Network cascade models. Branching process models. Epidemics, Social Network rumor spreading

(Nonparametric) timeseries clustering methods

Explicit generative model? Mixture model?

In principle, no tunable threshold required, because of explicit probabilistic model, threshold is just 1.

Experimental results note. It could just be that the false positive rate is all of the things that have high volume that we weren't able to normalize away, except that there's only a few of them, so maybe what we're doing is not that impressive. However, if we compare to the unnormalized ones, we do (get solid confirmation)

much better. Question: What false positive rate should we expect, even with the best method (just by looking at the data)?

Show experimenta; results for multiple data sets to make sure you didnt jyst tweak the signal normalization for a partucular dataset

1.3 My Approach

Simple models prove ineffective when exposed to the vast variety of real world situations. [SAY MORE ABOUT THIS] I propose a nonparametric framework for doing inference on timeseries (VAGUE). In this model, we posit that there exist of a set of latent timeseries, each corresponding to a prototypical event of a certain type, and that each observed timeseries is a noisy observation of one of the latent timeseries.

1.3.1 Anomaly Detection

A timeseires in a given window is compared to a set of reference timeseries that represents "normal" events. We compute the probability that the observed timeseries was generated by a latent timeseries corresponding to a timeseries in the reference set. We then declare observations with low probability to be anomalies.

1.3.2 Classification

A timeseries in a given window is compared to two *reference* sets of timeseries — one consisting of positive examples and the other of negative examples. We want to find out whether it is more likely that the observed timeseries was generated by one of the latent timeseries in the positive reference set or one of the latent timeseries in the negative reference set.

1.3.3 Prediction

A timeseries in a given window is compared to a single reference set of timeseries. We want to find out the probability that the observed timeseries was generated from the

same latent timeseries as each of the timeseries in the set. We then compute the most likely change in the observed timeseries by observing the changes in each reference timeseries and weighting the change by the probability that the observed timeseries and the reference timeseries were generated by the same latent timeseries.

**********Lots of work in clustering, but this has an explicit probabilistic model.

1.3.4 Application: Detecting Outbreaks of Popular Topics on Twitter

As an application, I apply the method (CALL IT SOMETHING) to the problem of detecting emerging popular topics (called *trends*) on Twitter. I use a set of labeled examples of timeseries correspondingn to topics that eventually became trending topics and topics that did not. I then perform online classification of a given topic to label it as trending or not trending at a particular time. **********Talk about results and main contributions.

Chapter 2

Detection Method

I propose a signal classification framework in which we assume that a signal of a particular category was generated by one of a set of latent signals representing that category.

Suppose that whenever a particular event happens it generates a signal measuring a particular observable property of the event. If the same type of event were to happen many times, we can suppose that the signals generated are noisy versions of a *latent* signal corresponding to that type of event.

Suppose that Π is the set of all possible events, Π_+ the set of all possible viral events and Π_- the set of all possible non-viral events. Suppose that all events are disjoint, in the sense that no two events can happen at once. Then if a viral event happens, it must be a noisy version of exactly one of the events in Π_+.

We observe a signal and would like to determine if this signal was generated by a topic that will become viral. To do this, we look at the signal for each positive event and ask if the test signal was generated by the same type of event that generated the positive event. We will assume that each signal was generated by a seperate event in Π.

Let us define the following variables:

- s_i — an reference signal.
- t_j — a latent signal.

- s — the observed signal that we would like to classify.
- V — the event that s was generated from a viral event.
- $\mathcal{R}_+$ — the indices of the set of viral reference signals.
- $\mathcal{R}_-$ — the indices of the set of nonviral reference signals.
- $\mathcal{L}_+$ — the indices of the set of latent signals corresponding to a viral reference signal
- $\mathcal{L}_-$ — the indices of the set of latent signals corresponding to a nonviral reference signal.

The probability that the obseved signal is viral is therefore

$$\begin{aligned}
\mathbb{P}(V|s) &= \sum_{i\in\mathcal{R}_+} \mathbb{P}(V, s \text{ shares a latent event with } s_i|s) \\
&= \sum_{i\in\mathcal{R}_+}\sum_{j\in\Pi_+} \mathbb{P}(s \text{ generated by } t_j, s_i \text{ generated by } t_j) \\
&= \sum_{i\in\mathcal{R}_+}\sum_{j\in\Pi_+} \exp\left[-\gamma d(s,t_j)\right]\exp\left[-\gamma d(s_i,t_j)\right] \\
&= \sum_{i\in\mathcal{R}_+}\sum_{j\in\Pi_+} \exp\left[-\gamma\left(d(s,t_j)+d(s_i,t_j)\right)\right] \qquad (2.1)
\end{aligned}$$

where $d(s,t)$ is the distance between signals s and t. For large γ, the term with the smallest exponent will dominate the sum over Π_+ and we can write

$$\mathbb{P}(V|s) \approx \sum_{i\in\mathcal{R}_+} \exp\left[-\gamma \min_{j\in\Pi_+}\left(d(s,t_j)+d(s_i,t_j)\right)\right] \qquad (2.2)$$

We now argue that for reasonable distance metrics, we can write

$$\min_{j\in\Pi_+}\left(d(s,t_j)+d(s_i,t_j)\right) = Cd(s,s_i)$$

for some C.

Chapter 3

Algorithm

In this chapter, I describe the practical implementation of the method described in Chapter 2.

3.1 Overview

The goal of the algorithm is to perform online classification of an infinite stream of samples from an observed digital signal. We will focus on the case of binary classification, in which we have positive signals and negative signals, but the results can be extended to multiple classes. For binary classification, one could imagine that one class represents events and the other non-events, and that we would like to detect events as soon as they happen.

To predict which class the observed signal belongs to at a given point in time, we compute the probability that the recent samples of the observed signal were generated by a "positive" process and the probability that they were generate by a "negative" process, based on previously observed positive and negative *reference signals*. Recall from Chapter 2 that a signal is generated by a particular type of process if it shares the same latent process as a signal of that type. To compute the probability that the recently observed samples share the same process with a particular reference signal, we compute the distance between the trajectory consisting of the recently observed samples and all trajectories of the same size in the reference signal, and take the

minimum over all such trajectories.

3.2 Implementation

Let S be the infinite observed signal and S_{cmpr} be the most recent CmprWindow samples. Let S' be a reference signal of length DetectionWindowHrs * SamplesPerHour

Algorithm 1 *Perform online binary classification on an observed signal S using sets of positive and negative reference signals R_+ and R_-.*

Detect(S, R_+, R_-, Gamma, Threshold, ReqConsecDetections, CmprWindow)

```
Detected ← False
repeat
  S_cmpr ← RecentSamples(S, CmprWindow)
  PosDists ← [ ]
  NegDists ← [ ]
  for S_ in R_ do
    PosDists.Append(DistToSignal(S_cmpr, S_+, CmprWindow))
  end for
  for S_ in R_ do
    NegDists.Append(DistToSignal(S_cmpr, S_−, CmprWindow))
  end for
  ProbRatio = ProbClass(PosDists) / ProbClass(NegDists)
  if ProbRatio > Threshold then
    Detected ← True
  end if
until Detected
```

Algorithm 2 *Compute the minimum distance between S_{cmpr} and pieces of S' of the same size.*

DistToSignal(S_{cmpr}, S')

```
CmprWindow ← length(S_cmpr)
MinDist = ∞
for i = 0 to length(S') - CmprWindow do
   S'_{cmpr,i} ← S'[i : i + CmprWindow]
   MinDist = Min(MinDist, Dist(S'_{cmpr,i}, S_cmpr))
end for
return MinDist
```

Algorithm 3 *Compute the distance between two signals S and S' of the same length*

Dist(S, S')

Chapter 4

Application: Identifying Trending Topics on Twitter

In this chapter, I consider the application of the method and algorithm proposed in chapters 2 and 3 toward detection of trending topics on Twitter. I discuss the Twitter service, the collectiion and pre-processing of data, and the experimental setup for the detection task.

4.1 Problem Statement

4.1.1 Overview of Twitter

Twitter is a real-time messaging service and information network. Users of Twitter can post short (up to 140 characters) messages called *tweets*, which are then broadcast to the users' *followers*. Users can also engage in conversation with one another. By default, tweets are public, which means that anyone can see them and potentially join a conversation on a variety of topics being discussed. Inevitably, some topics gain relatively sudden popularity on Twitter. For example, a popular topic might reflect an external event such as a breaking news story or an internally generated meme. Twitter surfaces such topics in the service as a list of Trending Topics.

4.1.2 Problem Statement

I apply the method and algorithm described in chapters 2 and 3 to the problem of detecting trending topics.

4.2 Data

4.2.1 Data Collection

The online time series classification task requires a set of labeled examples of trends (positive examples) and non-trends (negative examples). Each example consists of a topic and a signal representing a measurement about the topic over time. If the example is a positive one, it also contains the time of the *true onset* of the trend.

The data collection process can be summarized as follows. First, I collected 500 examples of trending topics and 500 examples of non-trending topics between June 1, 2012 and June 30, 2012 (hereafter refered to as the *time window*). Then, I labeled a sample of tweets with the topic or topics contained therein. Finally, I constructed a signal for each topic based on the tweet activity corresponding to that topic.

I obtained all data directly from Twitter as an employee of Twitter via the MIT VI-A thesis program. However, the type as well as the amount of data I have used is all publicly available via the Twitter API.

Topics

I collected a list of all trending topics on Twitter from June 1, 2012 to June 30, 2012, as well as the times that they were trending and their rank in the trending topics list on Twitter. Of those, I filtered out topics whose rank was never better than or equal to 3 between June 1, 2012 and June 30, 2012. In addition, I filtered out topics that did not trend for long enough (time of first appearance to time of last appearance is less than 30 minutes) as well as topics that reappeared multiple times in the time window (time of first appearance to time of last appearance is greater than 24 hours). The former eliminates many trends that are spurious and trend for a very short time.

The latter eliminates topics that correspond to multiple events, such as the name of a football player, whose name might trend every time there is an important game.

I collected non-trending topics in two steps. First, I sampled a list of n-grams (phrases consisting of n "words") in the time window for n up to 5. I filtered out n-grams that contain a trending topic, using the original, unfiltered list of all trending topics in the time window. For example, if "Whitney Houston" is a trending topic in the time window, then "Whitney" would not be accepted as a non-trending topic. I also removed n-grams shorter than three characters, as most of these were not meaningful topics. Finally, I sampled 500 n-grams uniformly from the filtered list of n-grams.

Tweets

I sampled 10% of all public tweets from June 1, 2012 to June 30, 2012 inclusive. I labeled each tweet with the topic or topics contained therein using a simple regular expression match between the tweet text and the topic text.

4.2.2 From Tweets to Signals

I discuss the process of converting the timestamped tweets for a given topic into a reference signal. Each of the steps below is followed in order for each topic.

Rates

First, we determine the rate of tweets about a topic over time by binning the tweets into time bins of a certain length. I used time bins of length two minutes.

Normalization to Remove Baseline

A first glance at the data (TODO: Show this!) reveals that many non-trending topics have a relatively high rate and volume of tweets, and many trending topics have a relatively low rate and volume of tweets.

On important difference is that many non-trending topics have a high baseline rate while most trending topics are preceded by little, if any, activity before they start gaining popularity. To remove whatever baseline rate may be present, we divide the rate by the mean rate over the entire window. We then take the quotient to a power β, which controls how much we reward and penalize rates above and below the mean rate.

Derivative to Reward Spikes

Another difference between the rates of tweets for trending topics and that of non-trending topics is the number and magnitude of spikes. The tweet rates for trending topics typically contains larger and more sudden spikes than that of non-trending topics. We reward such spikes by emphasizing them, while de-emphasizing smaller spikes. If the signal so-far in the transformation process is r, we transform it to

$$s(t) = |\dot{r}(t)|^{\alpha},$$

or in the discrete case,

$$s[n] = |r[n] - r[n-1]|^{\alpha},$$

where $\alpha > 1$.

Slicing

The signal resulting from the steps so far is as long as the entire time window from which all tweets were sampled. Such a long signal is not particularly useful as a reference signal. Recall from chapter 3 that to see how much the recent trajectory of the observed signal resembles part of a reference signal, we have to traverse the full length of the reference signal in order to find the piece that most closely resembles the recent observed trajectory. If the reference signal for a trending topic spans too long of a time window, only a small portion of it will represent activity surrounding the onset of the trend. In addition, it is inefficient to compare the recently observed trajectory to a reference signal that is needleslly long. Hence, it is necessary to select

a small slice of signal from the much longer rate signal. In the case of trending topics, we select a slice that terminates at the first onset of trend. That way, we capture the pattern of activity leading up to the trend onset, which is crucial for recognizing similar pre-onset activity in the observed signal. For non-trending topics, we assume that the rate signal is largely stationary and select slices with random start and end times. For simplicity, all slices are a fixed size.

Smoothing

Tweet rates, and the aforementioned transformations thereof, tend to be noisy, especially for small time bins. To mitigate this, we convolve the signal resulting from the previous step with a smoothing window.

