13. Online Learning

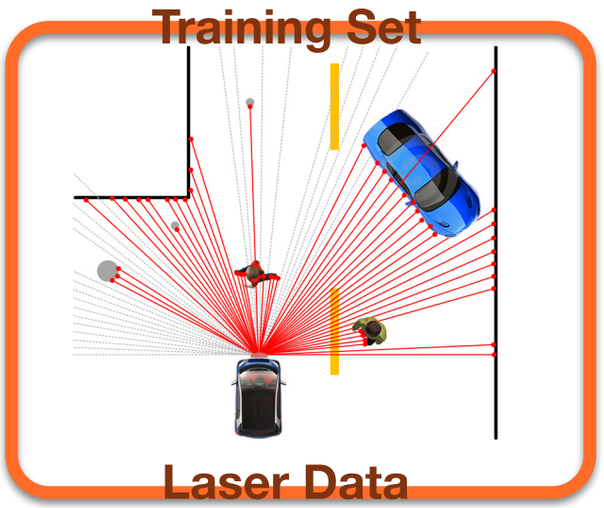
Motivation

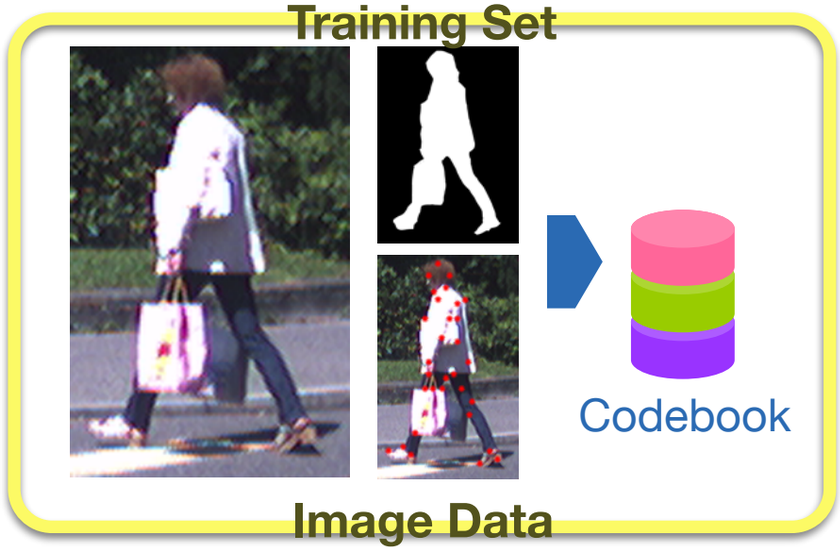
- So far, we have handled offline learning methods, but very often data is observed in an **ongoing** process
- Therefore, it would be good to adapt the learned models with newly arriving data **without** having to consider old data
- This is called **online** learning
- Major benefits:
  - algorithms are **adaptive** to new observations
  - learning is usually **faster**

Example: Pedestrian Detection

Offline Approach:

1. Collect and annotate a large data set from different sensor modalities, here: **camera** and **2D laser**



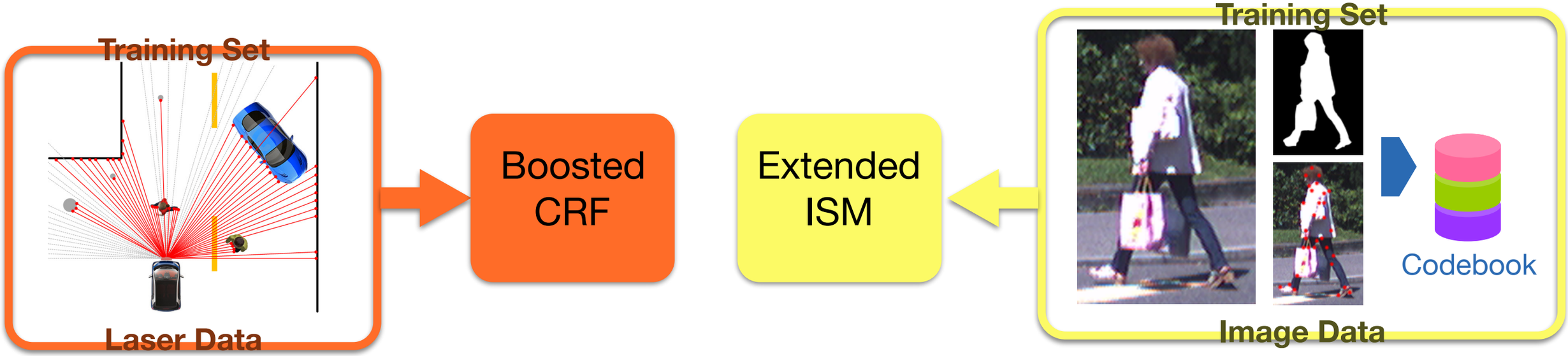


[Spinello, Triebel, and Siegwart, IJRR 2010]

Example: Pedestrian Detection

Offline Approach:

1. Collect and annotate a large data set from different sensor modalities, here: **camera** and **2D laser**
2. **Train** a classifier for each sensor modality

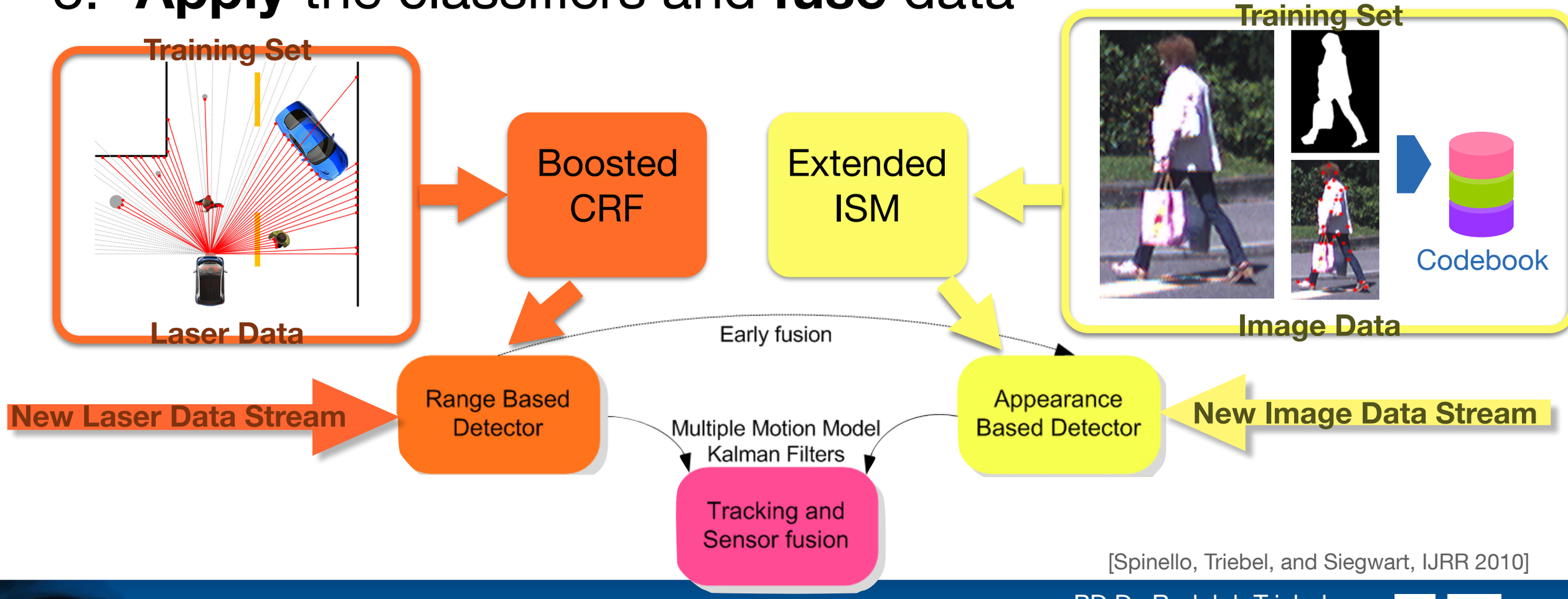


[Spinello, Triebel, and Siegwart, IJRR 2010]

Example: Pedestrian Detection

Offline Approach:

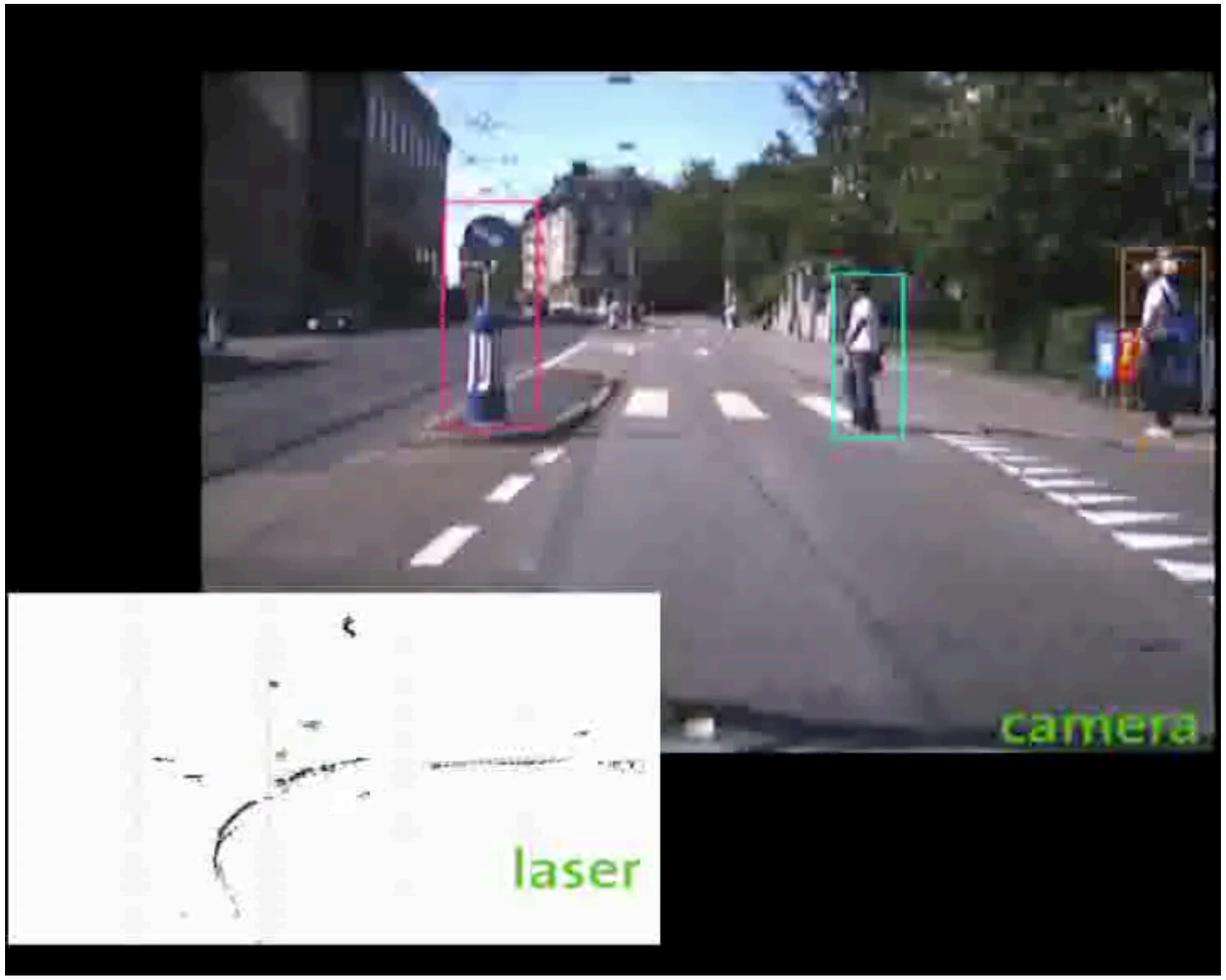
1. Collect and annotate a large data set from different sensor modalities, here: **camera** and **2D laser**
2. **Train** a classifier for each sensor modality
3. **Apply** the classifiers and **fuse** data



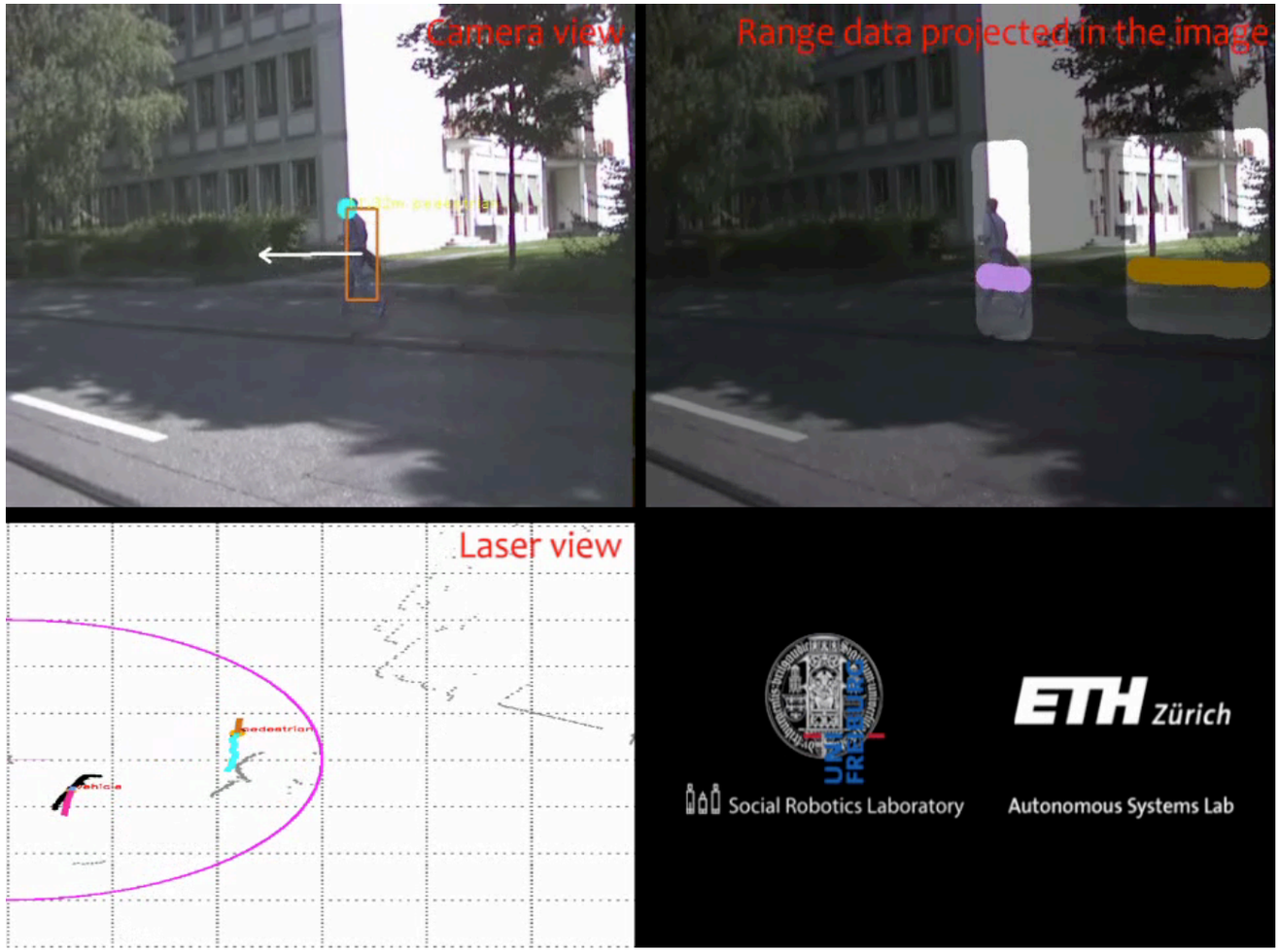
[Spinello, Triebel, and Siegwart, IJRR 2010]