Branching Processes and Exponential Variation

In the final step, we take the logarithm of each signal. It is well known that branching processes exhibit exponential variation.[**?**]

4.3 Experimental Setup

We have a rate of tweets over time

Chapter 5

Results and Discussion

In this chapter, I present the results of the trend detection experiment described in chapter 4. I show the quality of the trend detection algorithm using ROC curves and distributions of detection time relative to the true trend onset. I analyze the effect of the algorithm parameters on the tradeoff between false positive rate, true positive rate, and relative detection time. Finally, I propose parameter regimes appropriate for three situations: 1) the cost of a false positive outweighs the cost of a false negative, 2) the cost of a false negative outweighs the cost of a false positive, and 3) the costs of a false positive and a false negative are comparable.

5.1 ROC Curve Envelopes

Figures 5-2 and 5-1 shows the false positive rates (FPR) and true positive rates (TPR) that result from varying each detection parameter, aggregated over all combinations of the remaining parameters. The left side of each plot shows a scatter plot of false positive and true positive rates, while the right side shows the upper-left-most envelope of the set of all ROC curves.

5.2 Examples

True positives. Slow rising vs fast rising.

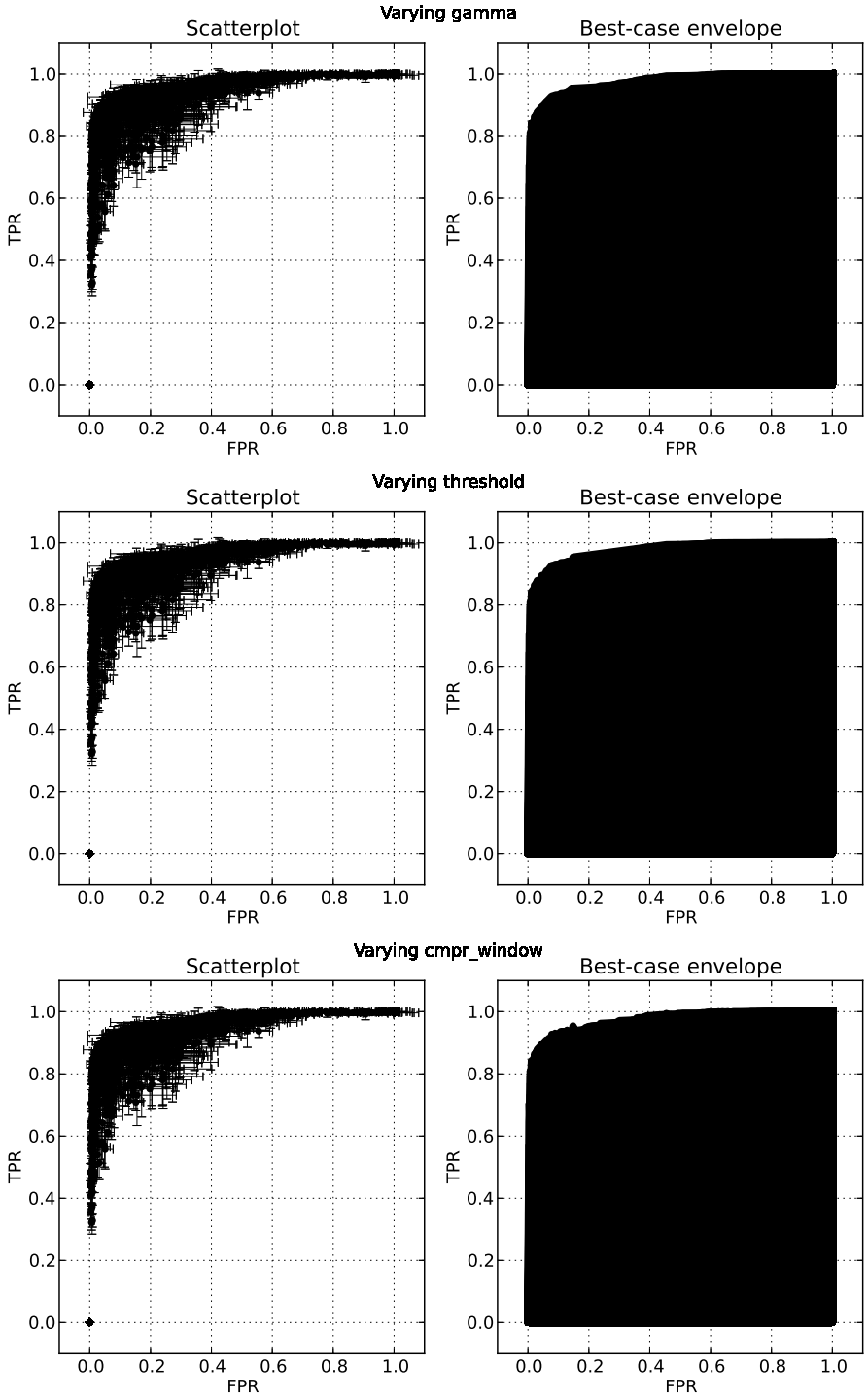

Figure 5-1:

True negatives.

False positives. Stuff with significant volume or spikes.

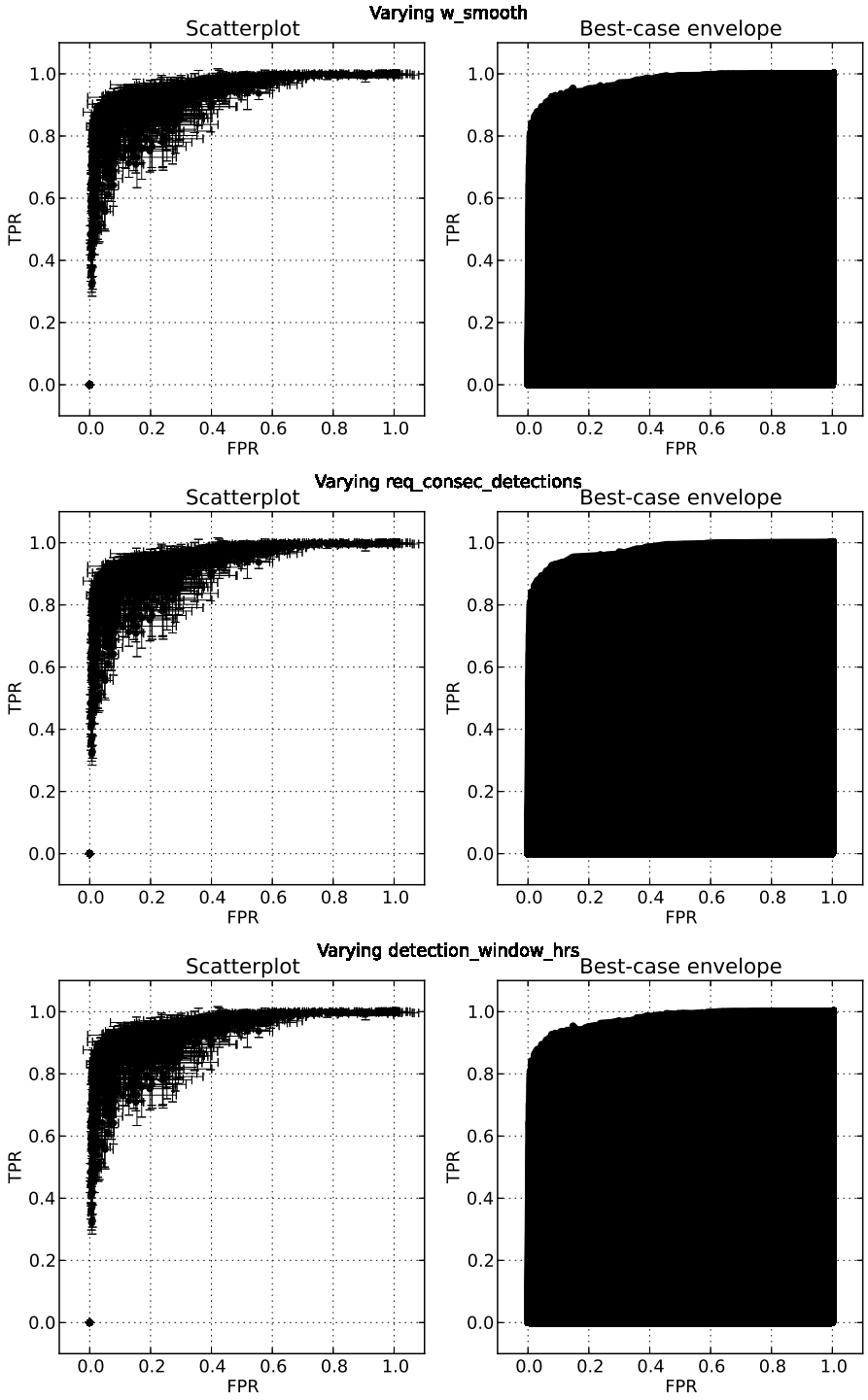

Figure 5-2:

False negatives. Stuff with too little volume or not enough spikes.

5.3 Effect of Parameters

For each ROC curve, we have a parameter that varies to produce the ROC curve, which we will call the variable paramter, and a fixed combination of the remaining paramters, which we will call the constant parameters.

Do we move up or down the curve?

We show how varying a given parameter p trades off FPR for TPR by computing the discrete derivative of FPR and TPR with respect to p. For each ROC curve, corresponding to the variable parameter p and some fixed combination of remaining parameters, we compute

$$\Delta_{p,i}^{FPR} = \frac{FPR(p_i) - FPR(p_{i-1})}{p_i - p_{i-1}} \tag{5.1}$$

$$\Delta_{p,i}^{TPR} = \frac{TPR(p_i) - TPR(p_{i-1})}{p_i - p_{i-1}} \tag{5.2}$$

for each ROC curve associated with p and for i ranging from the second to the last value of p in increasing order. If each point on the ROC curve is produced by multiple trials, we compute the above for all possible combinations of ROC curves. Finally, we compute the above across all combinations of fixed parameters.

The result is a distribution of discrete derivatives of FPR and TPR with respect to a variable parameter of interest p which highlights the effect of p on tradeoffs between FPR and TPR. We can refer this effect as moving "up" the ROC curve, or "down" the ROC curve. If most of the mass of Δ_p^{FPR} and Δ_p^{TPR} is at values greater than 0, then an increase in p causes a decrease in FPR at the expense of lower TPR, moving down the curve. If, on the other hand, most of the mass is at values less than zero, an increase in p causes an increase in TPR at the expensive of higher FPR, moving up the curve.

Sometimes, the curve moves neither toward $(0, 0)$ ("down the curve") nor toward $(1, 1)$ ("up the curve") but toward $(0, 1)$ or $(1, 0)$. The former represents an increase in TPR in addition to a decrease in FPR — a win-win situation. The latter represents the exact opposite of that — an increase in FPR and a decrease in TPR.

Note that consecutive $(x0, 0)$ or $(1, 1)$ points on an ROC curve were not counted when computing Δ_p^{FPR} and Δ_p^{TPR}, since no amount of change in the variable paramter can move further and the resulting delta of 0 is meaningless.

Figures 5-3 through 5-8 shows this effect for each parameter.

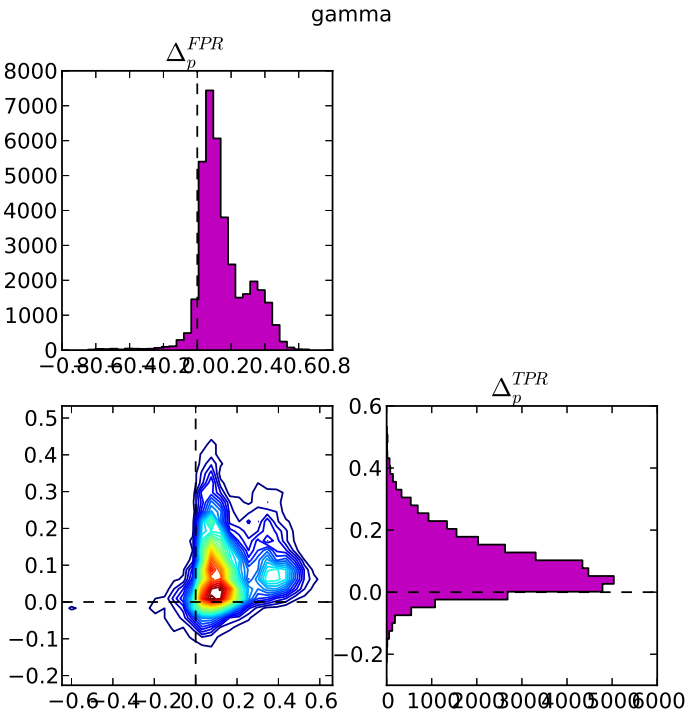

Figure 5-3: Distribution of $[\Delta_p^{FPR}, \Delta_p^{TPR}]^T$ for variable parameter GAMMA.

It is clear that varying some parameters has the same effect regardless of the remaining parameters, at least within the range of values explored.

Increasing GAMMA always moves us up the curve. Increasing THRESHOLD always moves us down the curve. Increasing REQCONSECDETECTIONS always moves us down the curve.