Example: Pedestrian Detection

Testing Phase:



Fused pedestrian detection and tracking



Fused detection and tracking of cars and pedestrians

But: non-adaptive; large training set required

[Spinello, Triebel, and Siegwart, IJRR 2010]

Major Problem of this Approach

Offline learning:

- cannot adapt to new environments
- cannot learn new classes
- cannot consider new instances of a given class
- is often very slow
- requires a large amount of manually annotated training data

Ideas:

- use **online** learning for adaptivity
- use **active** learning to reduce (human) work load

Clarification of Notions

There are (at least) three notions:

- **incremental** learning:
  - model is updated with new samples, but old samples might be considered again
- **online** learning:
  - after considering a data sample, it can be disregarded (no need to store it)
- learning at **frame rate**:
  - learning is fast enough such that it can be performed in one perception cycle

Converting Offline into Online

- Often, we can start from a given **offline** algorithm and modify it so that it is an **online** algorithm
- Usually there is a **trade-off** between performance and speed (online vs offline)
- We show two examples to build online methods from existing offline algorithms:
  - boosting (GradientBoost)
  - bagging (Random Forest)

Example: Gradient Boost

- Initialize

$$f_0(\mathbf{x}) = \arg\min_{\gamma} \sum_{i=1}^{N} L(t_i, \phi(\mathbf{x}_i, \gamma))$$

- for $m = 1, \ldots, M$
  - Compute the gradient residual

$$r_{im} = -\left[\frac{\partial L(t_i, f(\mathbf{x}_i))}{\partial f(\mathbf{x}_i)}\right]_{f(\mathbf{x}_i)=f_{m-1}(\mathbf{x_i})}$$

  - Use the weak learner to compute γ_m that minimizes

$$\sum_{i=1}^{N} (r_{im} - \phi(\mathbf{x}_i; \gamma_m))^2$$

  - Update

$$f_m(\mathbf{x}) = f_{m-1}(\mathbf{x}) + \nu\phi(\mathbf{x}, \gamma)$$

- Return $f(\mathbf{x}) = f_M(\mathbf{x})$

Online Gradient Boost

Online: outer loop over data points, inner loop over weak classifiers

- For each point, all weak classifiers are updated

Algorithm 1: Online Multi-class Gradient Boost [Saffari et al. 2010]

Data: training data $(\mathcal{X}, \mathbf{y})$ with C classes
Input: number of weak learners M, loss function ℓ, agreement function a
Output: weak learners $f_1, \ldots, f_M$

```
Initialize(f_1, ..., f_M)
for n = 1, ..., N do
    w_n ← 1
    g_n ← 0
    for m = 1, ..., M do
        f_m ← UpdateWeakLearner(f_m, x_n, y_n, w_n)
        p_nm ← f_m(x_n)
        α_nm ← a(p_nm, y_n)
        g_n ← g_n + α_nm
        w_n ← −∇ℓ(g_n)
    end
end
```

- **Agreement** α between prediction $\mathbf{p}_{nm}$ and ground truth y_n: $a(\mathbf{p}_{nm}, y_n) = p_{nm}^{(y_n)} - 1/C$
- Sample receives low weight if agreement high

Online Random Forest

- As in Online Boosting, the two for loops are switched:
  - outer loop over al data points
  - inner loop over all random trees
- Every tree in the Random Forest grows with new data points
- Each node has to see a minimum number of samples to be split
- Each split has to achieve a minimum information gain
- Details: A. Saffari, M. Godec, T. Pock, C. Leistner, and H.Bischof, “Online multi-class LP-Boost,” in *CVPR*, 2010.

Online Learning for Active Learning

- To reduce the required training data, the learner can **select** the data it needs to learn from.
- This is called **active** learning
- Major advantages:
  - only those samples where classification is hard are used for re-training
  - humans only need to give ground truth labels for a smaller set of samples
- In principle, active learning can be used with offline and online learning, but online makes more sense

Active Learning (Rep.)

Training data

New Data

Supervisor

Optimization

function

$f(\mathbf{x})$

Prediction

Uncertainty,

Labels

Label Query

New training

data

Extend training

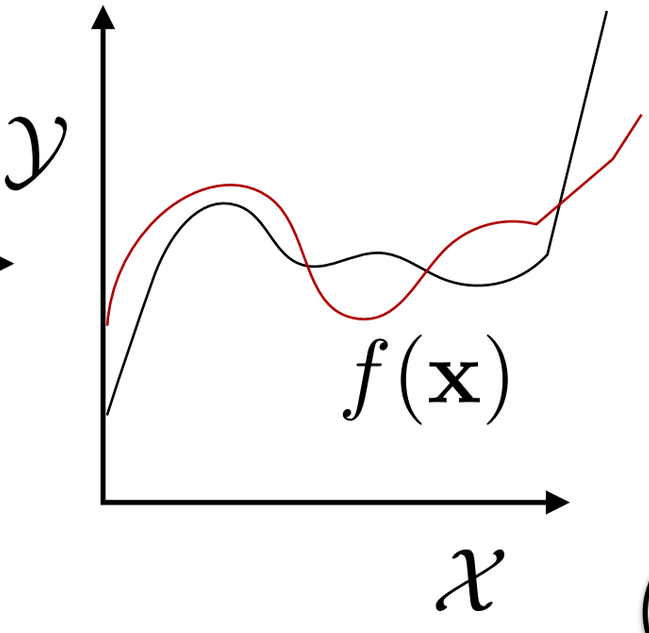
data

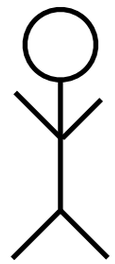
training

data

“Laptop”

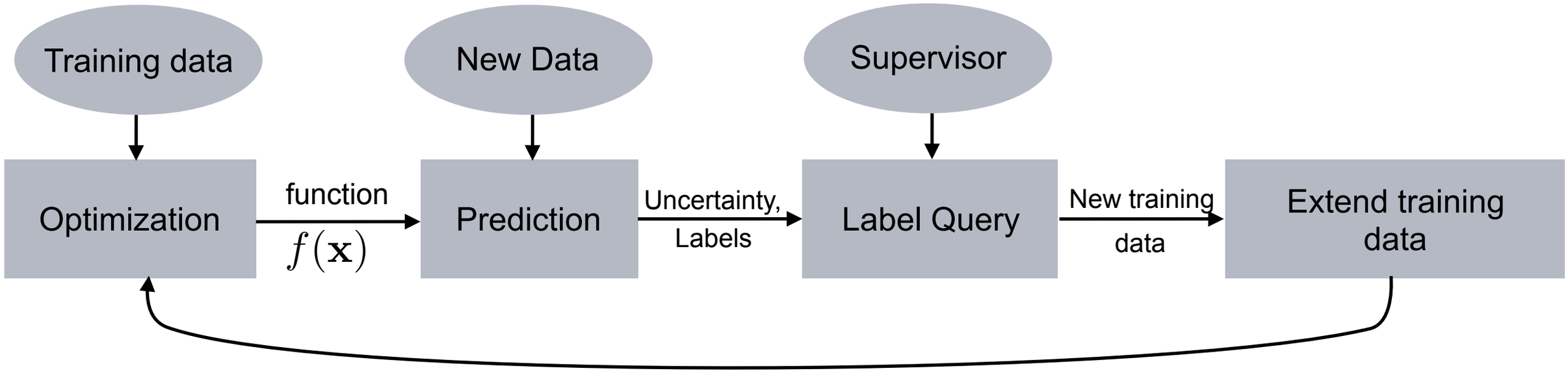
“Cup”





“Present”

Active Learning (Rep.)



Major Benefits:

- Adapts to new situations
- Requires less training samples

Uncertainty Estimates (Rep.)

- A key step in active learning is the **selection** step
- It requires a good estimate of **uncertainties**
- Examples for classification algorithms to compute uncertainties:
  - Support Vector Machine (“Platt scaling”)
  - Gaussian Process Classifier (“predictive variance”)
  - Tree-based classifiers (entropy in leaf nodes)

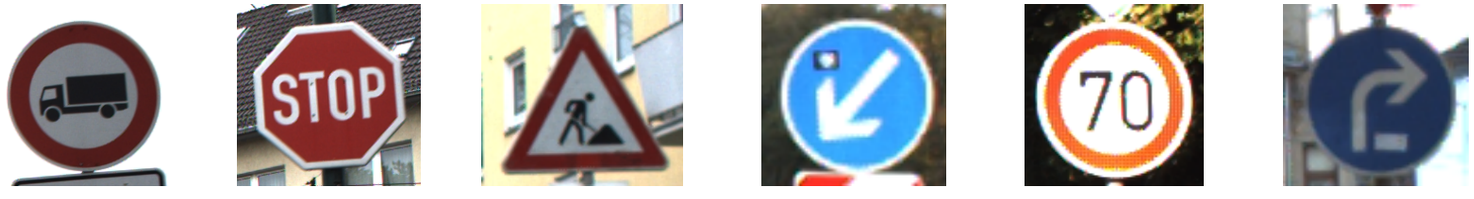
But: It is important how uncertainty correlates to incorrect classifications!

Traffic Sign Classification (Offline)

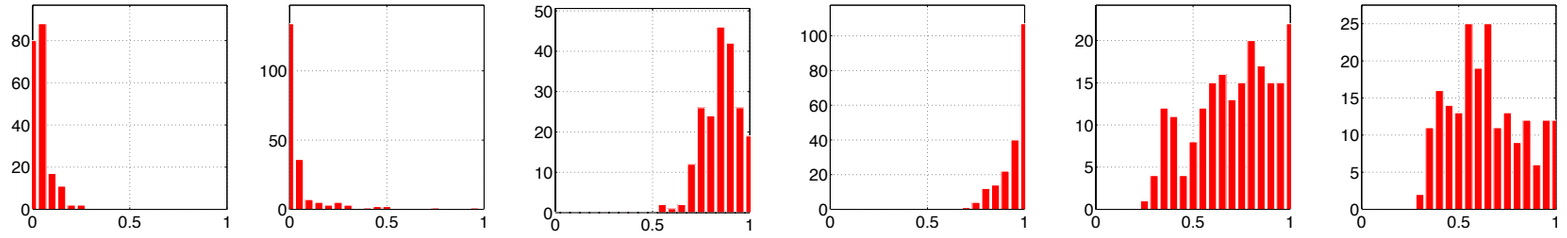
uncertainty histograms

Trained classes

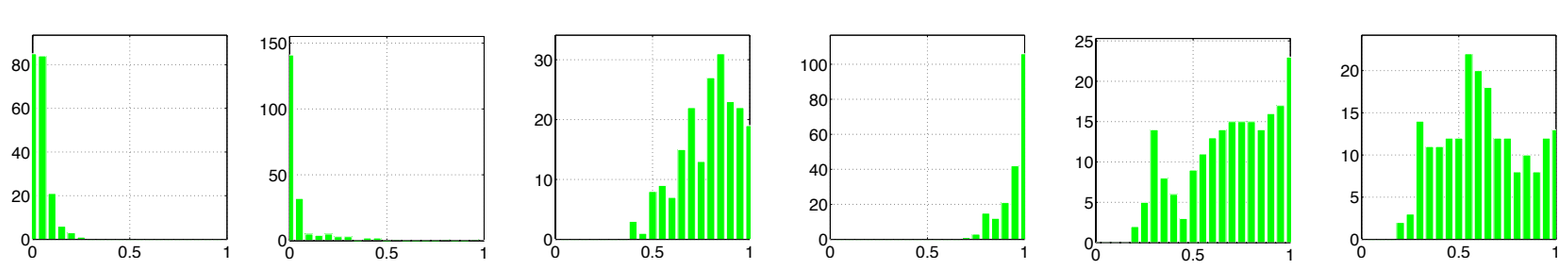
Unseen classes



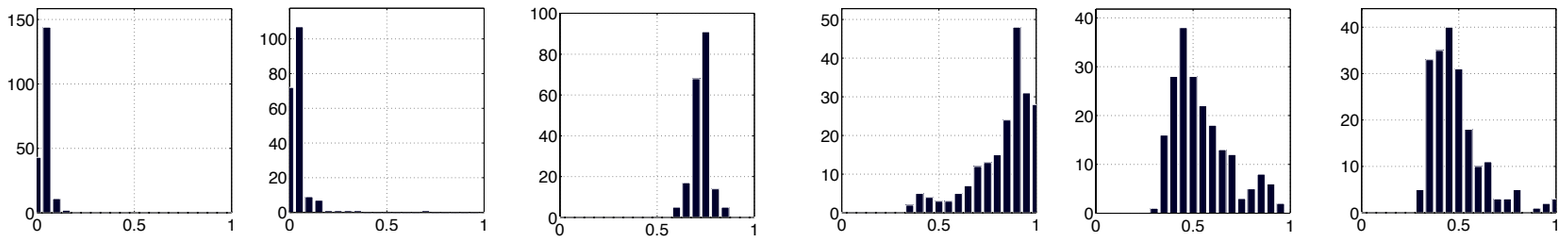
Non-linear GPC



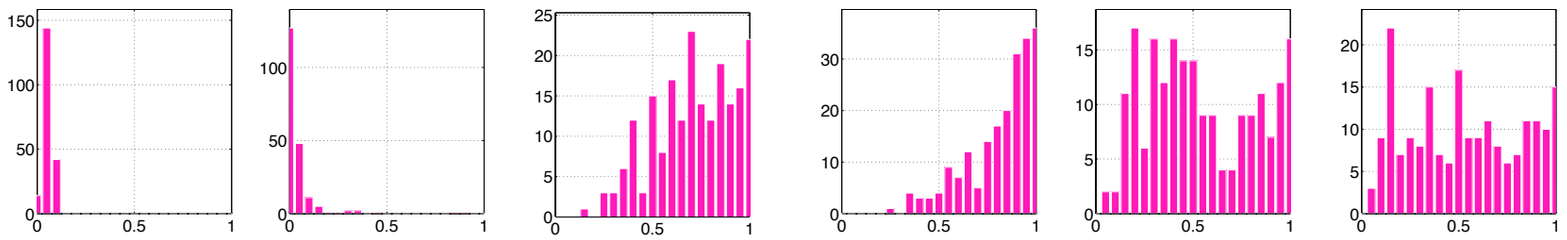
Linear GPC



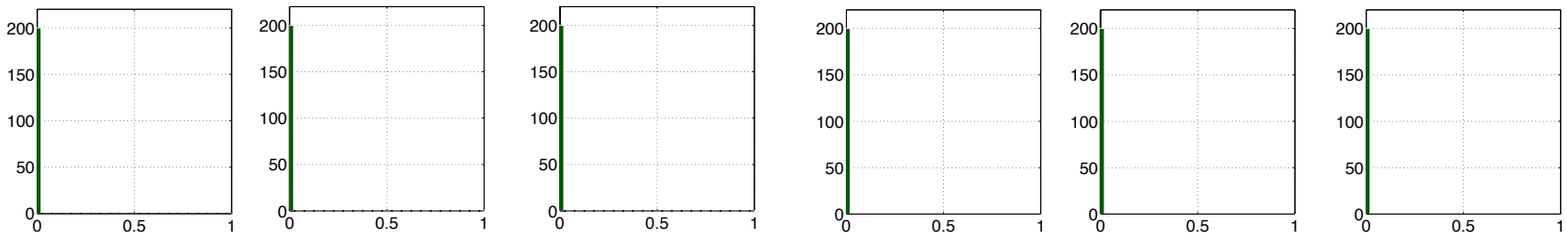
Non-linear SVM



Linear SVM



Logit Boost



The GP is less *overconfident* than the SVM!

[Grimmett, Paul, Triebel, Posner ICRA 2013]

Over- and Underconfidence

Definition 1: The **overconfidence** of a classifier is the mean of all *confidence* values that correspond to *incorrect* classifications.

Definition 2: The **underconfidence** of a classifier is the mean of all *uncertainty* values that correspond to *correct* classifications.

Why is the GPC Less Overconfident?

1. A GP is a distribution over **functions**: $f(x) \sim \mathcal{GP}(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}^{'}))$

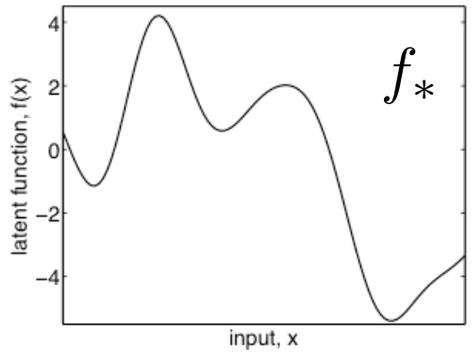
Mean Function — Covariance Function

2. New function values f_* can be predicted by **conditioning** on known values:

$$p(f_* \mid X, \mathbf{f}, \mathbf{x}_*) = \mathcal{N}(\mathbf{k}_*^T K^{-1} \mathbf{f}, k(\mathbf{x}_*, \mathbf{x}_*) - \mathbf{k}_*^T K^{-1} \mathbf{k}_*)$$

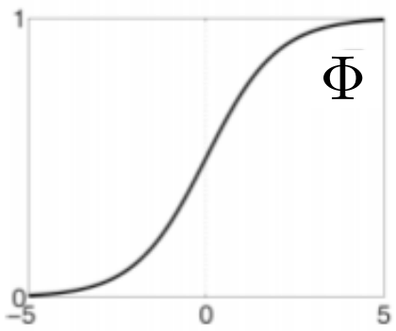
3. In the presence of noise we **marginalize** over the latent functions:

$$p(f_* \mid X, \mathbf{y}, \mathbf{x}_*) = \int p(f_* \mid X, \mathbf{x}_*, \mathbf{f}) p(\mathbf{f} \mid X, \mathbf{y}) d\mathbf{f}$$



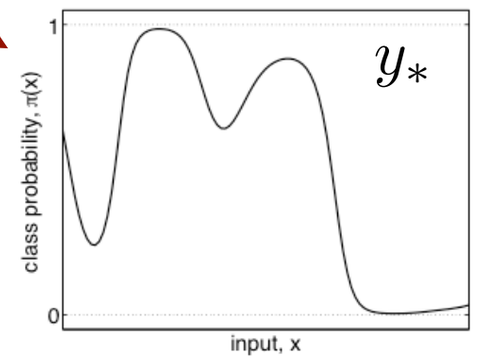
4. For classification, we squash the latent function through a **sigmoid**:

$$\Phi(z) = \int_{-\infty}^{z} \mathcal{N}(x \mid 0, 1) dx$$



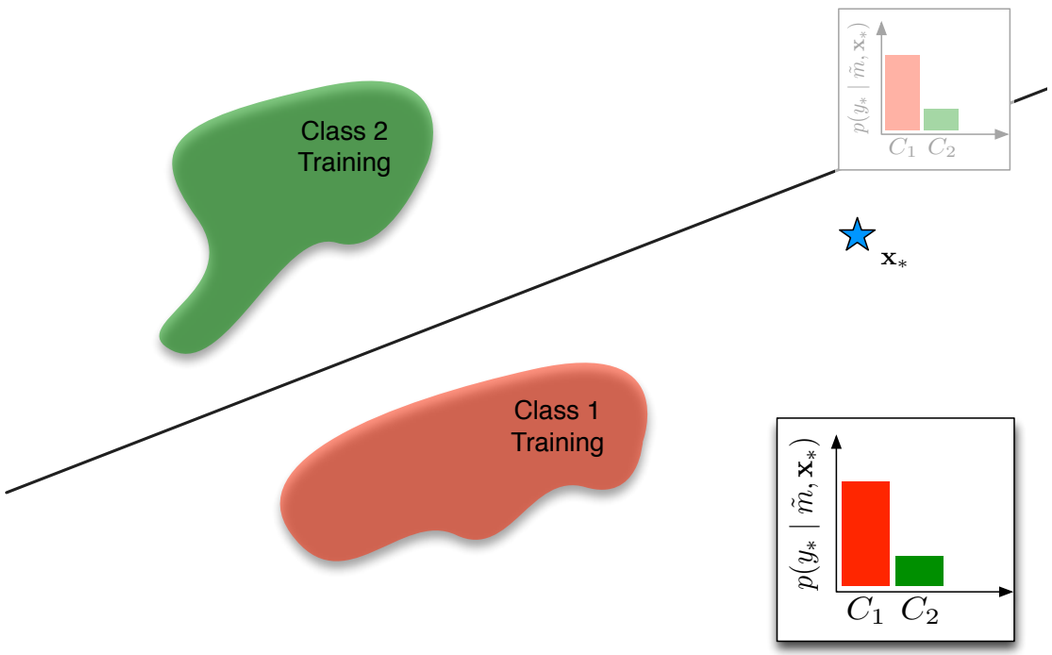
5. We get the predictive distribution via **marginalization**:

$$p(y_* = +1 | X, \mathbf{y}, \mathbf{x}_*) = \int \Phi(f_*) p(f_* | X, \mathbf{y}, \mathbf{x}_*) df_*$$



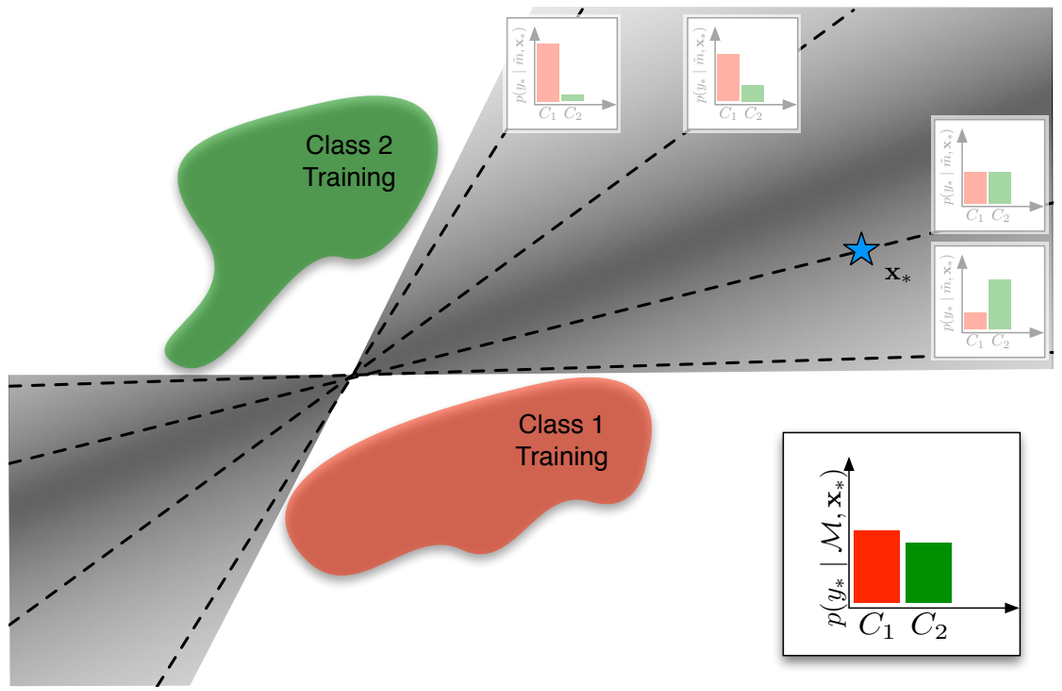
from: Rasmussen and Williams, (2006)

Why is the GPC Less Overconfident?



single model (SVM)

$$p(y^* = 1 \mid f(\mathbf{x}^*)) = \frac{1}{1 + \exp(af(\mathbf{x}^*) + b)}$$



marginalization (GPC)

$$p(f_* \mid X, \mathbf{y}, \mathbf{x}_*) = \int p(f_* \mid X, \mathbf{x}_*, \mathbf{f}) p(\mathbf{f} \mid X, \mathbf{y}) d\mathbf{f}$$

- However: The GPC is very expensive
- Sparse versions of the GP exist, e.g. the Informative Vector Machine (IVM)

[Grimmett et al. ICRA 2013]