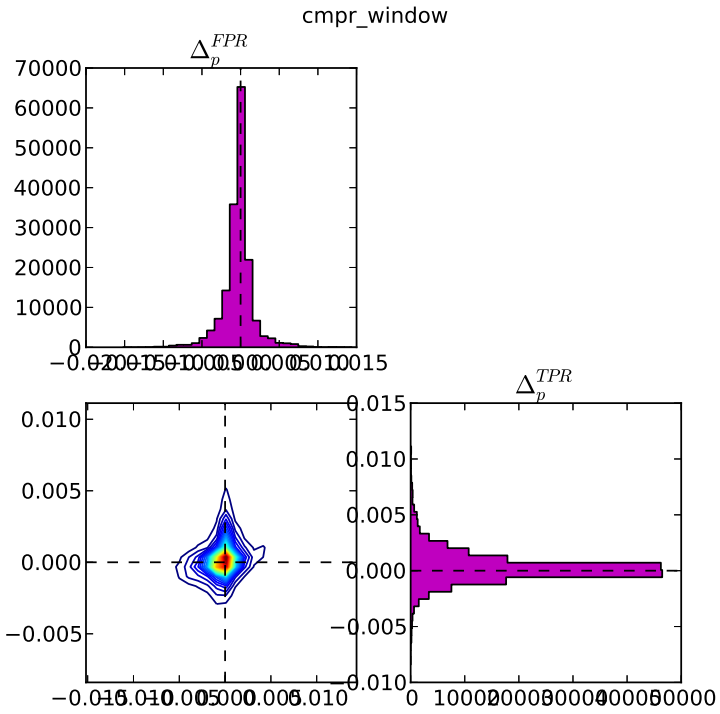

Figure 5-4: Distribution of $[\Delta_p^{FPR}, \Delta_p^{TPR}]^T$ for variable parameter CmprWindow.

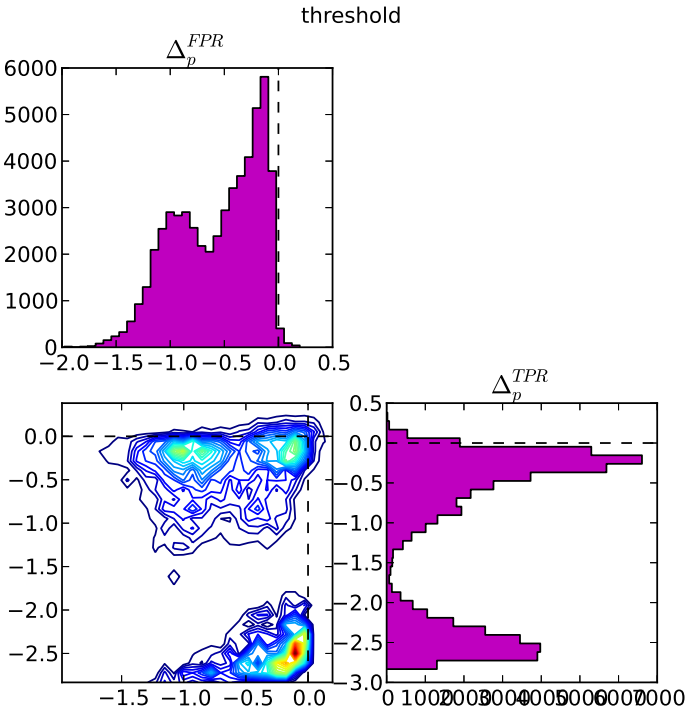

Figure 5-5: Distribution of $[\Delta_p^{FPR}, \Delta_p^{TPR}]^T$ for variable parameter THRESHOLD.

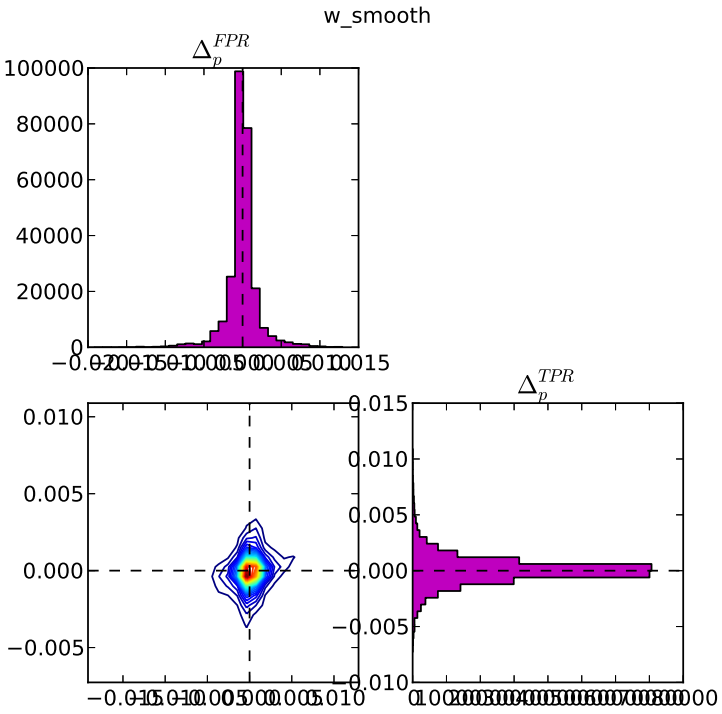

Figure 5-6: Distribution of $[\Delta_p^{FPR}, \Delta_p^{TPR}]^T$ for variable parameter WSmooth.

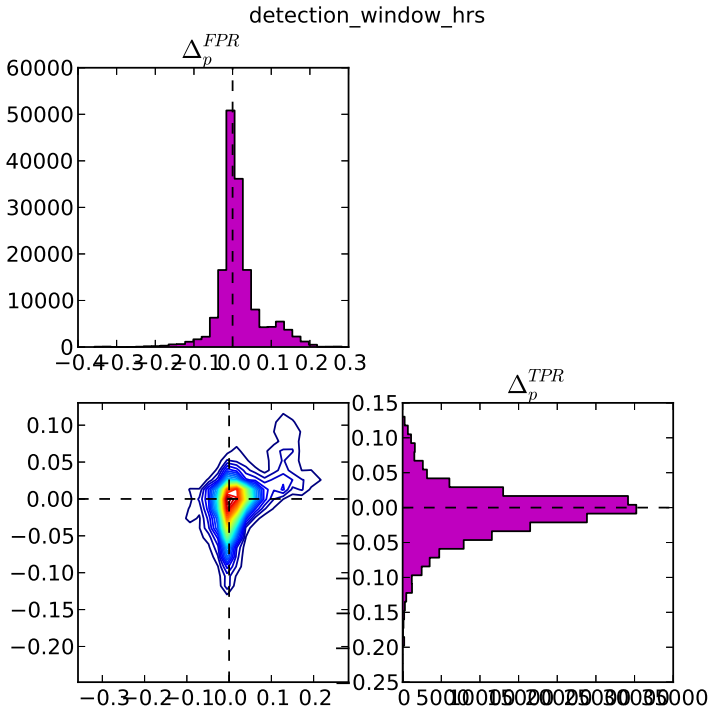

Figure 5-7: Distribution of $[\Delta_p^{FPR}, \Delta_p^{TPR}]^T$ for variable parameter DetectionWindowHrs.

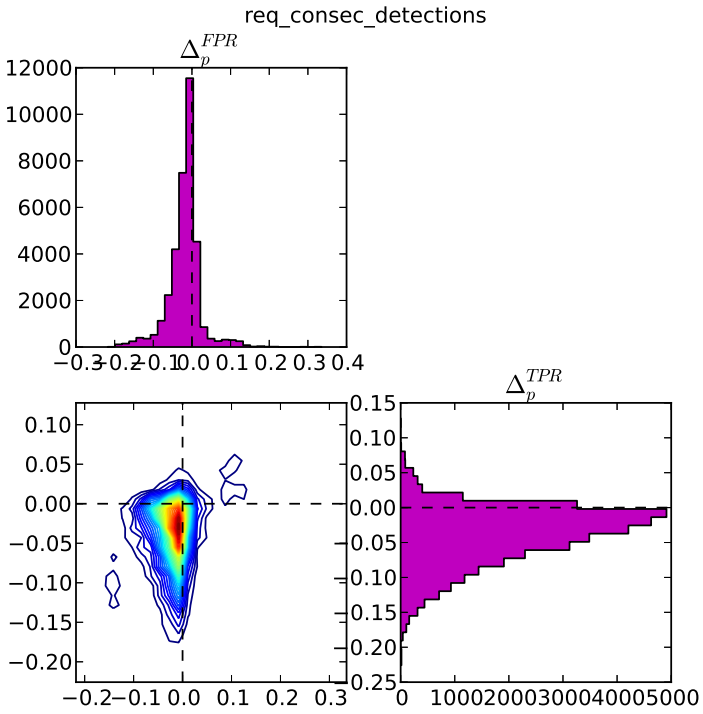

Figure 5-8: Distribution of $[\Delta_p^{FPR}, \Delta_p^{TPR}]^T$ for variable parameter ReqConsecDetections.

Some variable parameters, however, do not appear to have a consistent effect across all combinations of constant parameters. To investigate how pairs of parameters interact, we show the distribution of $[\Delta_p^{FPR}, \Delta_p^{TPR}]^T$ resulting from a given variable paramter, various settings of a chosen constant parameter, and all combinations of the remaining constant parameters. Figures 5-9 through 5-14 show these effects.

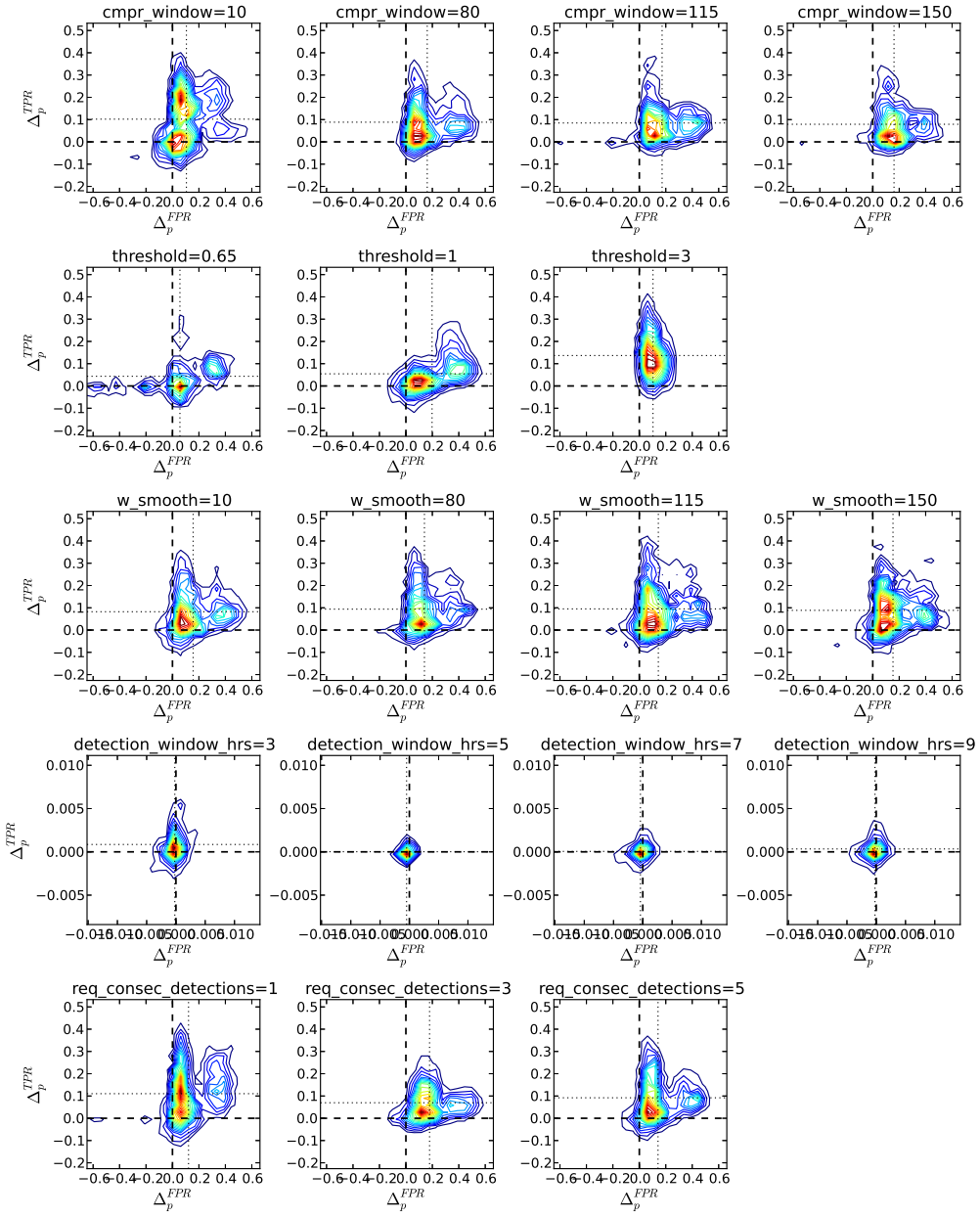

Figure 5-9: Secondary effects of constant parameters for variable parameter GAMMA