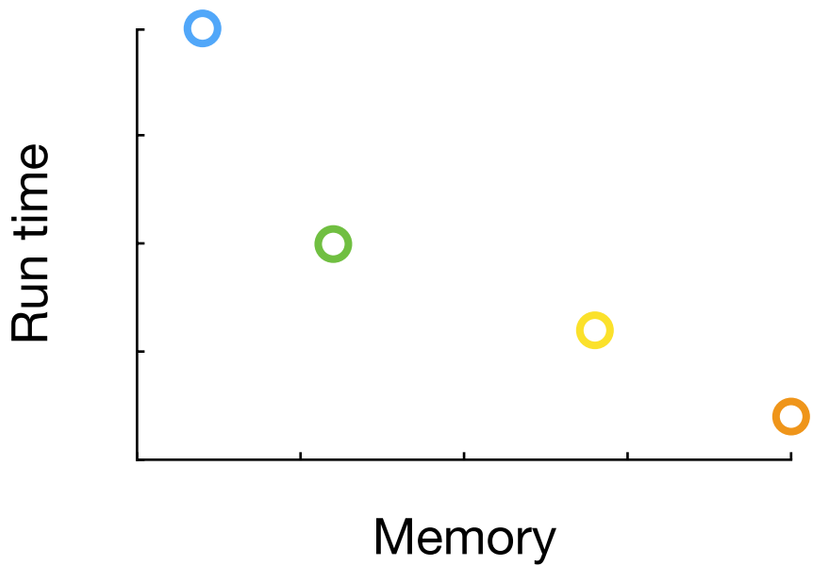
The 3 Criteria of Learning Algorithms

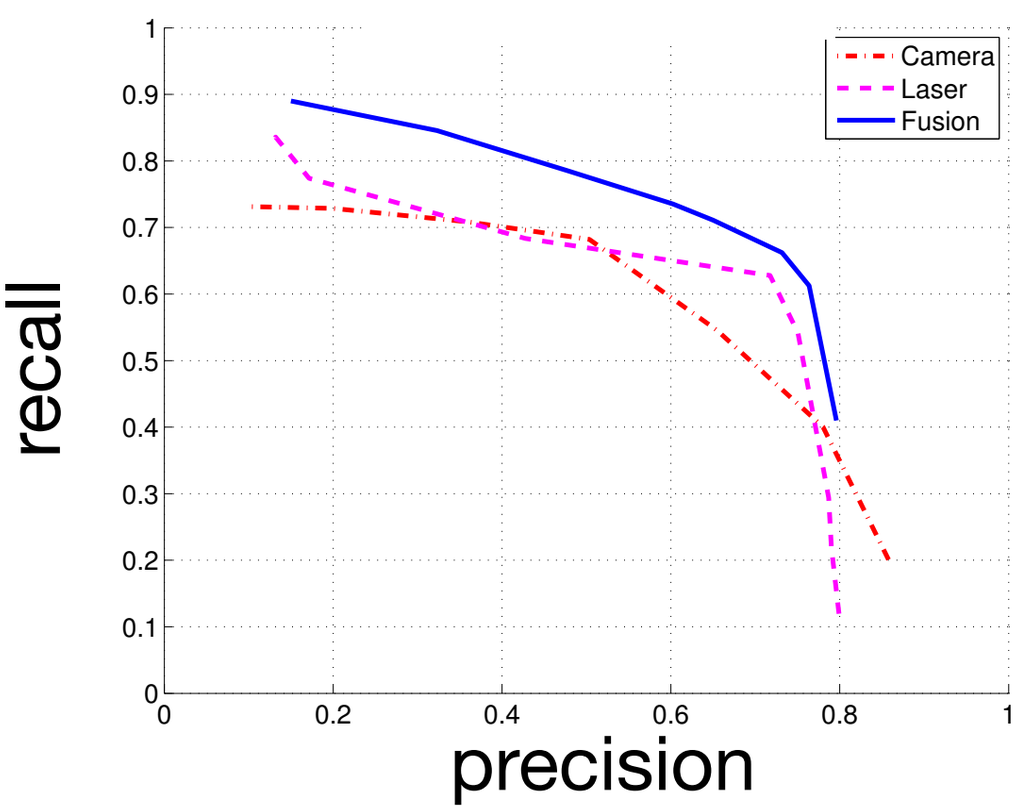
Efficiency

- memory
- run time

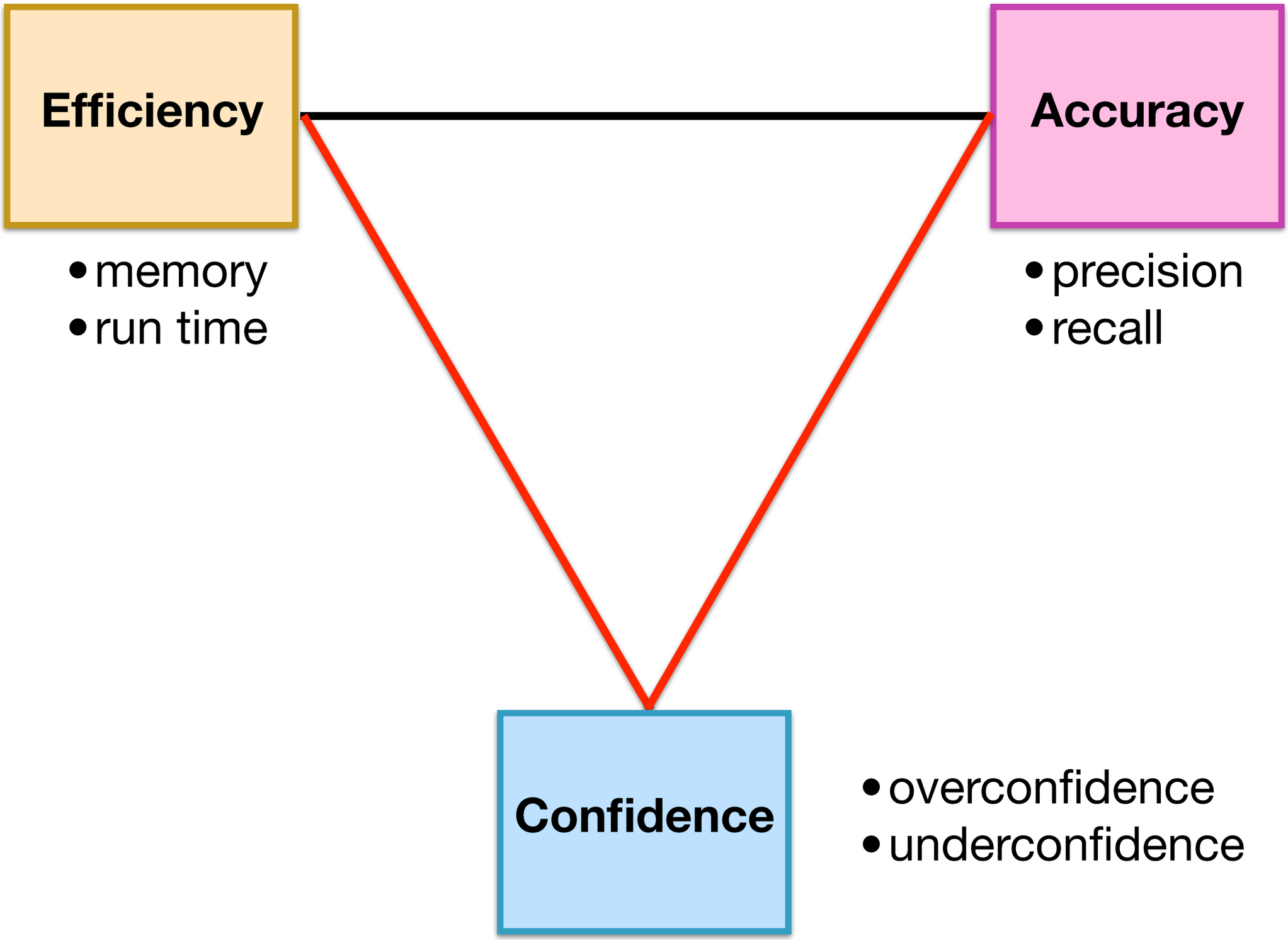
Accuracy

- precision
- recall

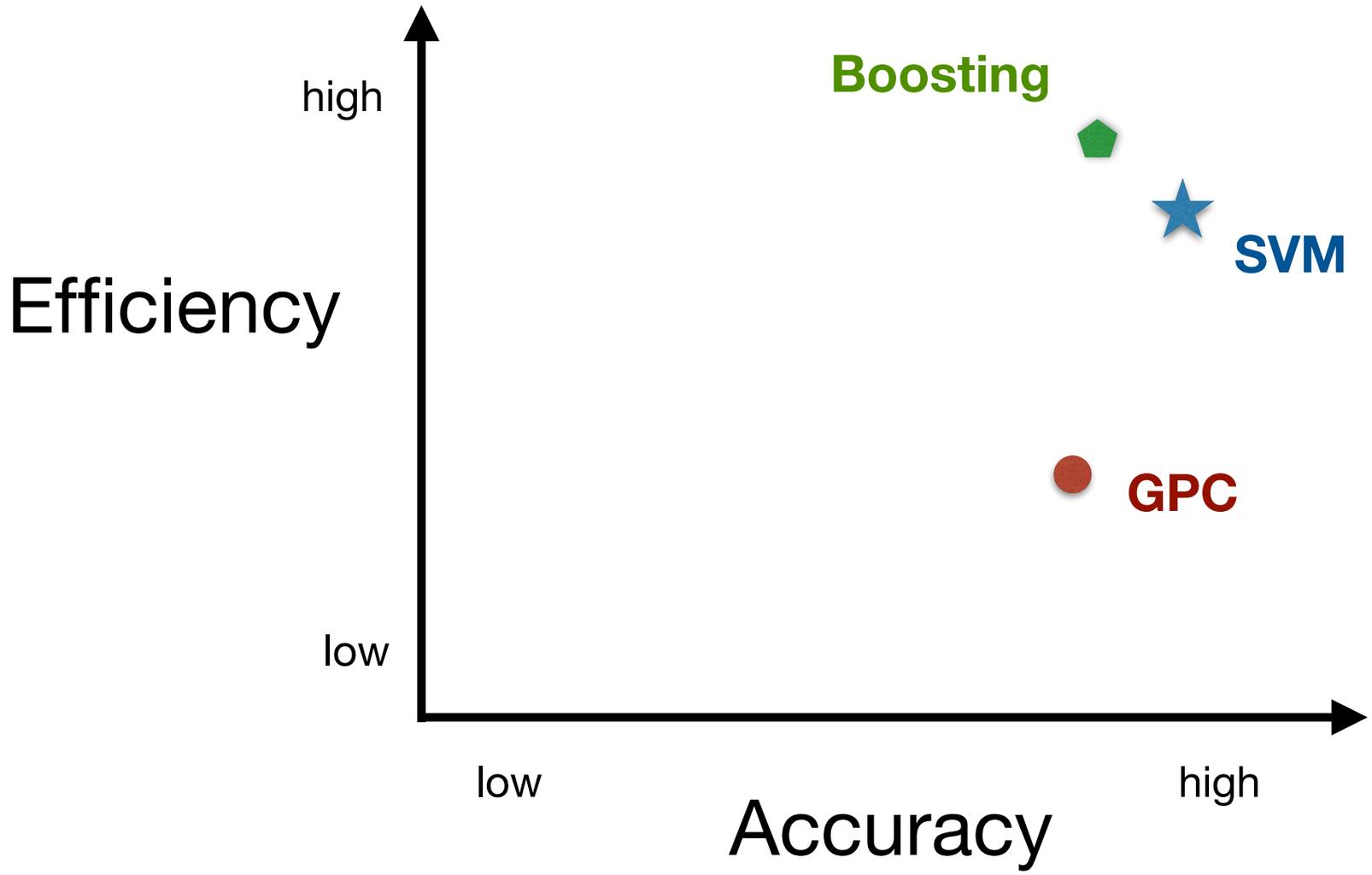




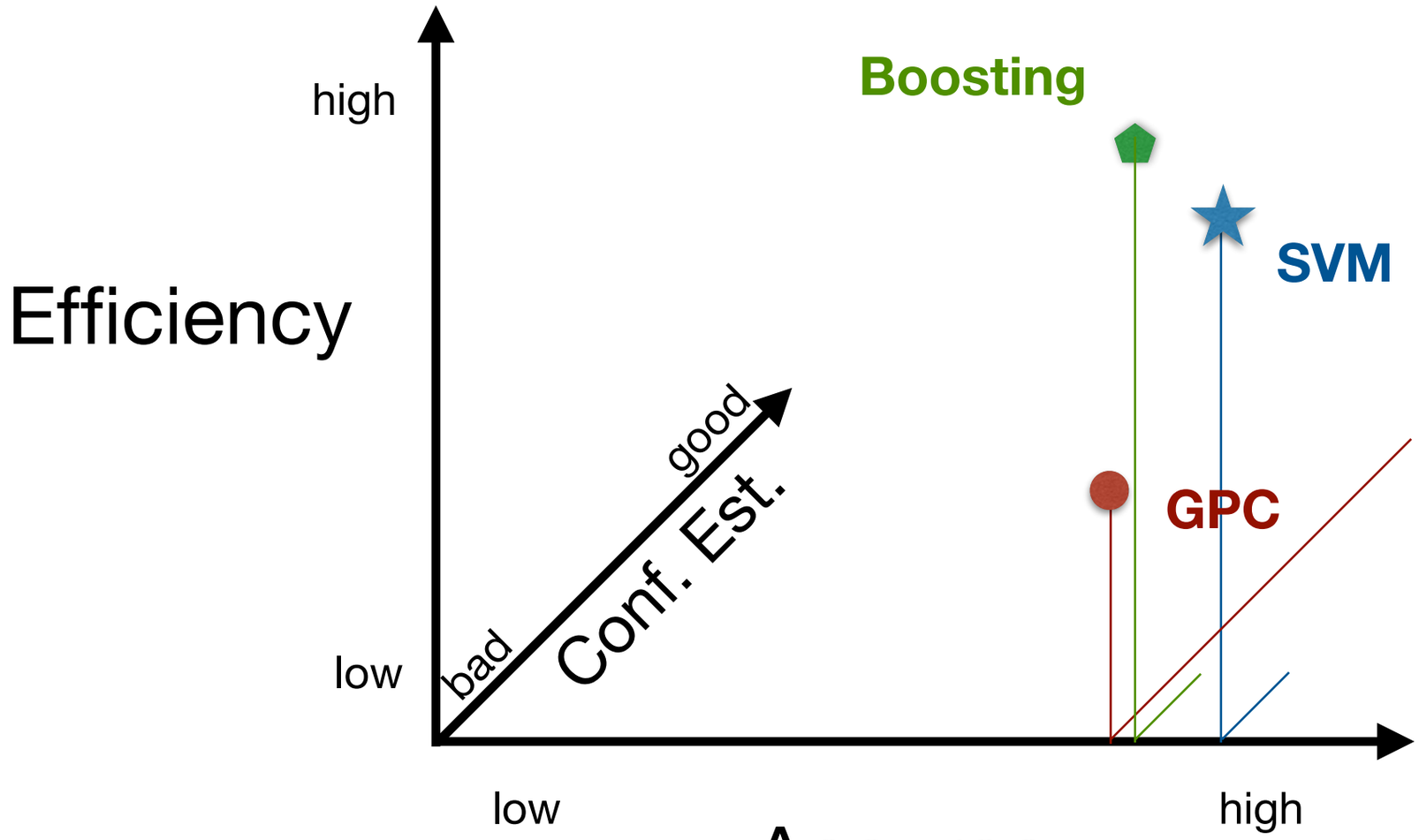
The 3 Criteria of Learning Algorithms



Which Classifier Should We Choose?



Which Classifier Should We Choose?



- SVMs are accurate, but (more) overconfident
- GPCs are less overconfident, but very inefficient
- Boosting is very efficient, but also overconfident

Can We Reduce Over-(Under-)Confidence?

- Start with a very efficient multi-class boosting classifier
- Modify it so that it reduces overconfidence
- Use also the **confidence** to weight training samples

Result ("Confidence Boosting"):

- wrong, certain classified samples receive **higher weight**
- overconfidence reduced in every learning round
- overall classifier is **less overconfident**

  ⇒ better for Active Learning

Note: This is the alternative approach to making a non-overconfident classifier efficient!

Online Confidence Boosting

Main idea:

Use also the **confidence** to compute the agreement

Intuitively:

- if classification is **correct** ⇒ use the **positive** confidence
- if classification is **wrong** ⇒ use **negative** confidence

Result:

- wrong, certain classified samples receive **higher weight**
- overconfidence reduced in every learning round
- overall classifier is **less overconfident**
  ⇒ better for Active Learning

Online Confidence Boosting

- Only Modification of Online Gradient Boost is the **agreement** function a_c:

$$a_c(\mathbf{p}_{nm}, y_n) = (-1)^{\xi} \left(1 - \frac{h(\mathbf{p}_{nm})}{(C-1)^{\xi}}\right)$$

where $$\xi = I(\arg\max_i \mathbf{p}_{nm}^{(i)} \neq y_n).$$

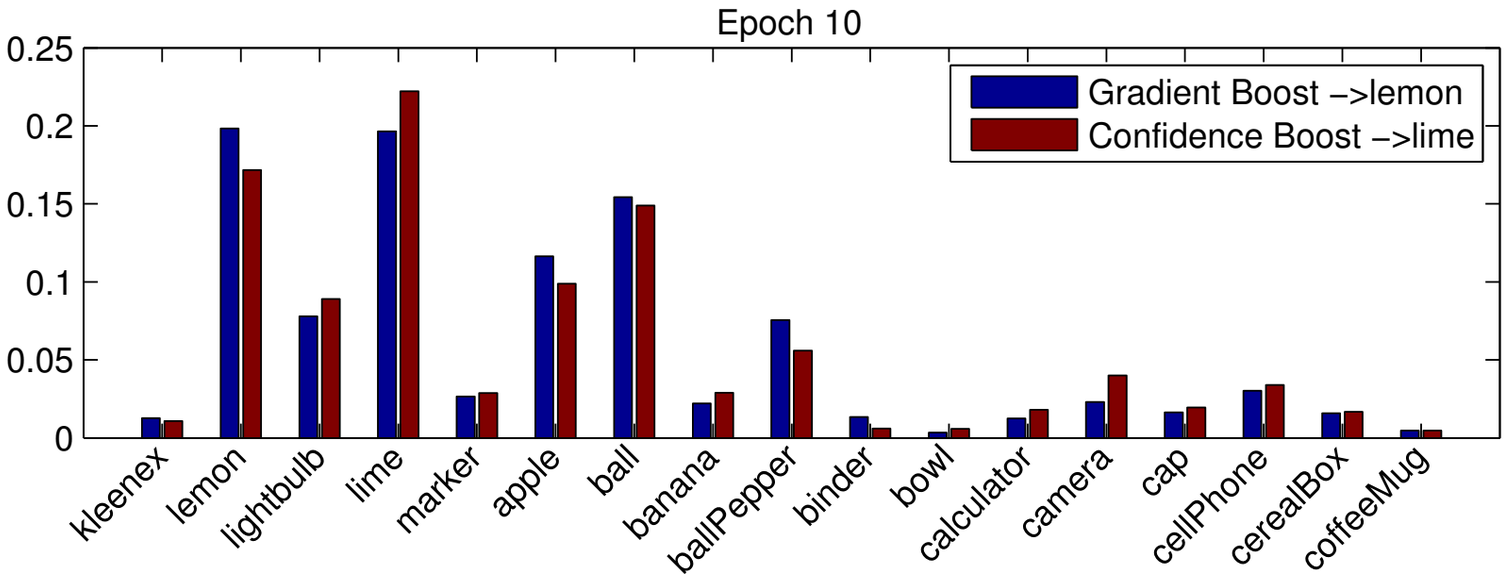
and h is uncertainty, e.g. normalized entropy:

$$h(\mathbf{p}) := -\sum_{i=1}^{C} p_i \log_C(p_i)$$

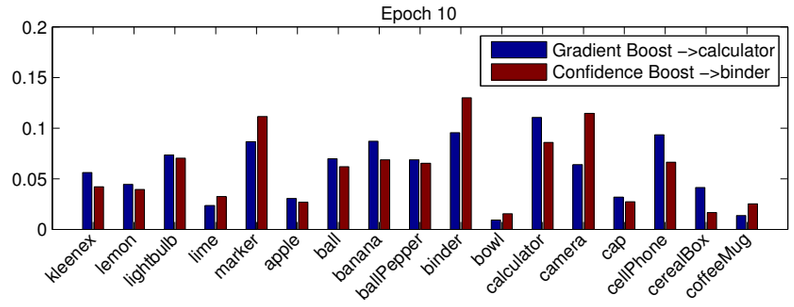
Example: AL for 3D Object Recognition

Qualitative Results:

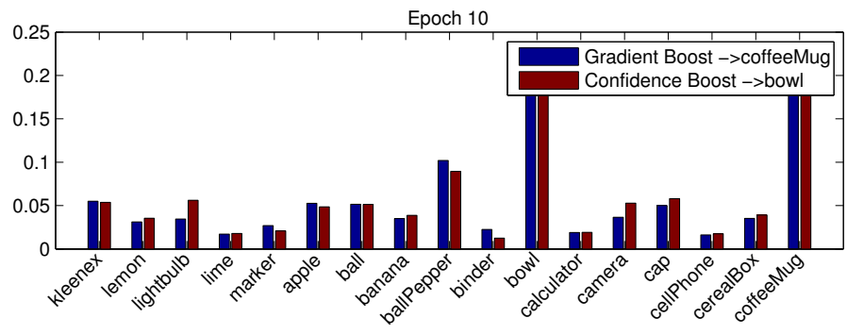
True Label =
lime



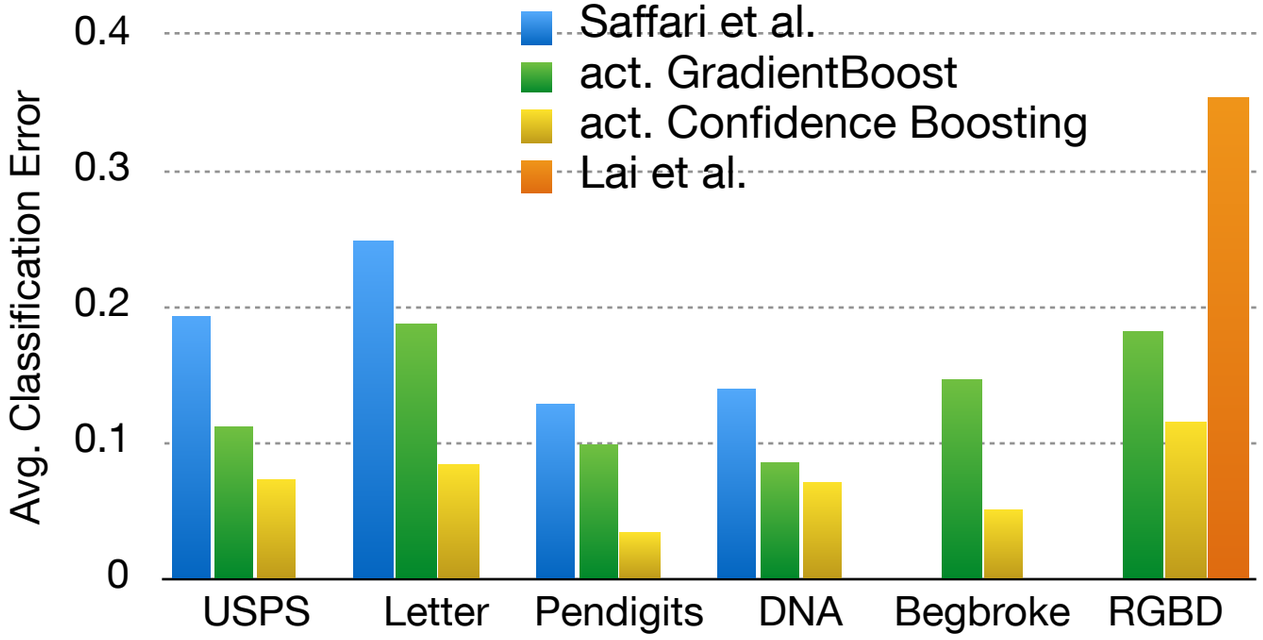
True Label =
binder

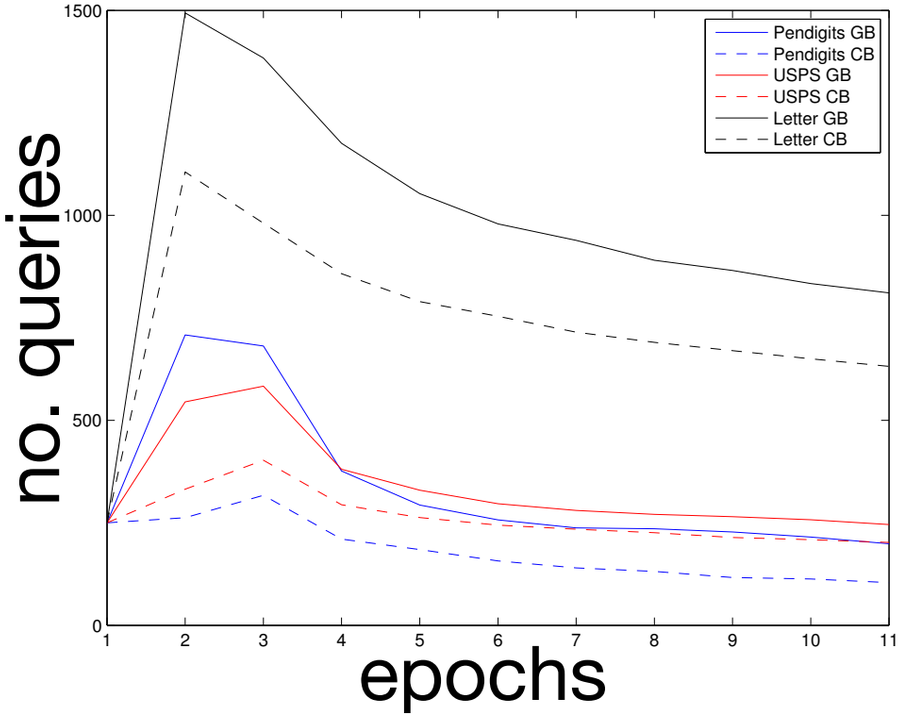


True Label =
bowl



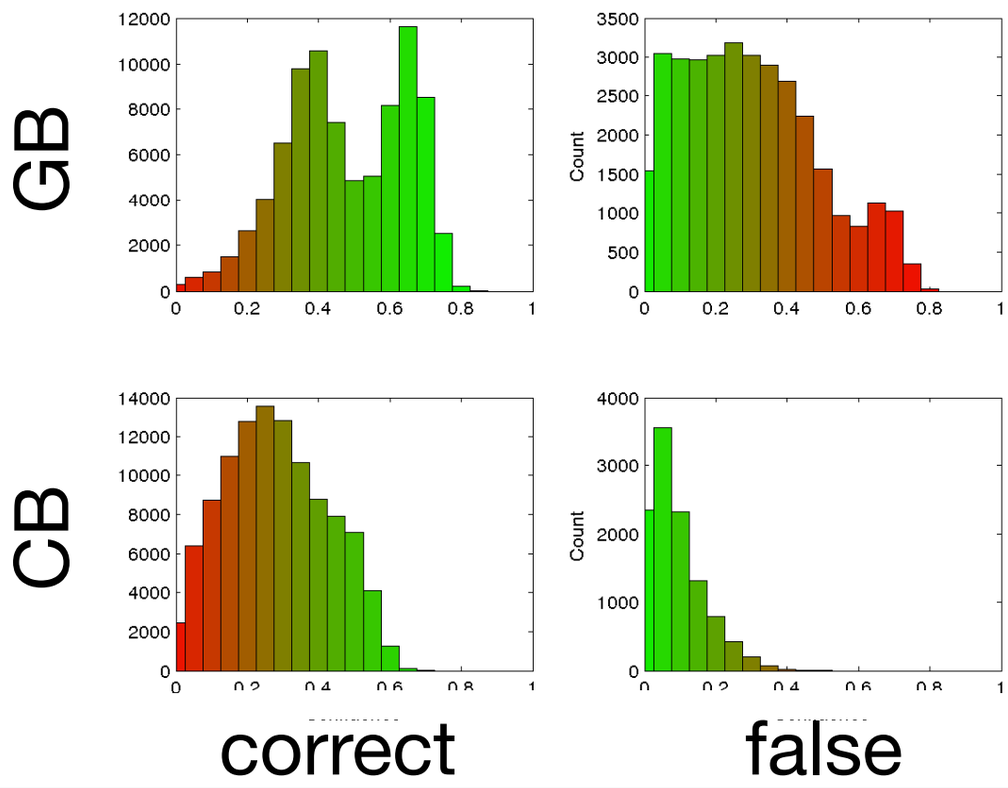
Example: AL for 3D Object Recognition



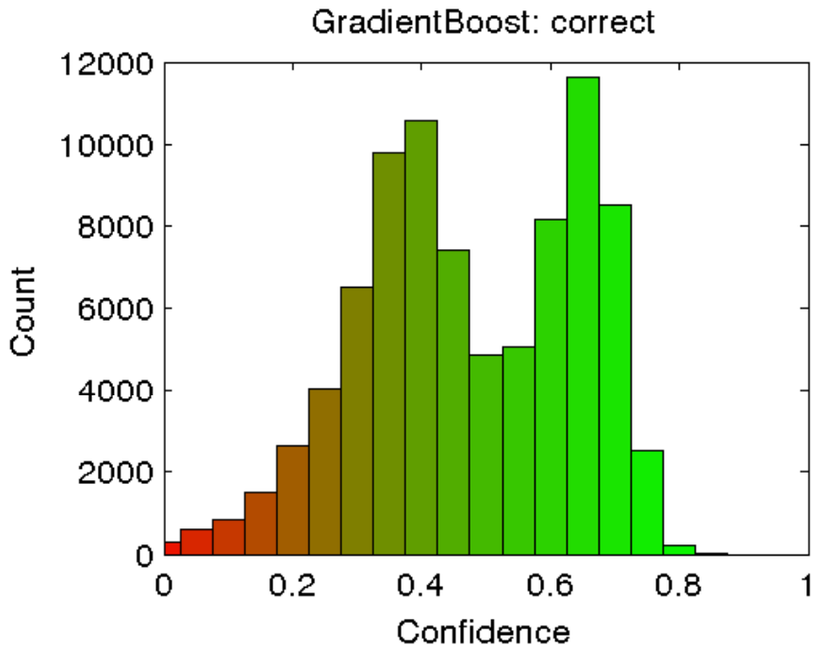


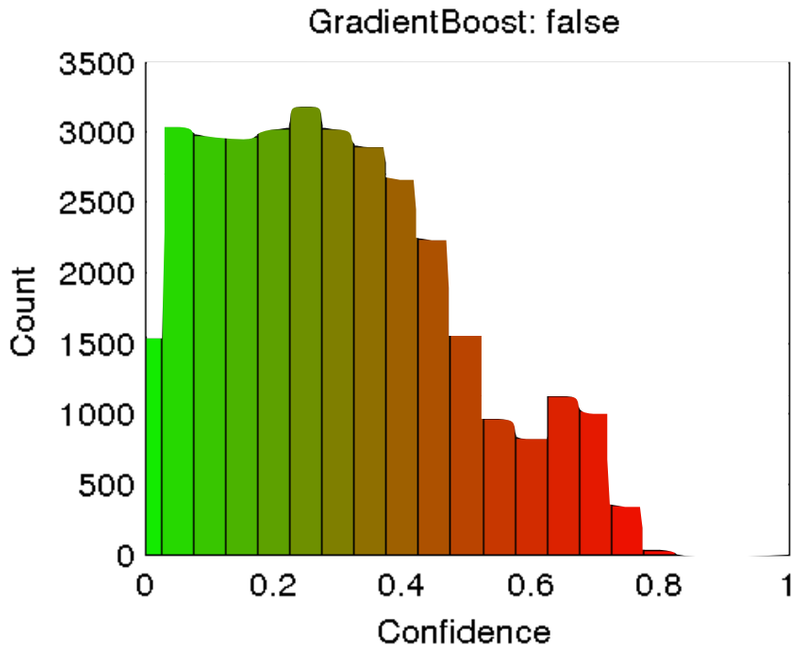
Confidence Boosting

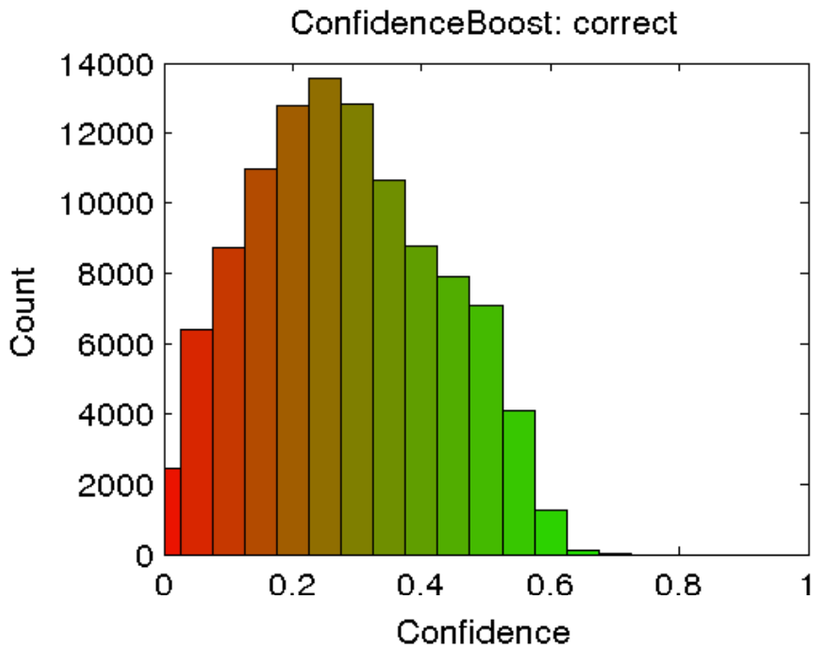
- classifies better than GradientBoost.
- generates fewer label queries.
- is less overconfident.
- is orders of magnitude faster than GPC.

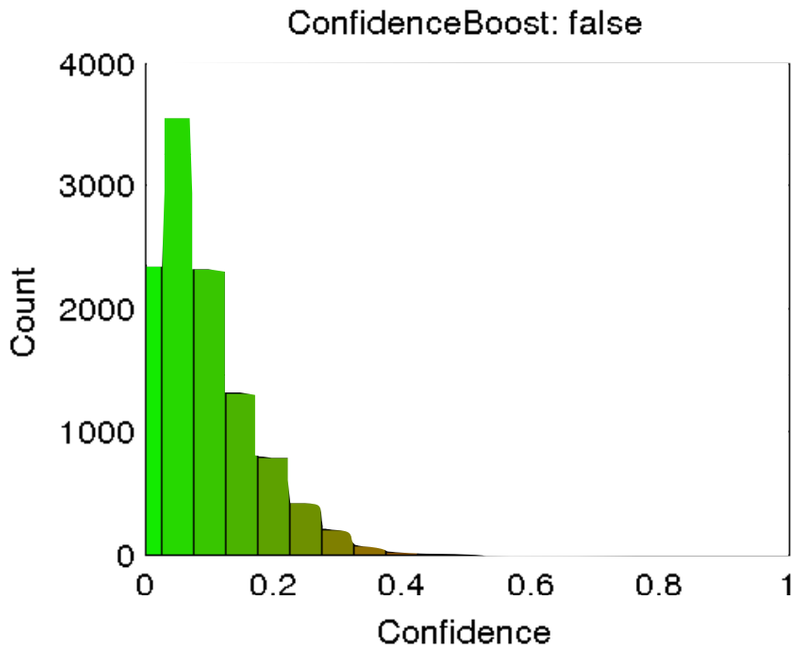


How Can We Find a Good Threshold?