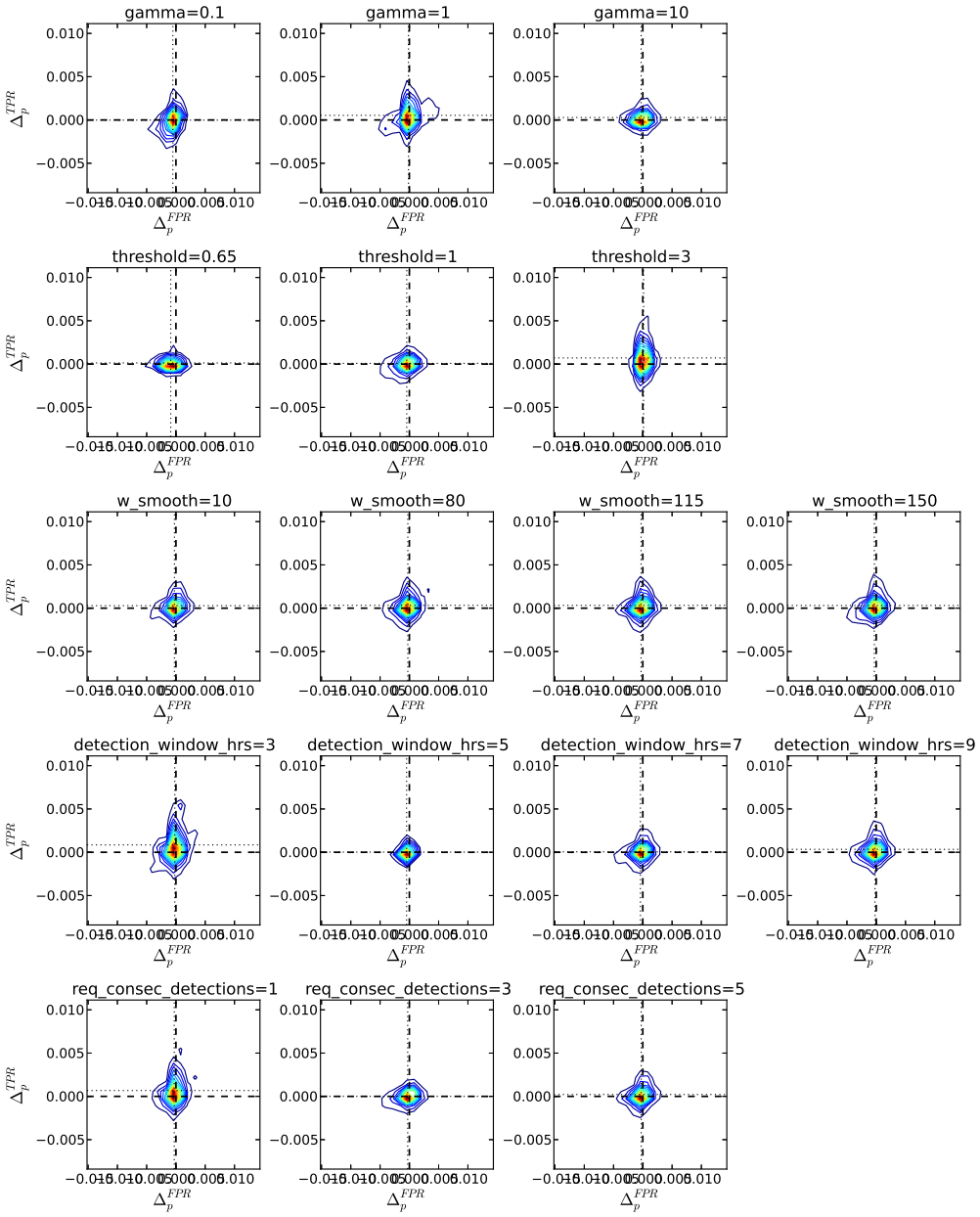

Figure 5-10: Secondary effects of constant parameters for variable parameter CmprWindow

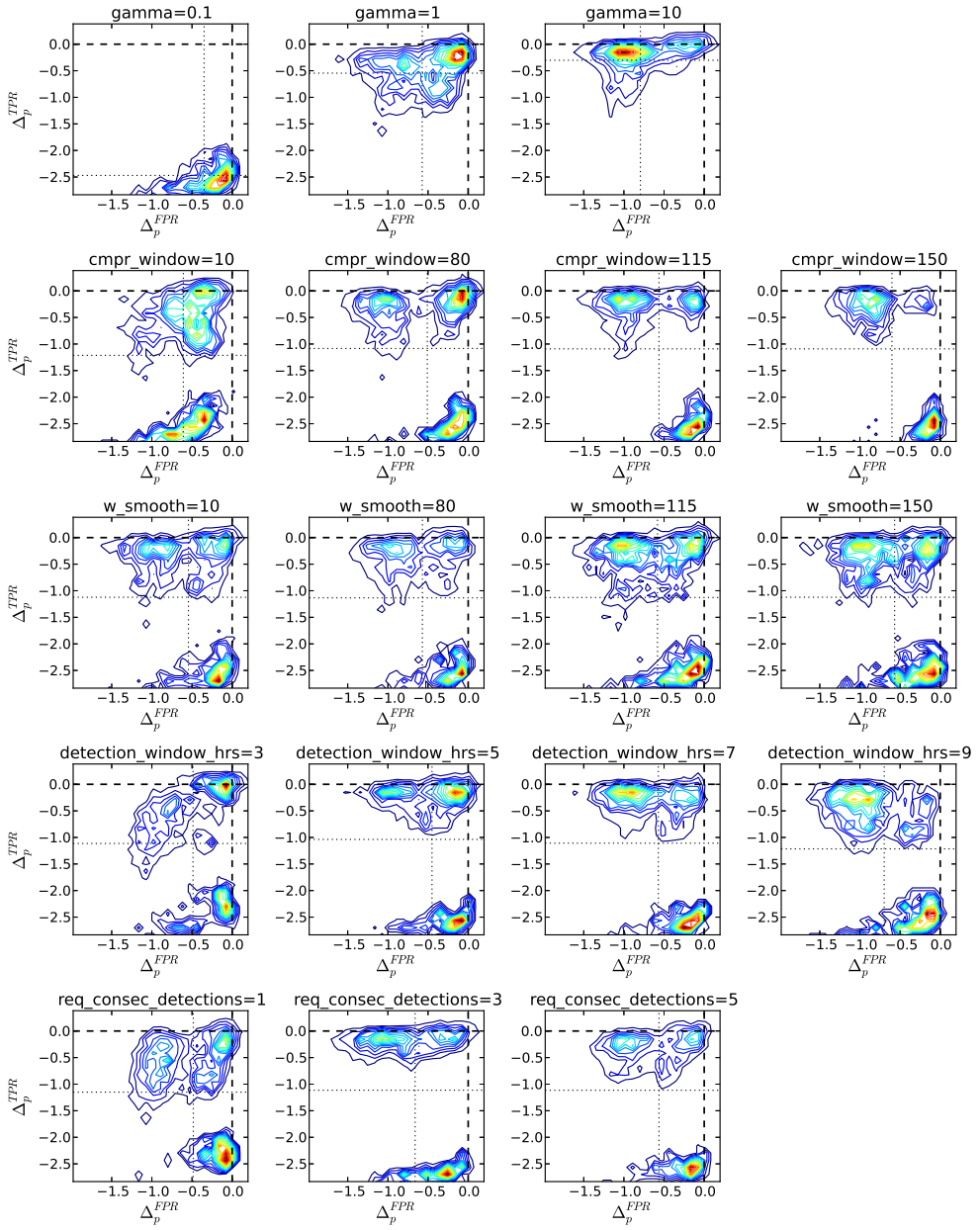

Figure 5-11: Secondary effects of constant parameters for variable parameter Threshold

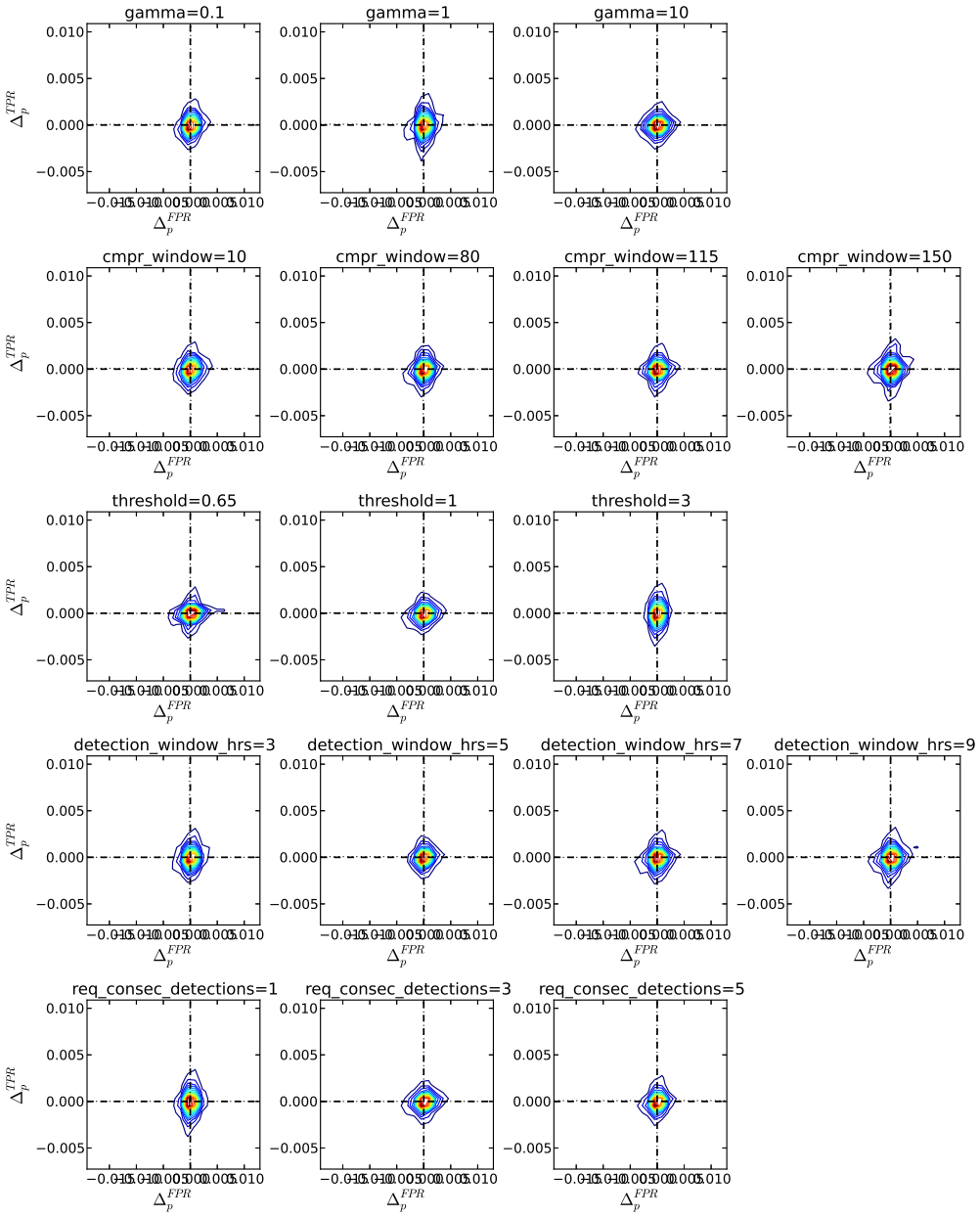

Figure 5-12: Secondary effects of constant parameters for variable parameter WSMOOTH