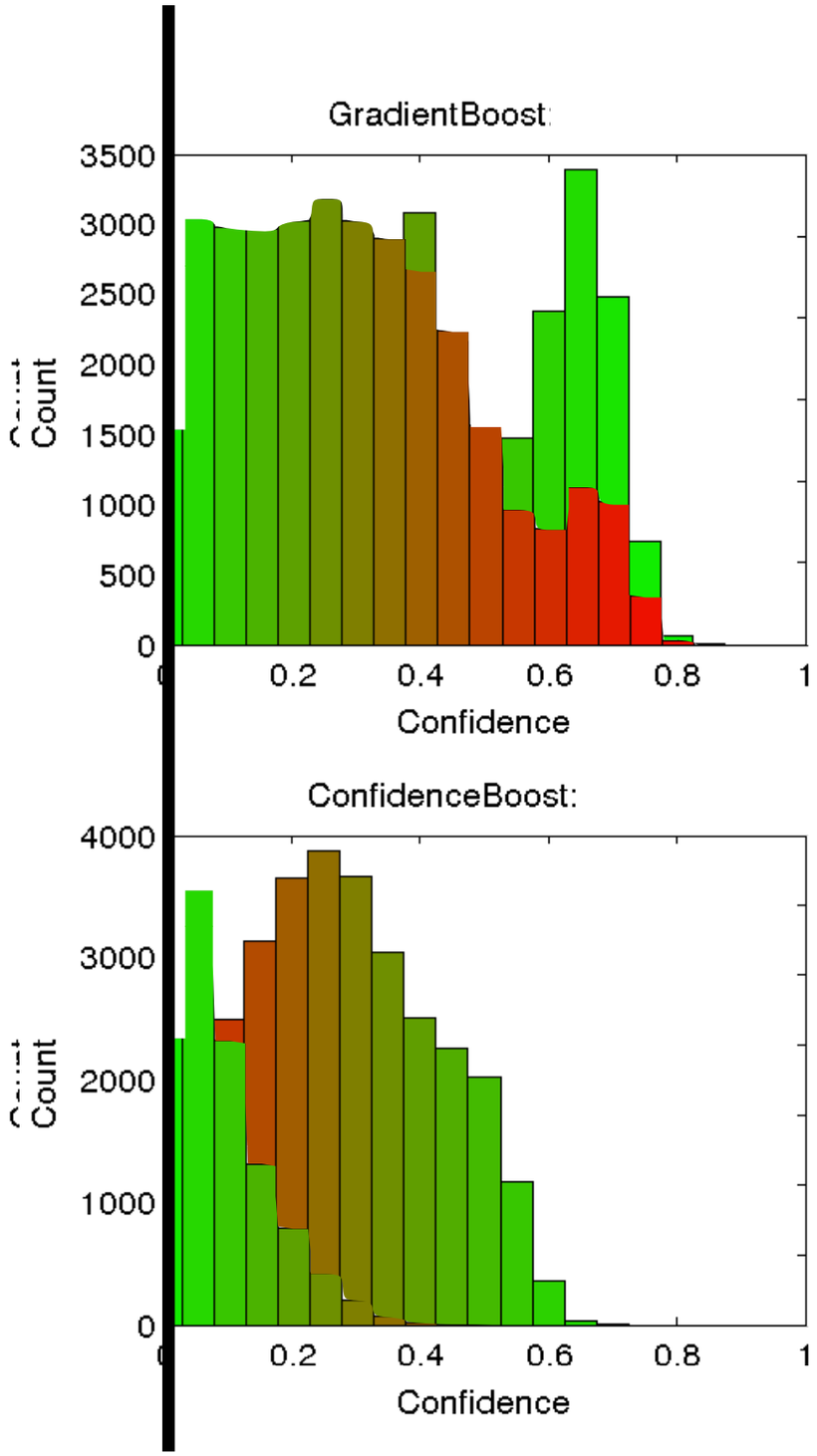


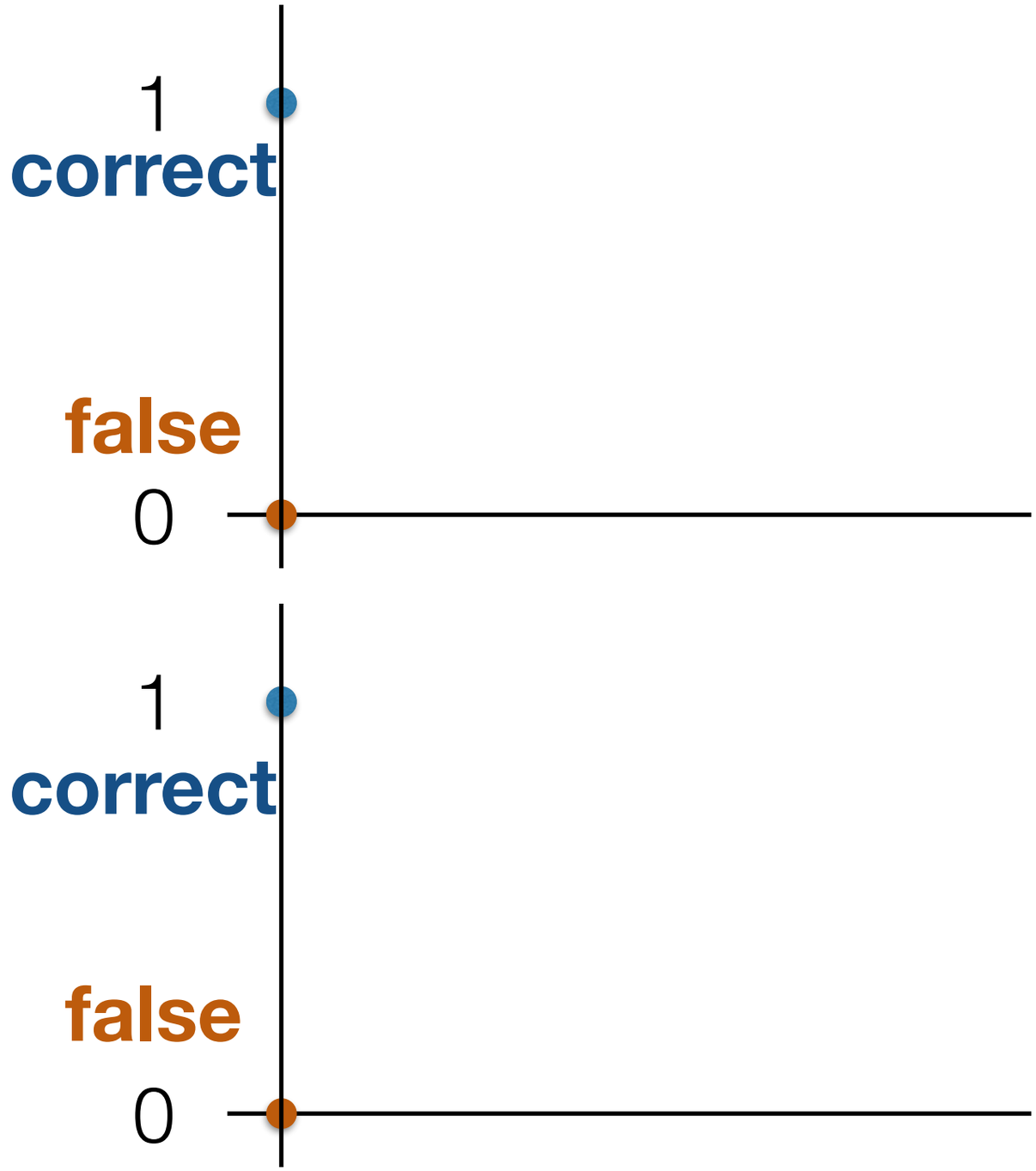




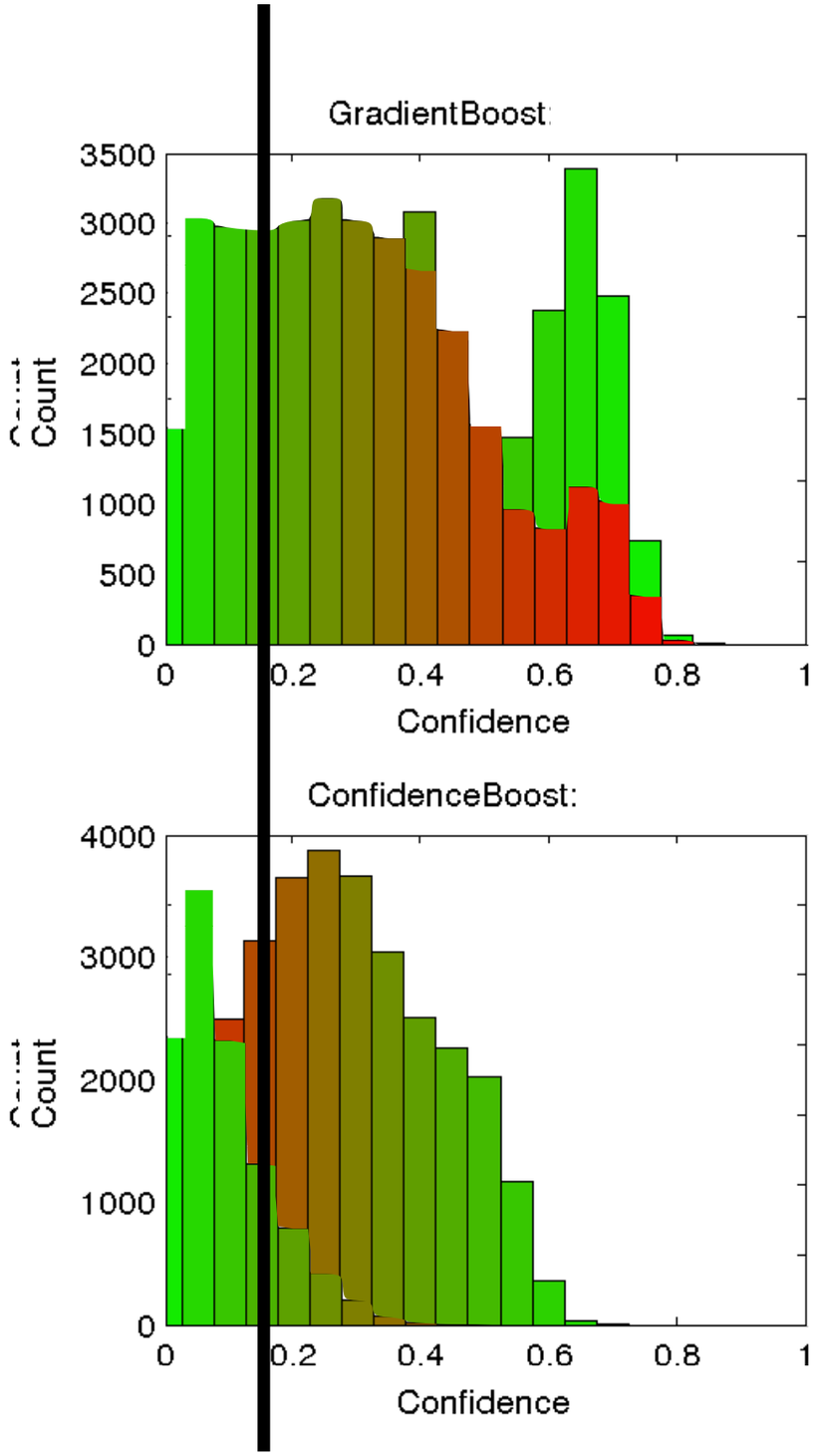


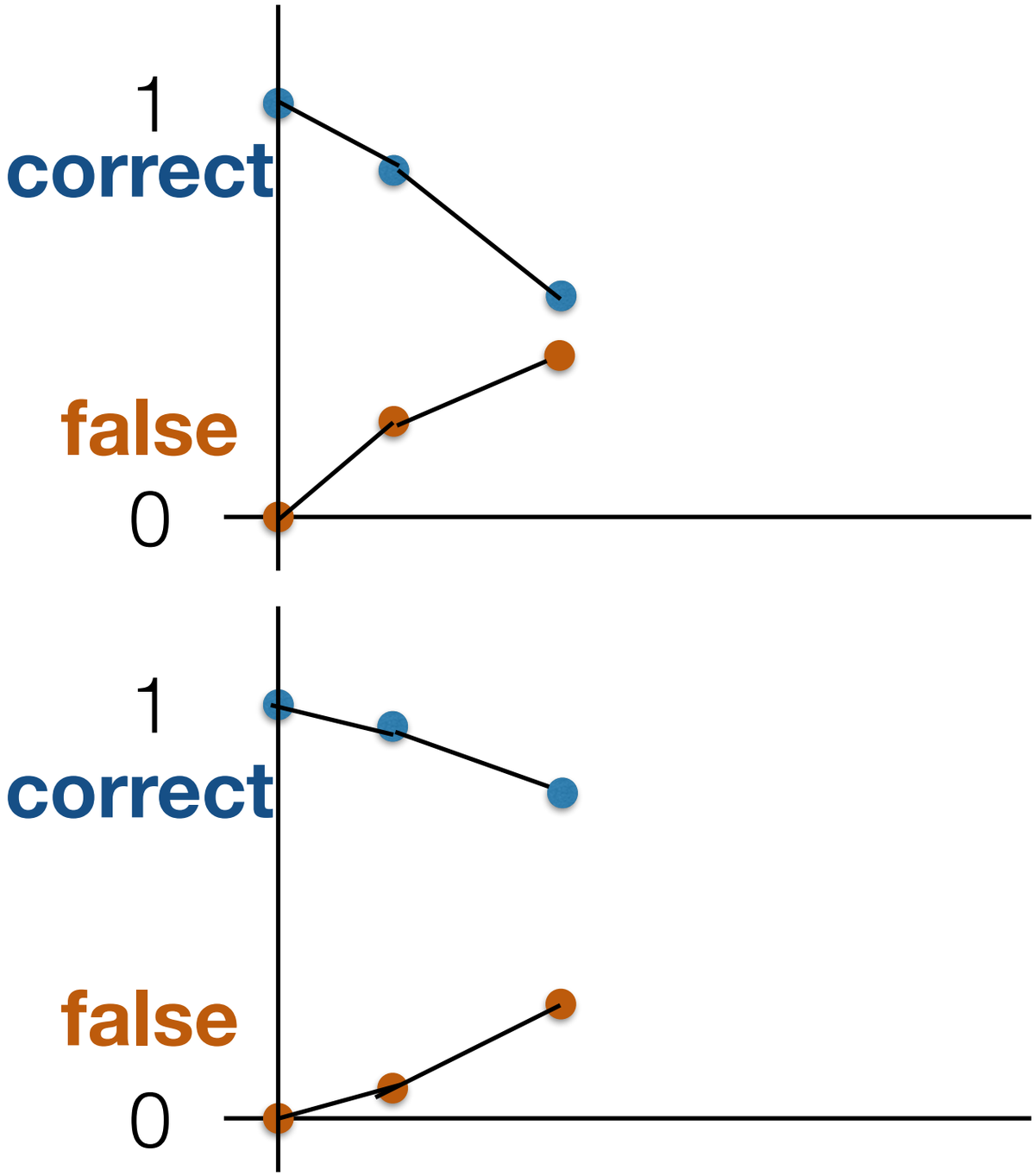
How Can We Find a Good Threshold?



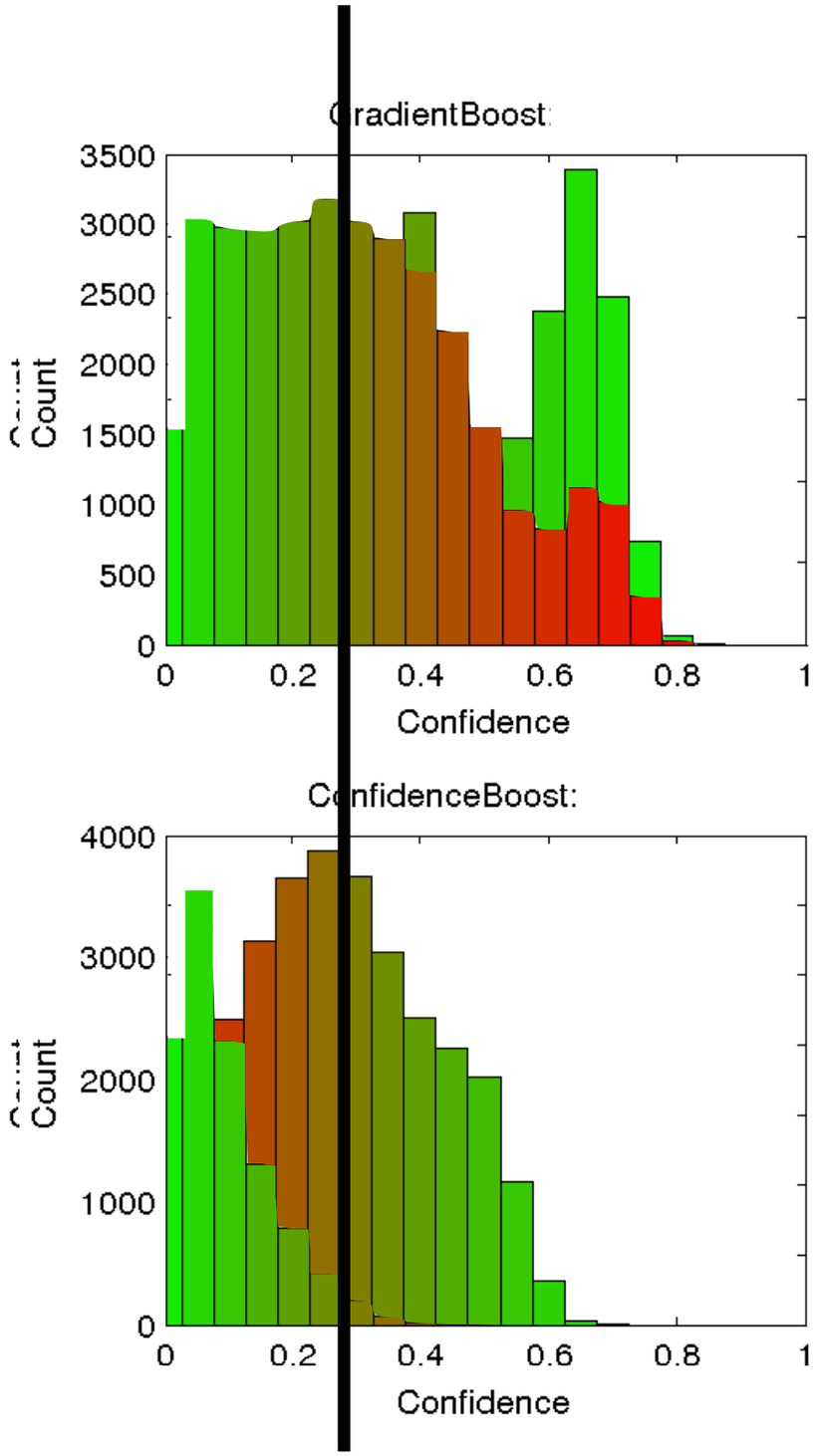


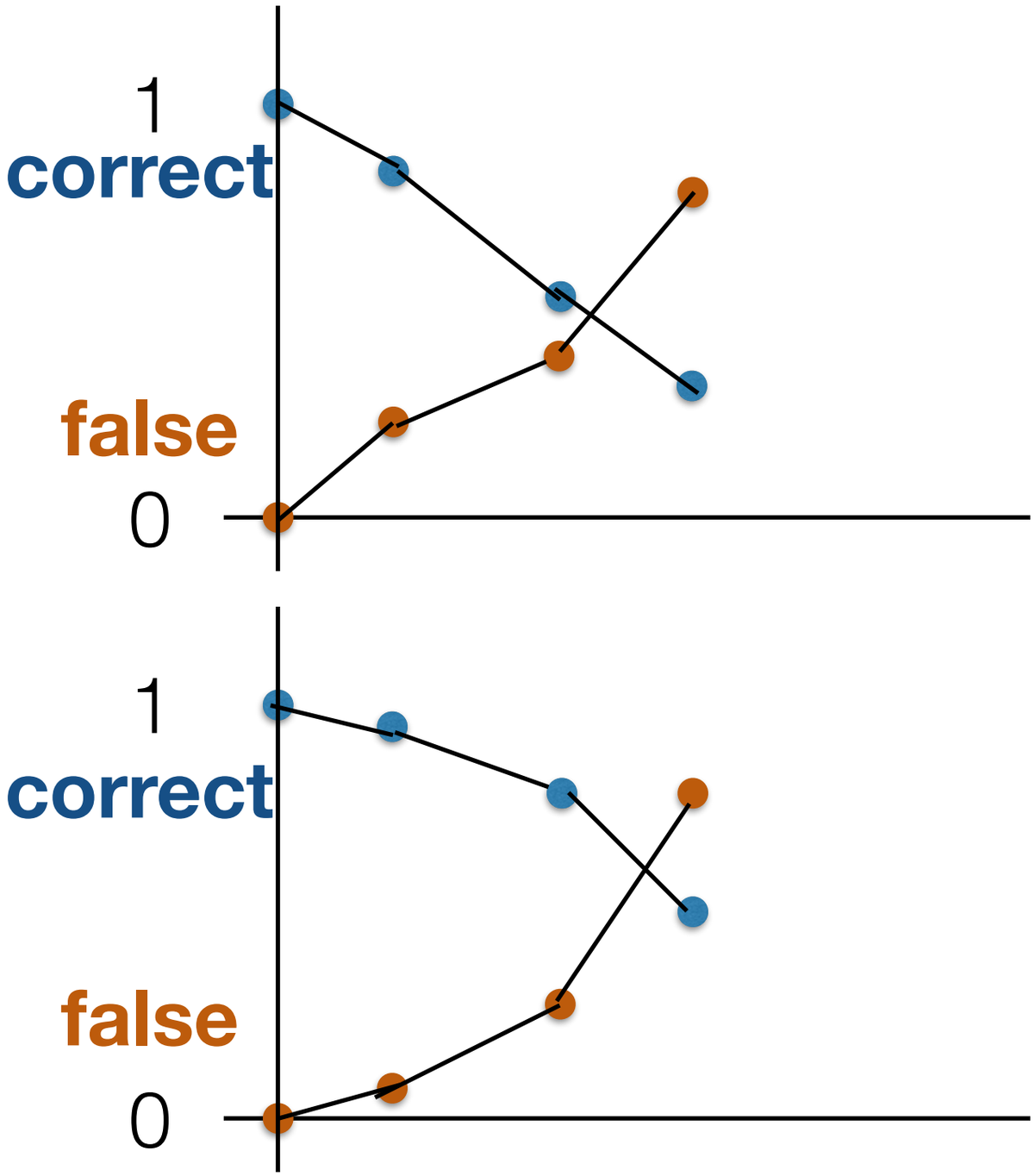
How Can We Find a Good Threshold?



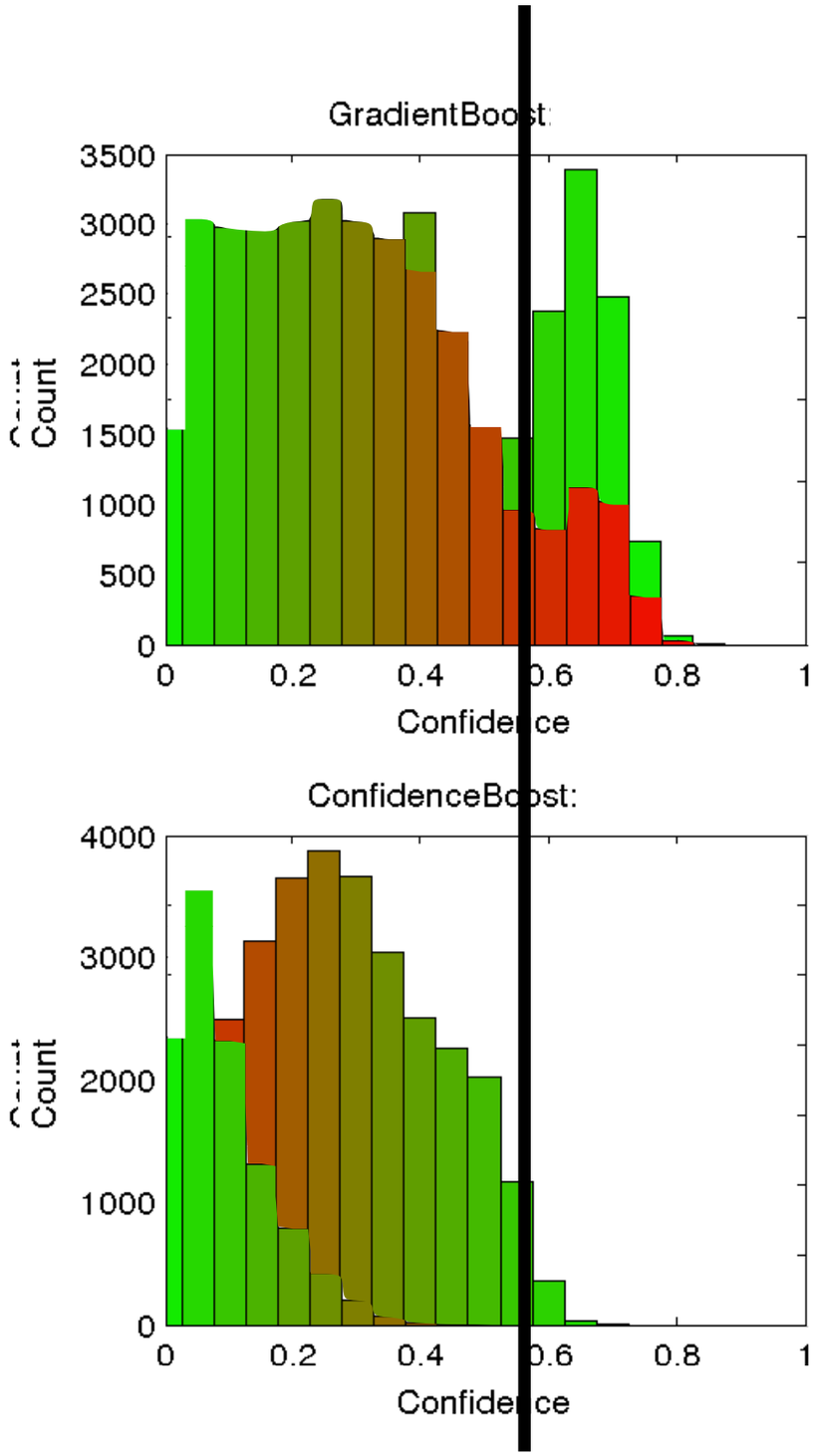


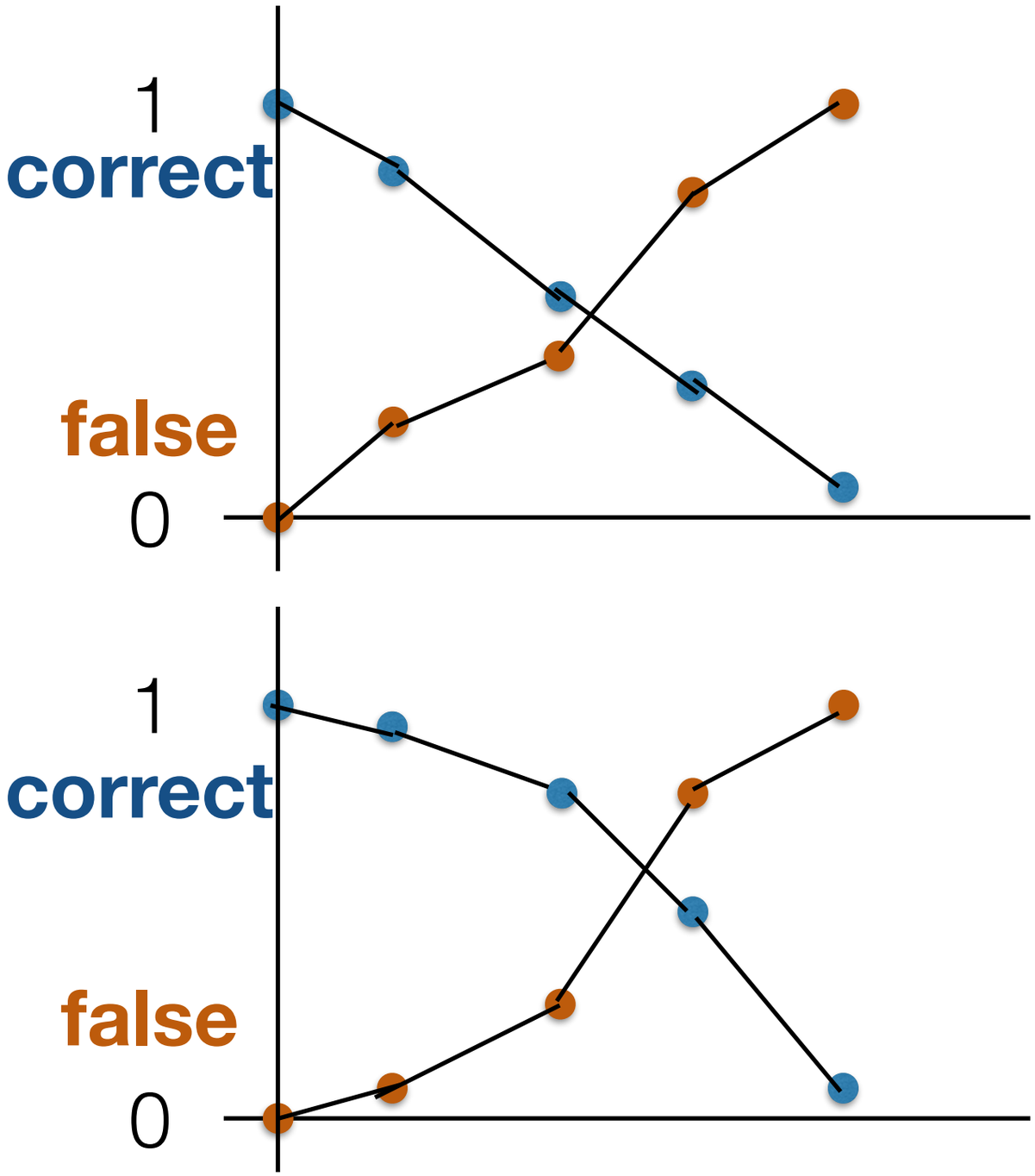
How Can We Find a Good Threshold?