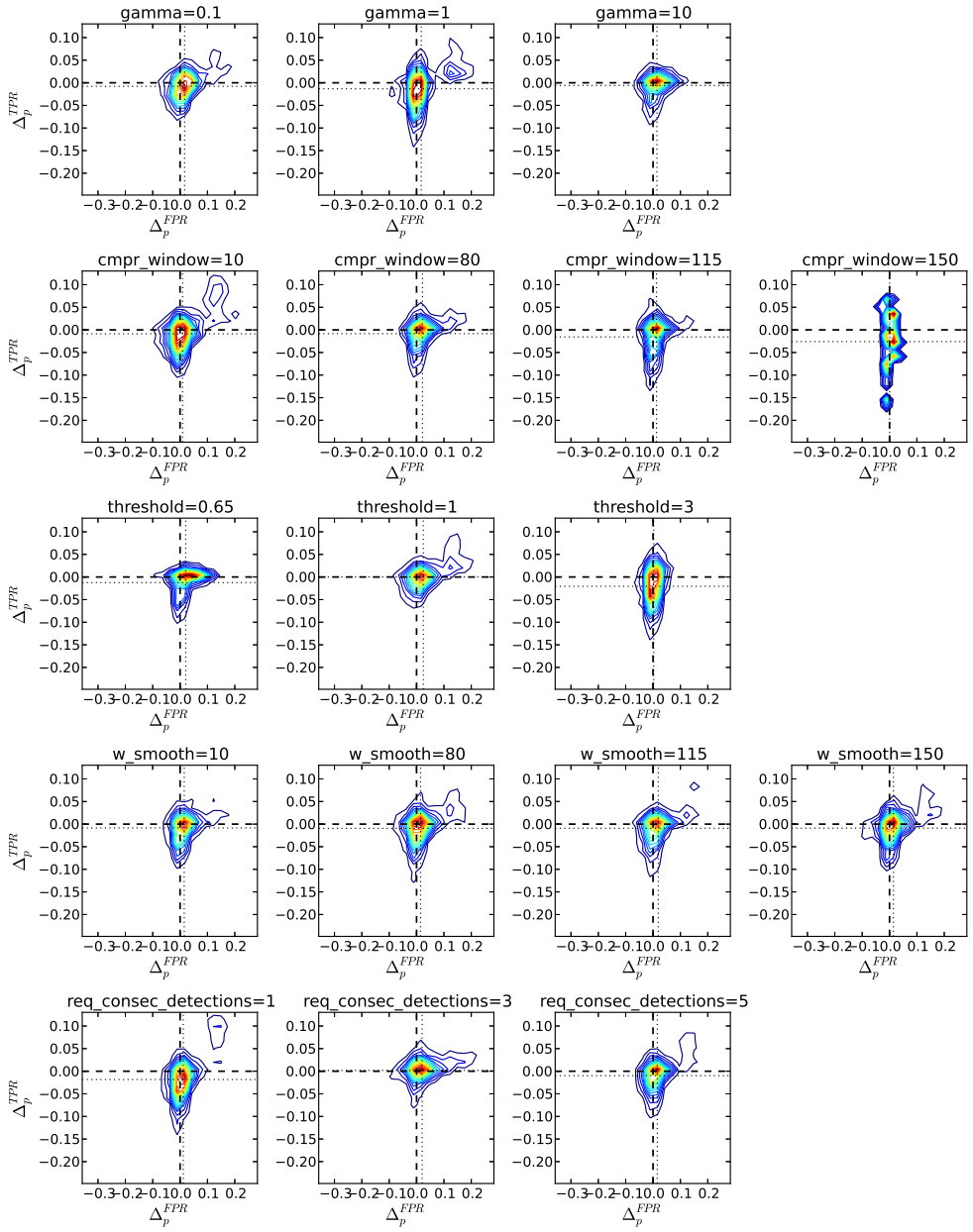

Figure 5-13: Secondary effects of constant parameters for variable parameter DetectionWindowHrs

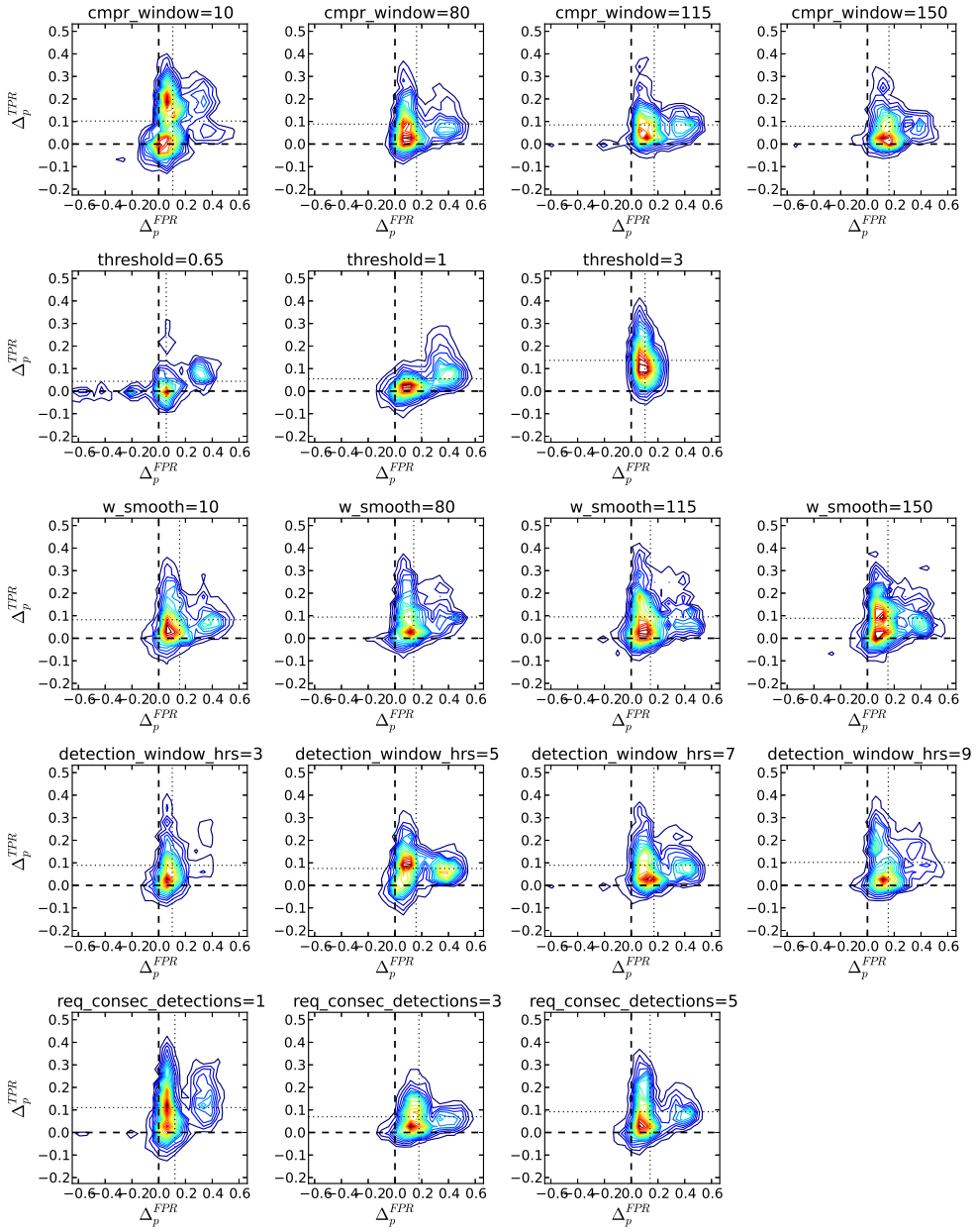

Figure 5-14: Secondary effects of constant parameters for variable parameter ReqConsecDetections

Figures 5-15 through 5-26 show the top and bottom ROC curves ranked by mean Δ_p^{FPR} and mean Δ_p^{TPR}. For each variable parameter, the ROC curves with the top mean Δ_p^{FPR} show the values of the constant parameters for which the variable parameter has the largest positive effect on FPR. Similarly, the bottom-ranked ROC curves represent ROC curves for which the variable parameter has the lowest positive (possibly negative) effect on FPR.

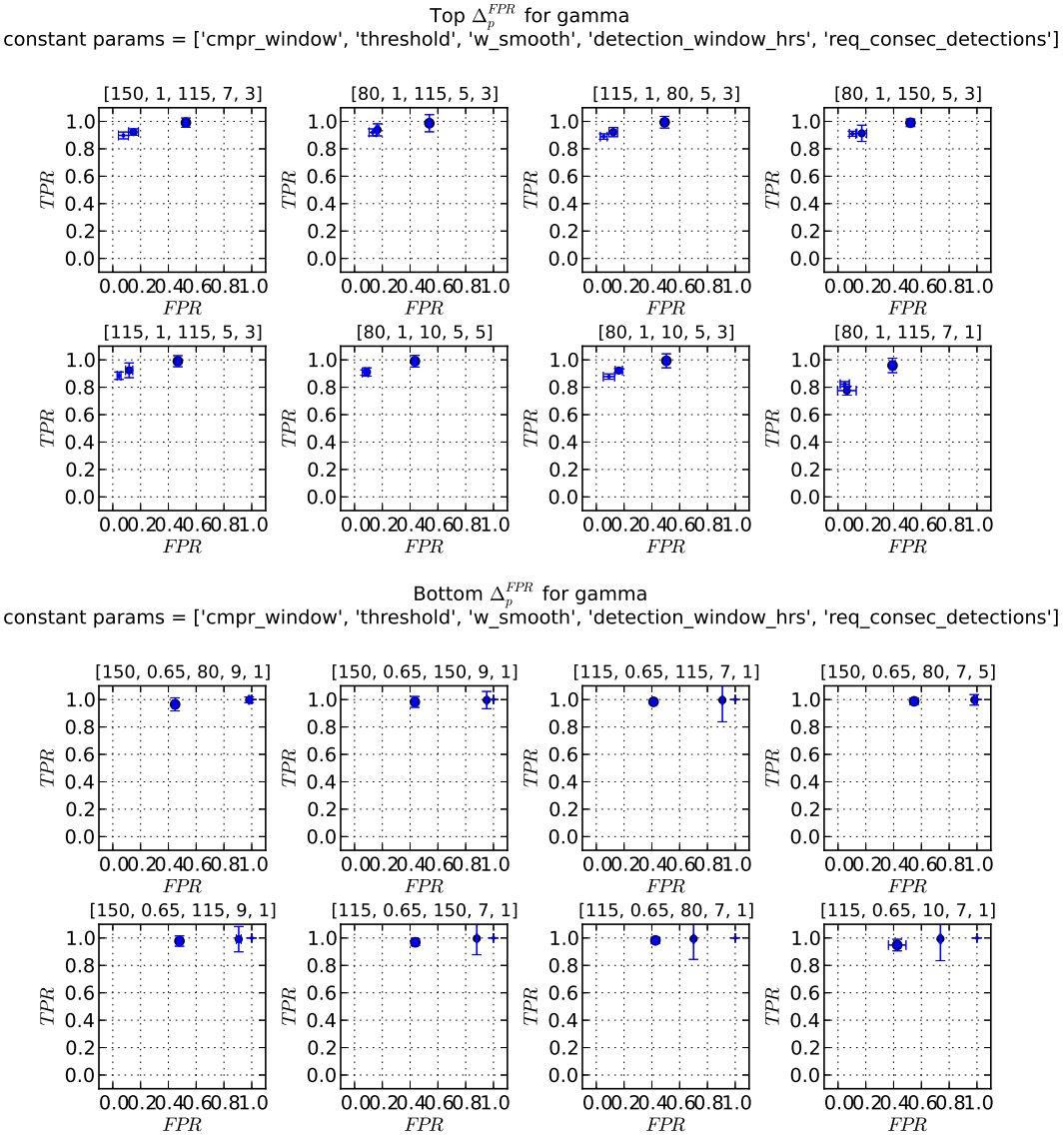

Figure 5-15: Top and bottom Δ_p^{FPR} for Gamma

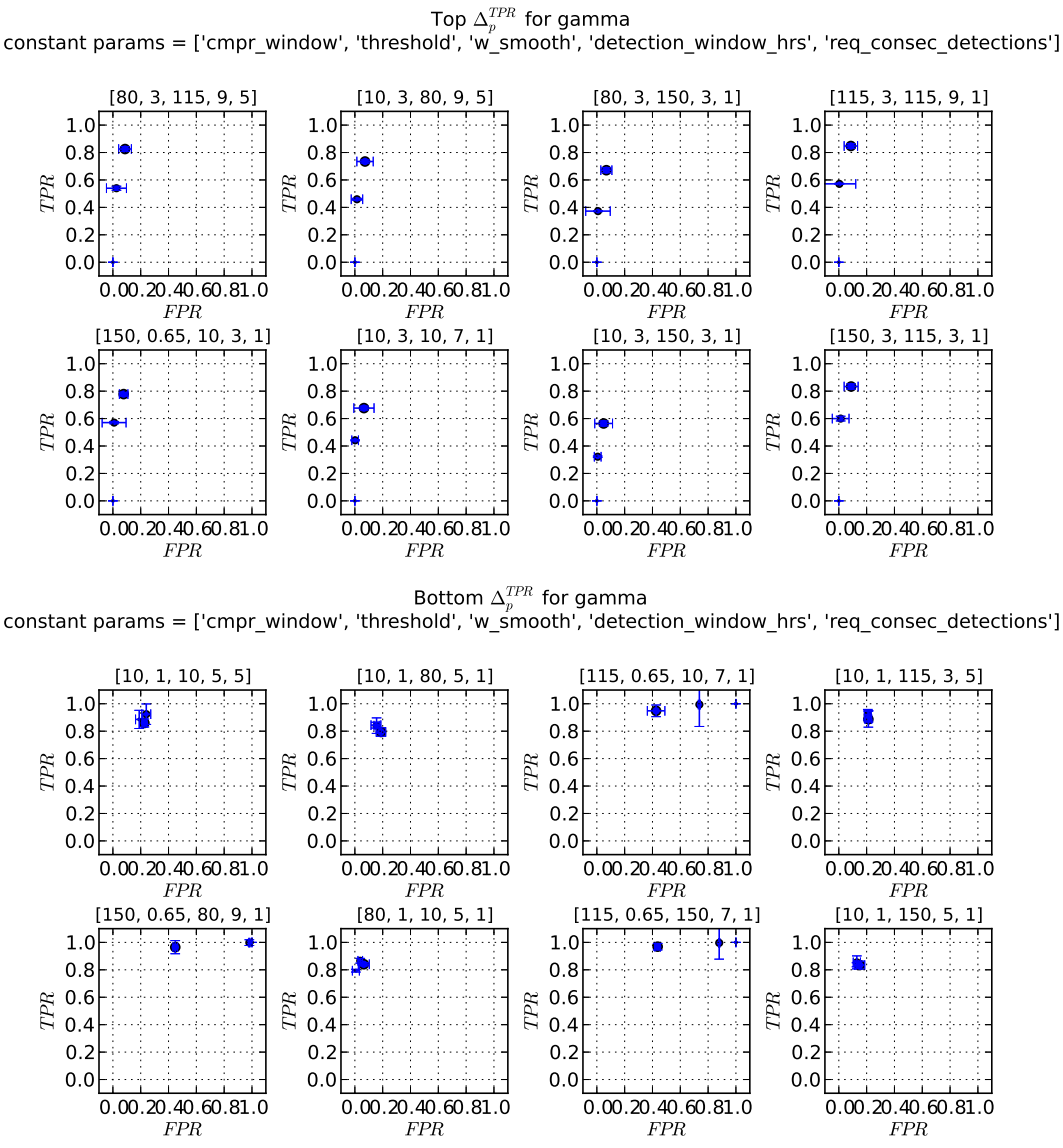

Figure 5-16: Top and bottom Δ_p^{TPR} for Gamma

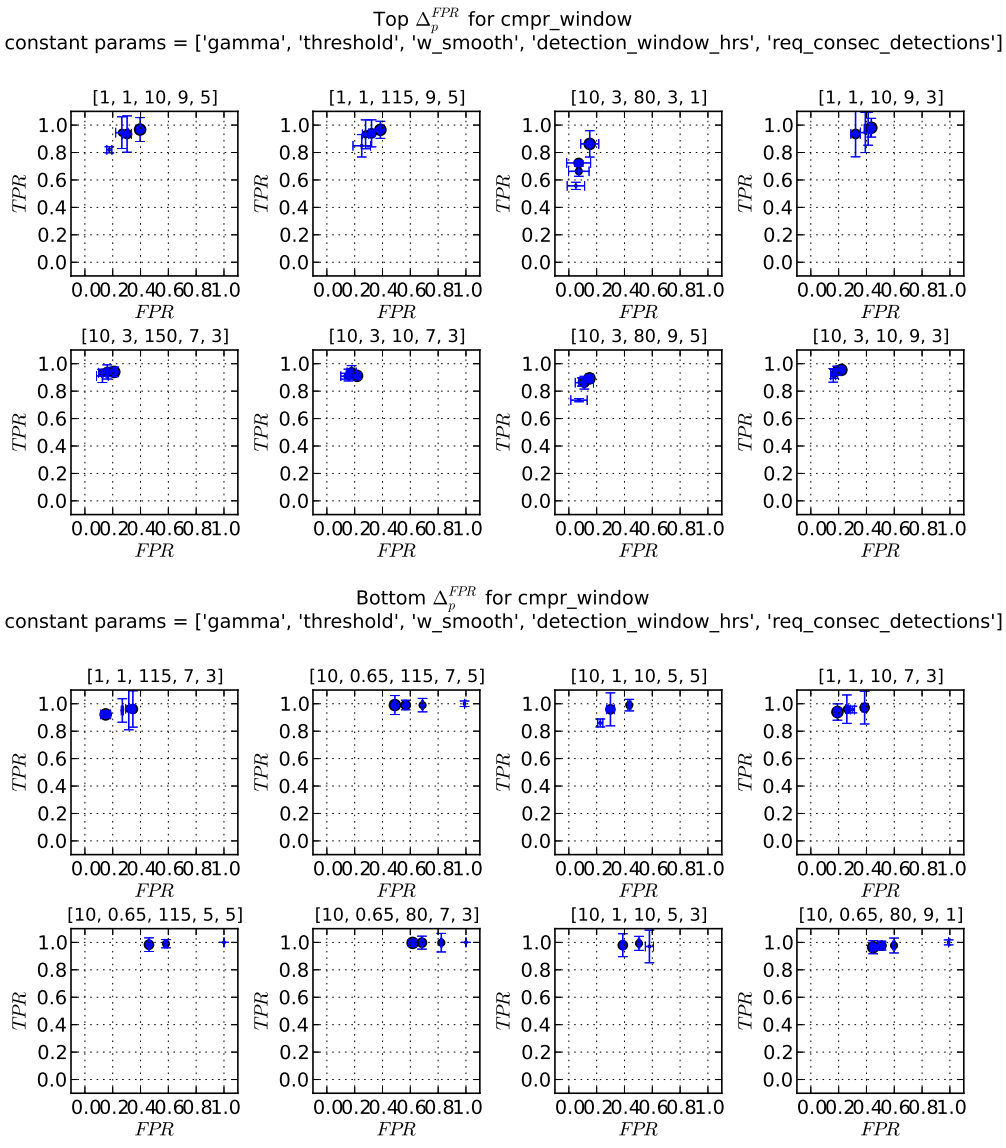

Figure 5-17: Top and bottom Δ_p^{FPR} for CmprWindow

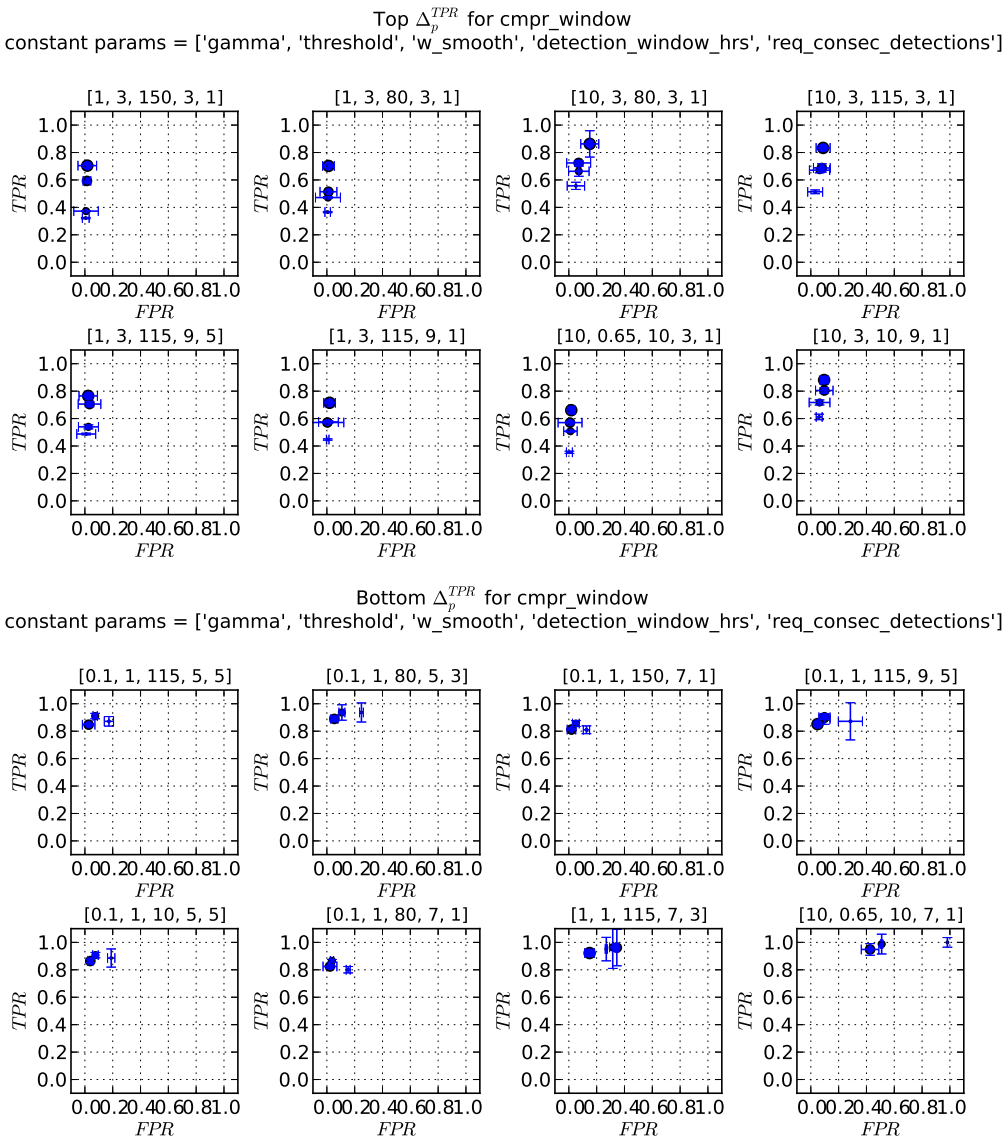

Figure 5-18: Top and bottom Δ_p^{TPR} for CmprWindow

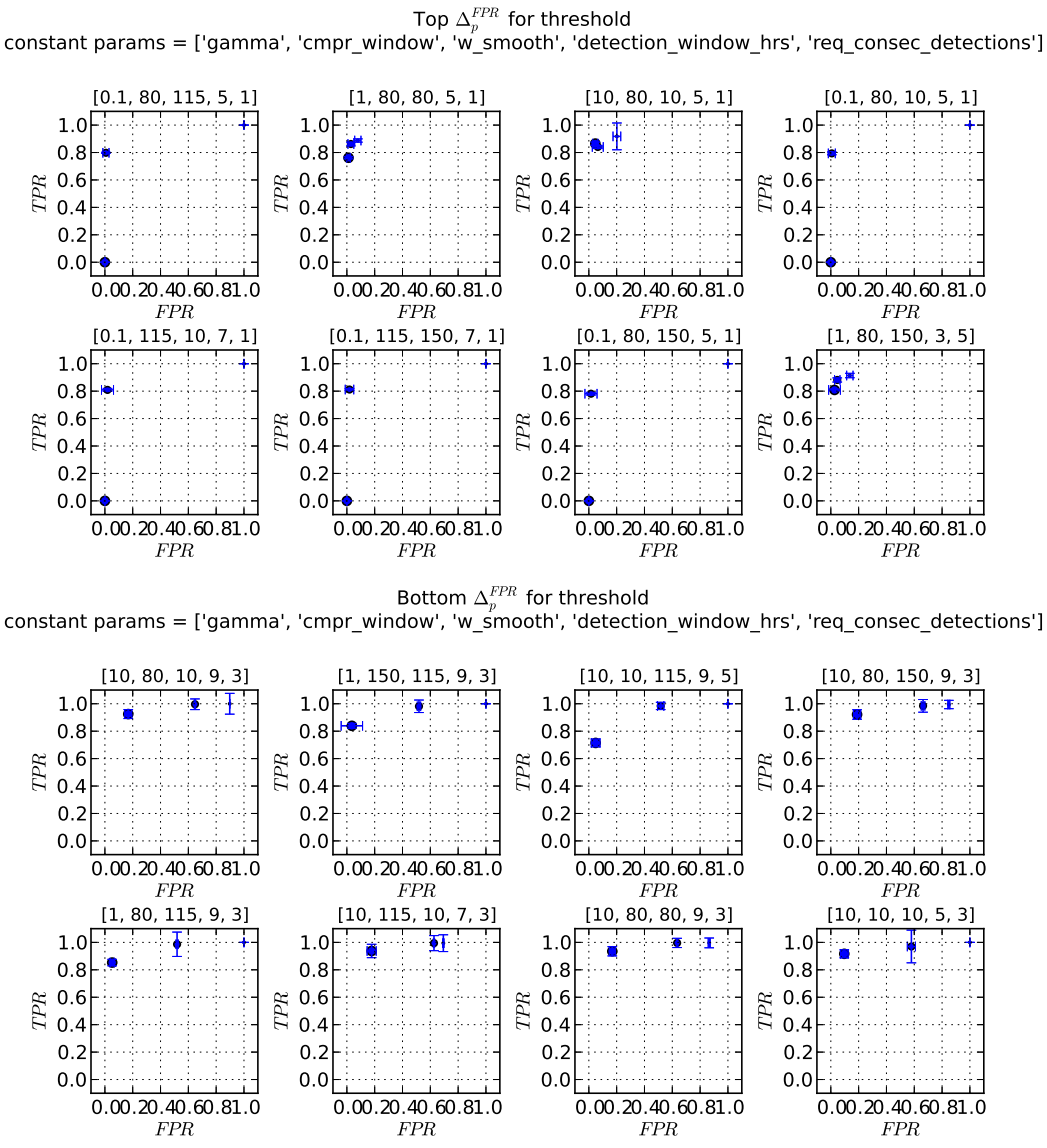

Figure 5-19: Top and bottom Δ_p^{FPR} for Threshold

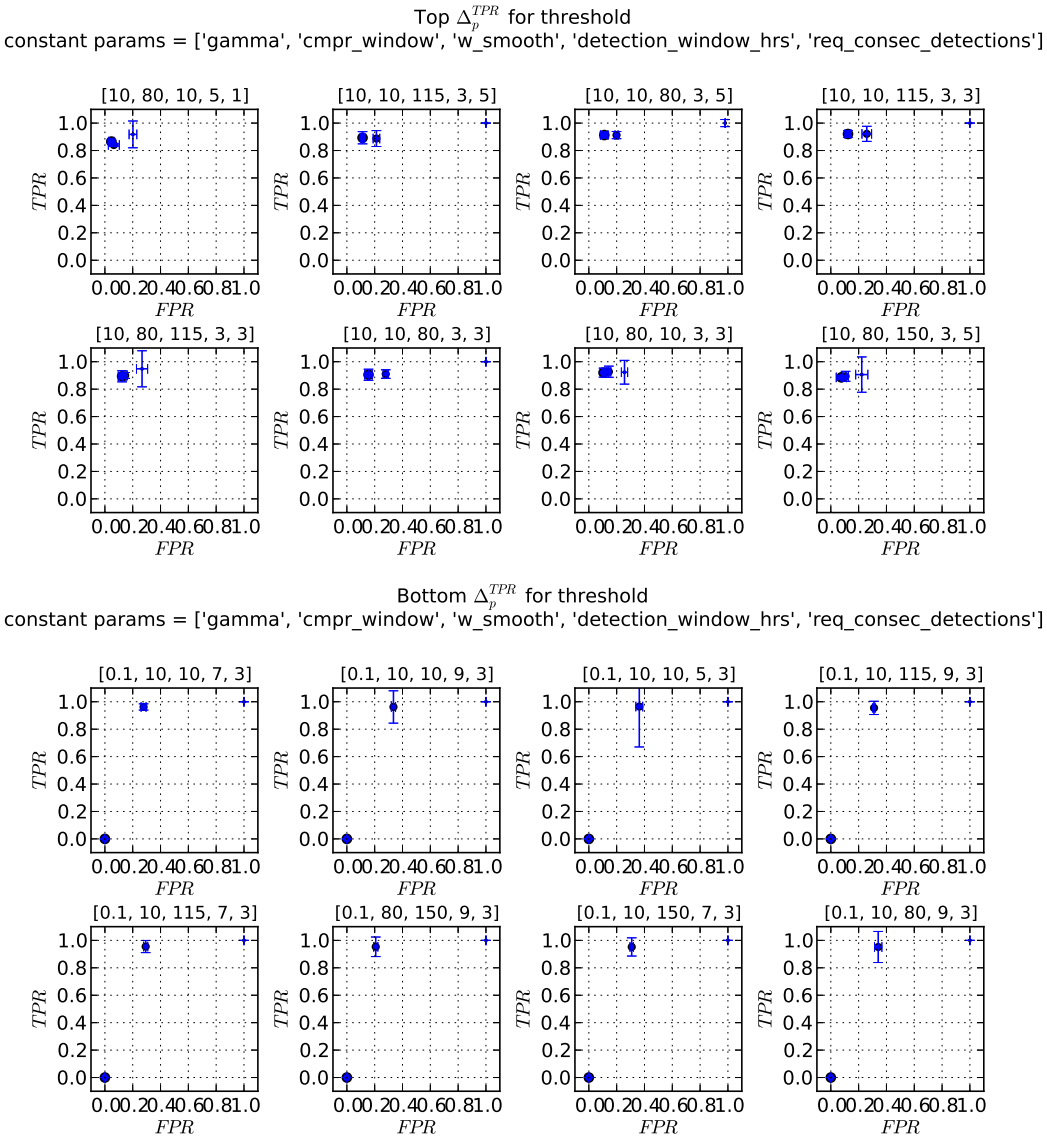

Figure 5-20: Top and bottom Δ_p^{TPR} for Threshold

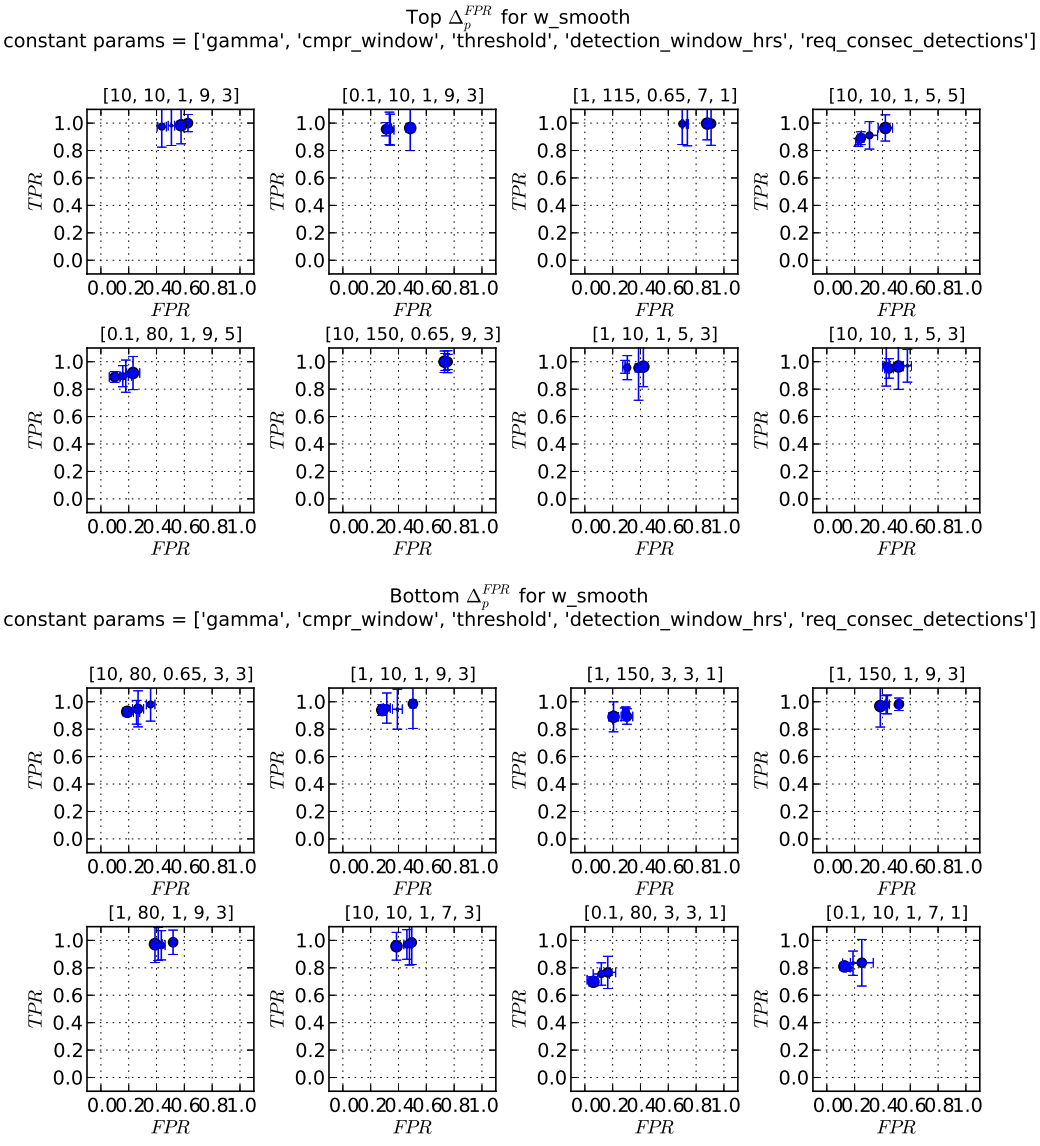

Figure 5-21: Top and bottom Δ_p^{FPR} for WSmooth

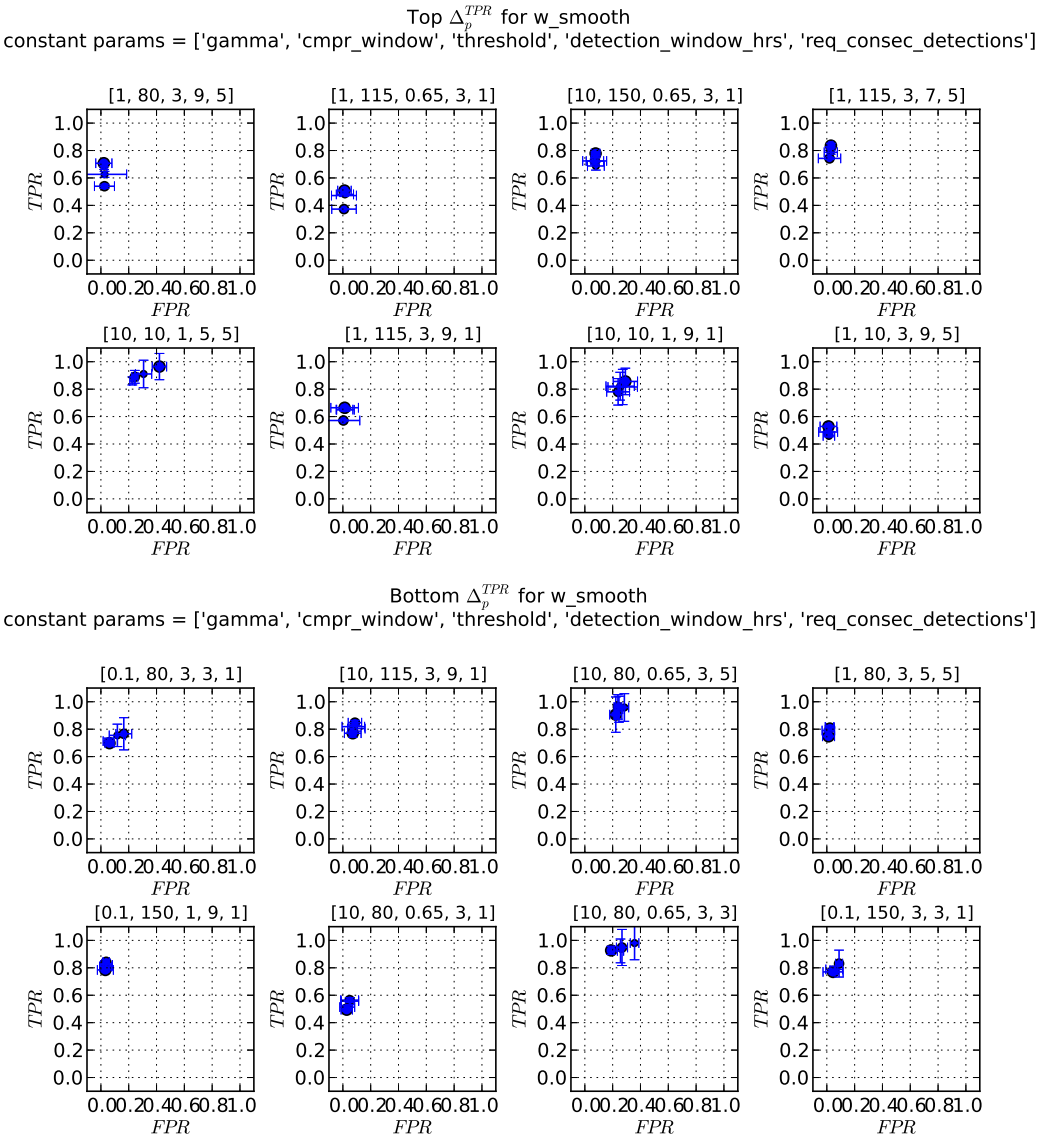

Figure 5-22: Top and bottom Δ_p^{TPR} for WSmooth

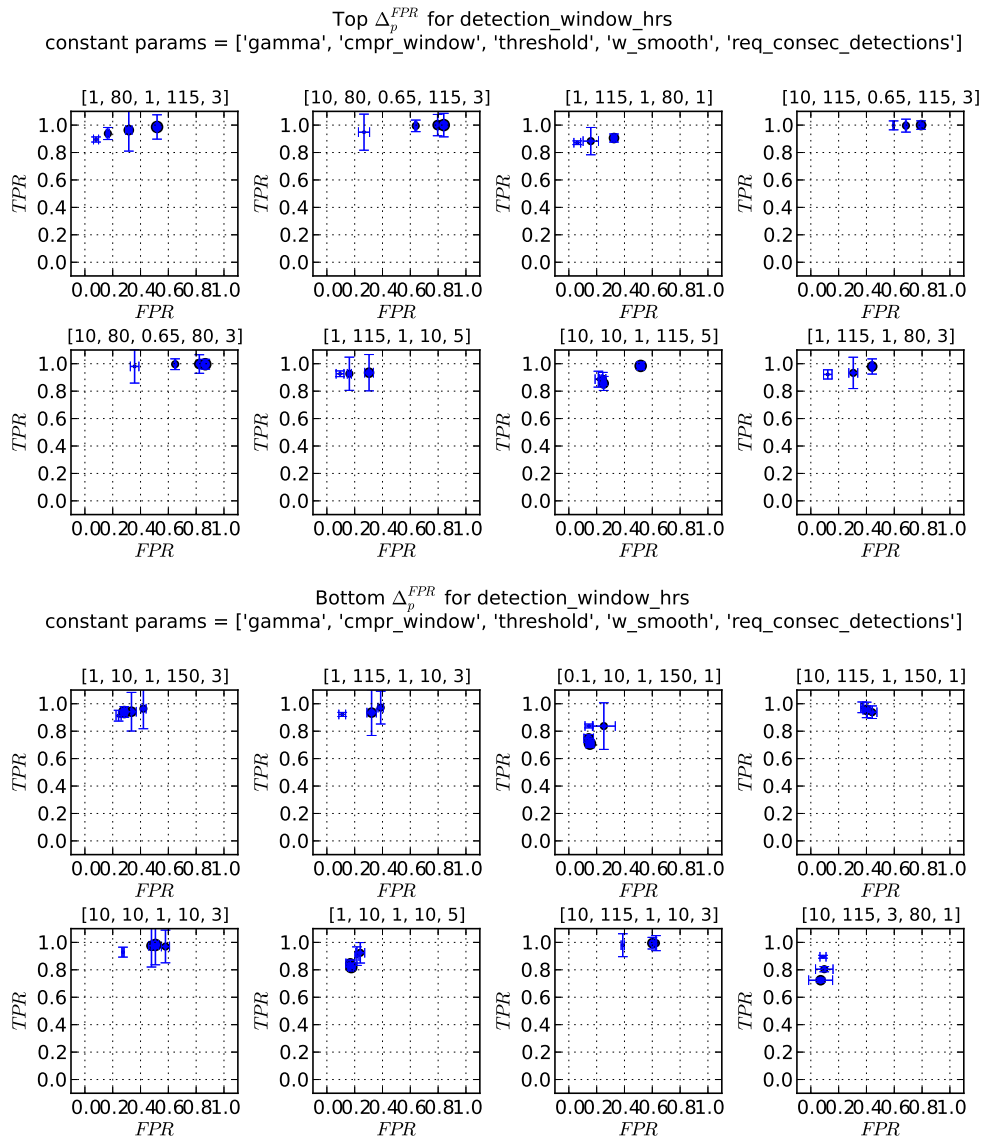

Figure 5-23: Top and bottom Δ_p^{FPR} for DetectionWindowHrs

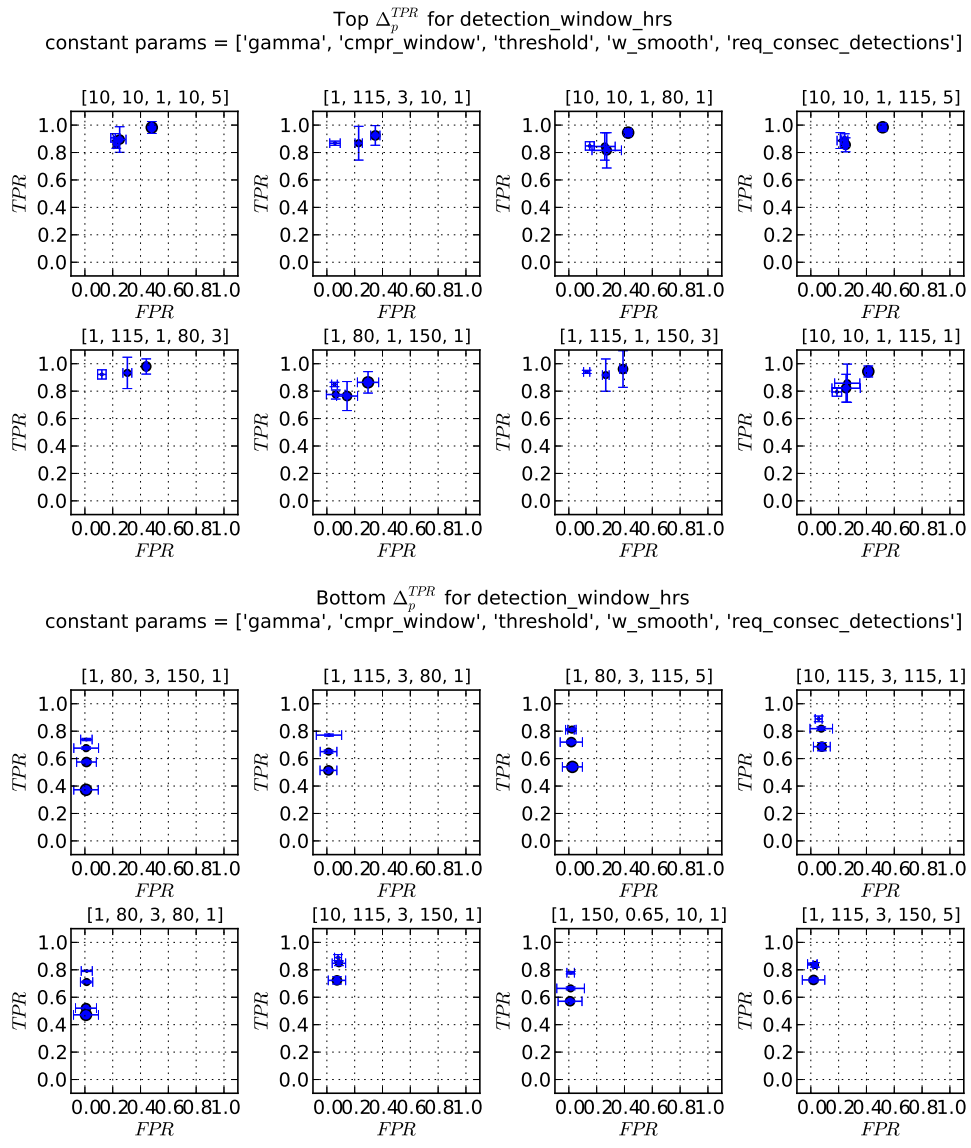

Figure 5-24: Top and bottom Δ_p^{TPR} for DetectionWindowHrs

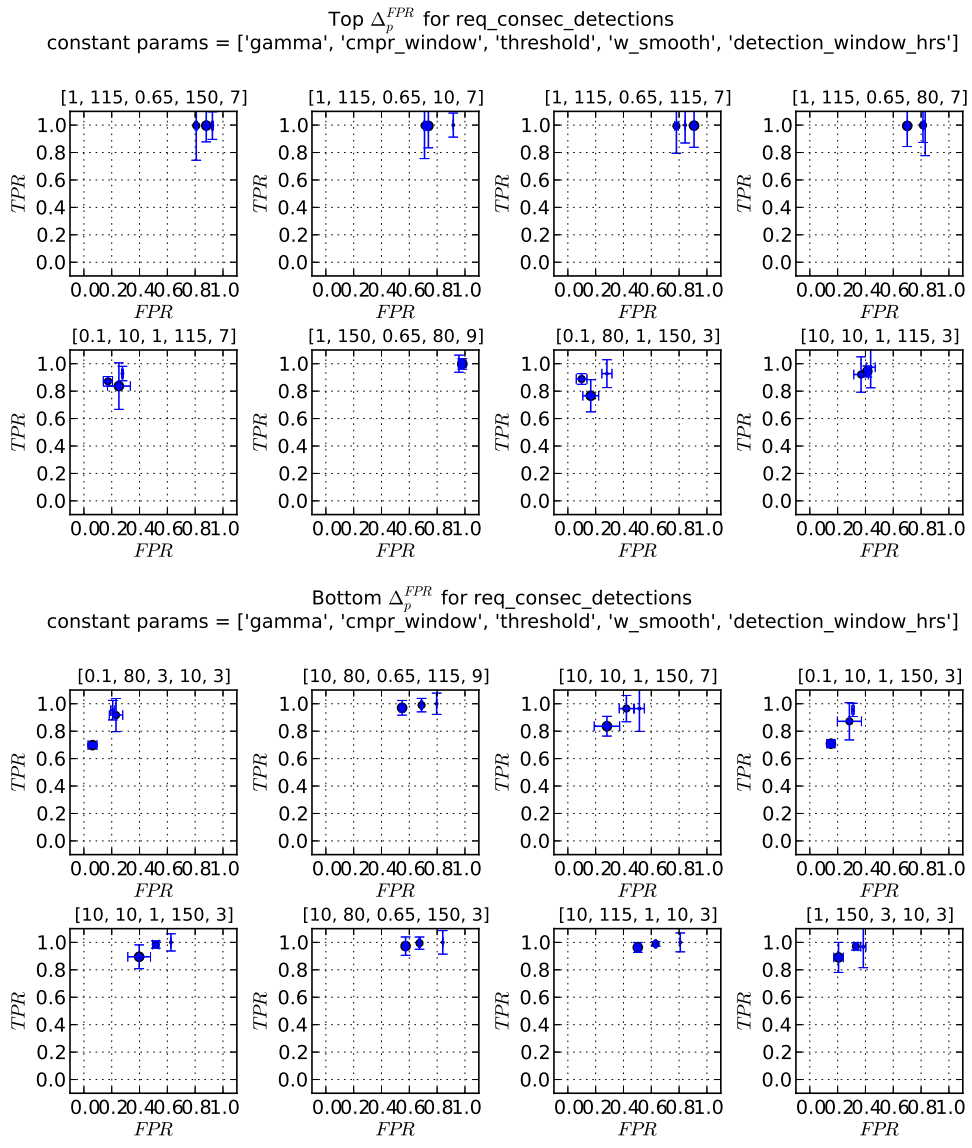

Figure 5-25: Top and bottom Δ_p^{FPR} for ReqConsecDetections

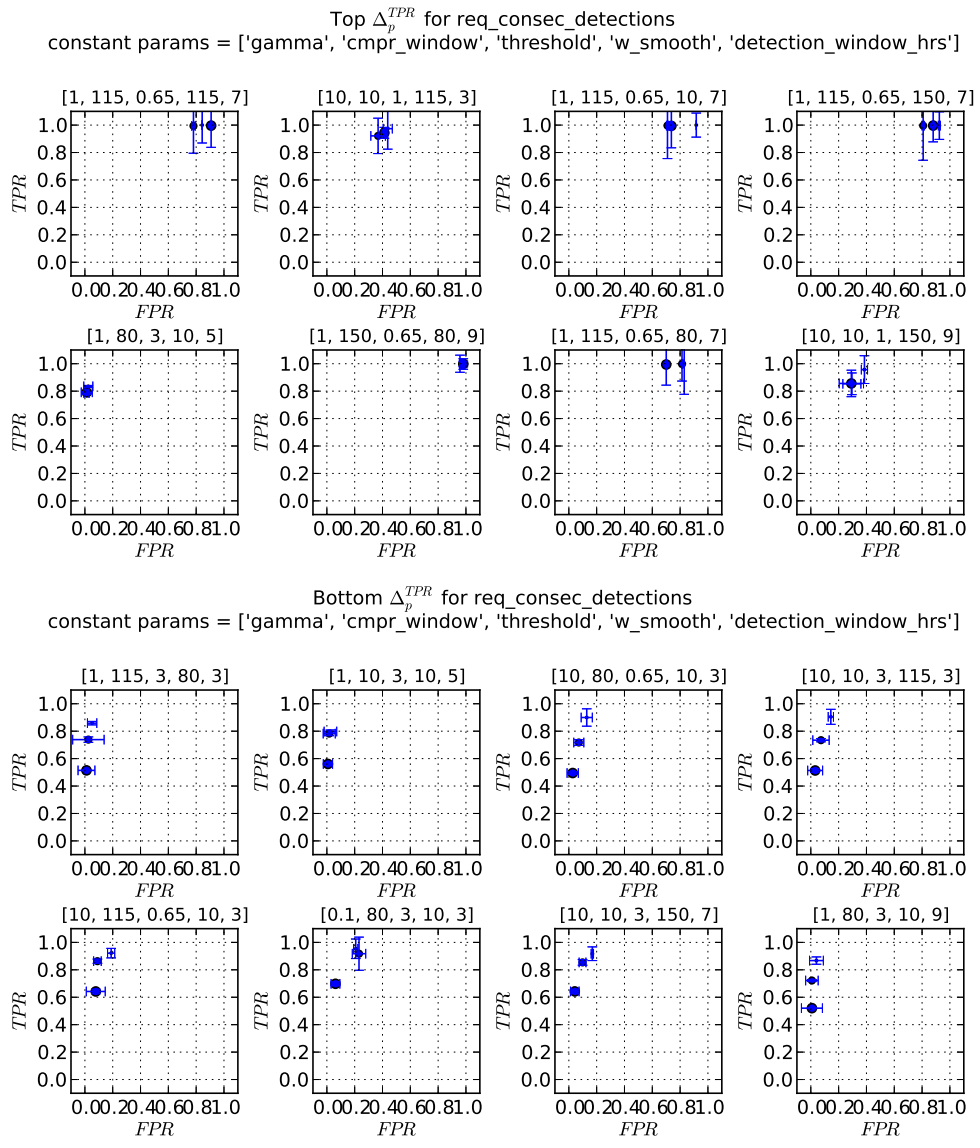

Figure 5-26: Top and bottom Δ_p^{TPR} for ReqConsecDetections

5.4 Recommendations

Chapter 6

Summary and Contributions