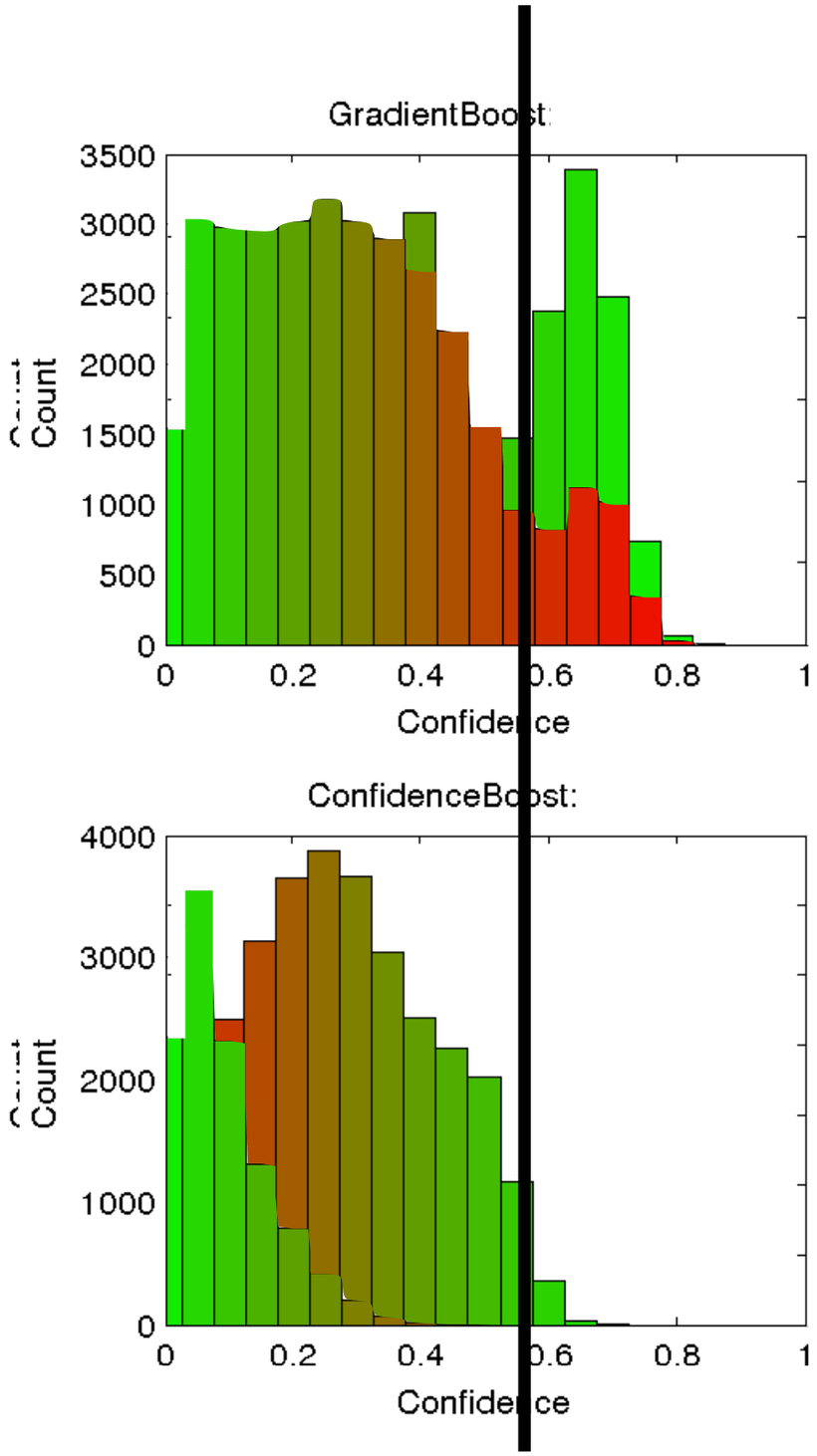


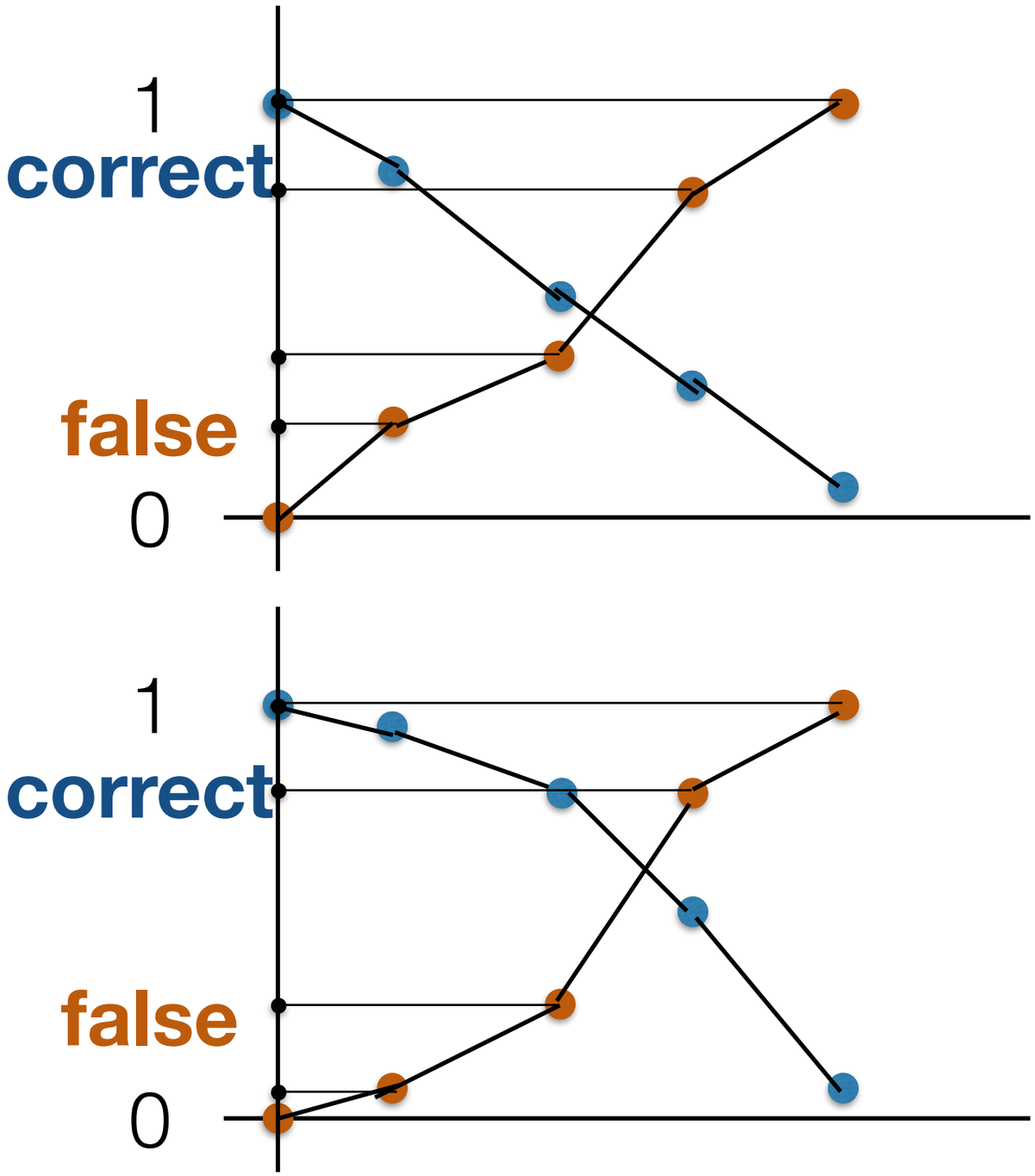
How Can We Find a Good Threshold?



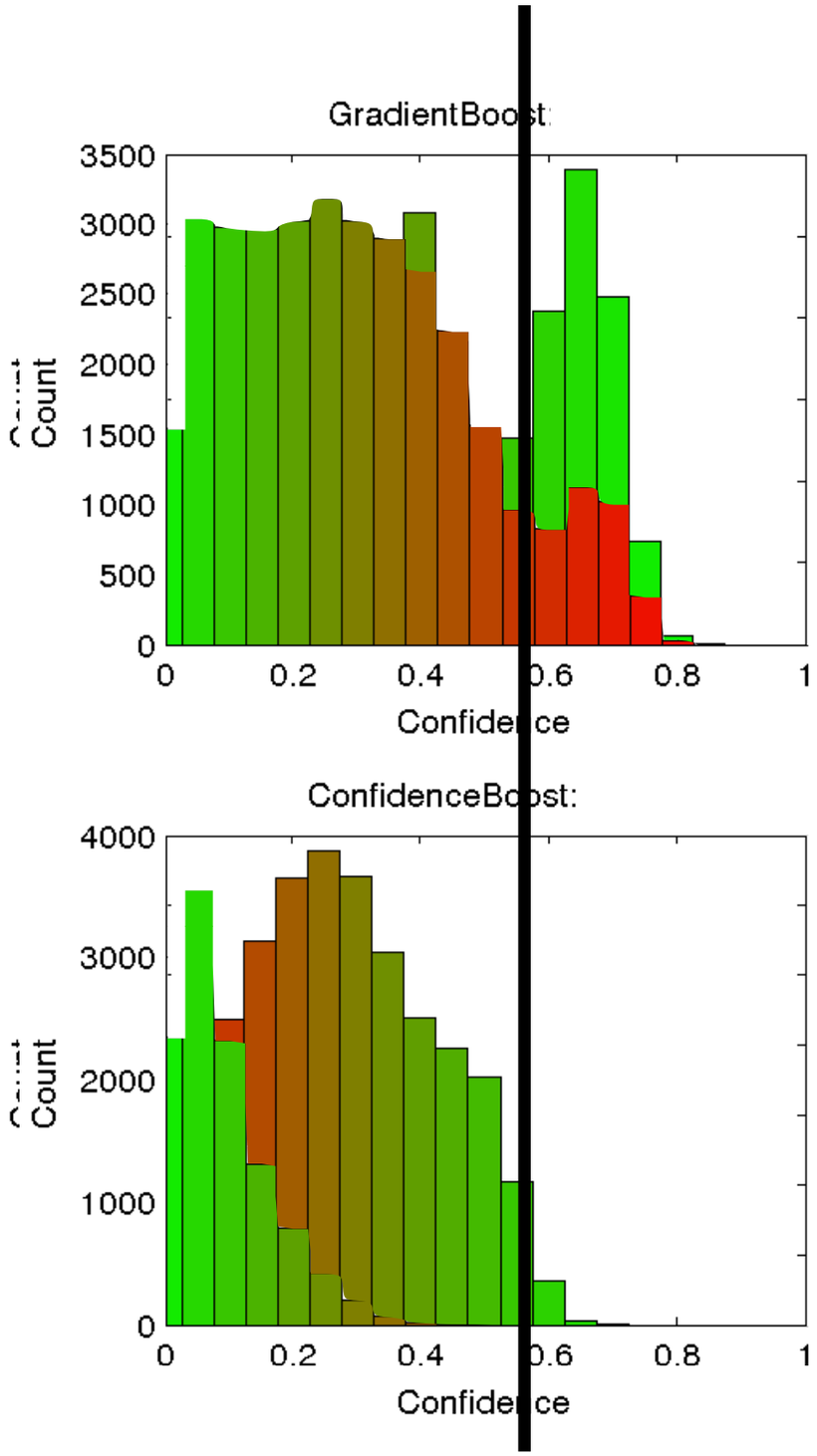


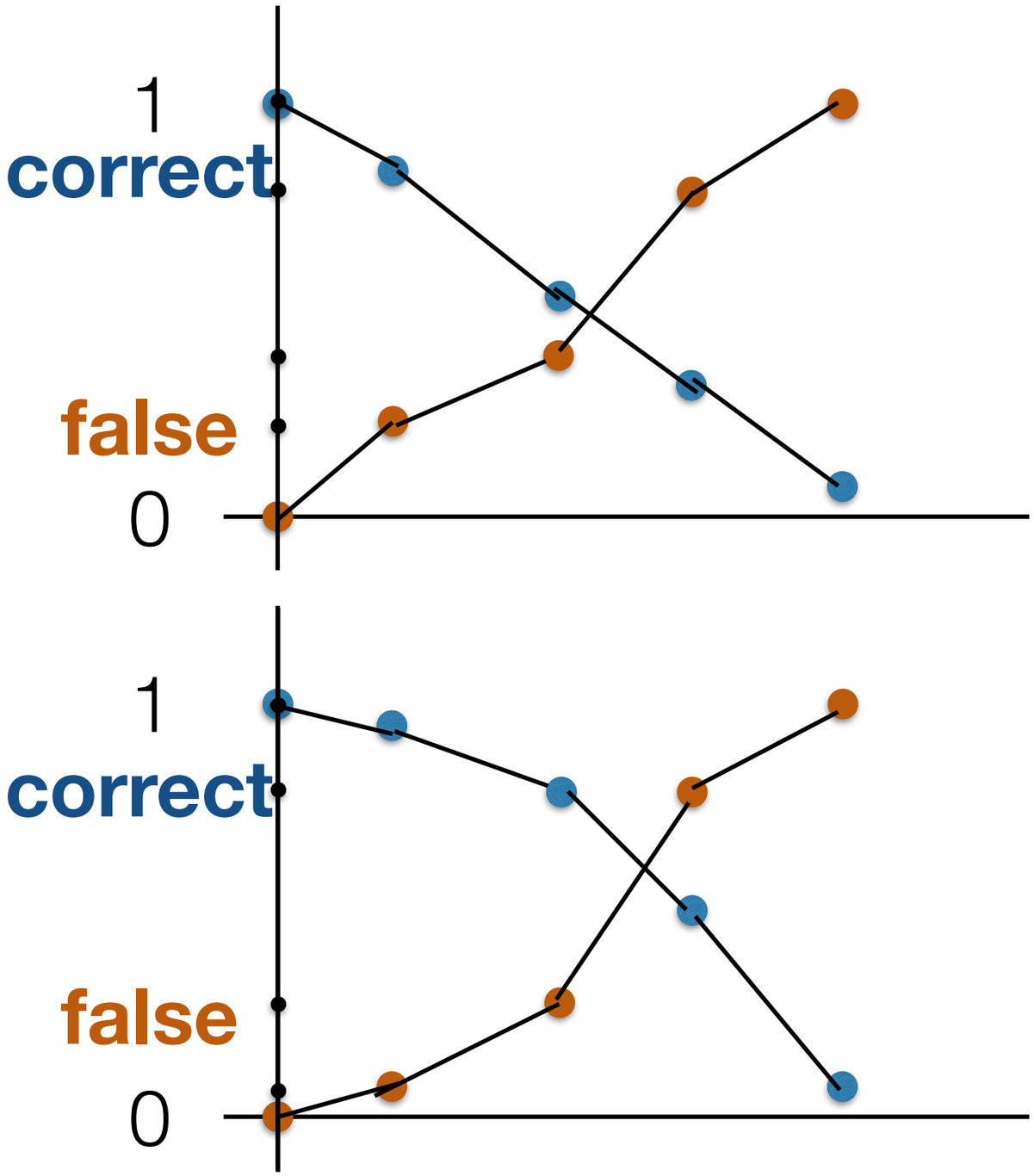
How Can We Find a Good Threshold?



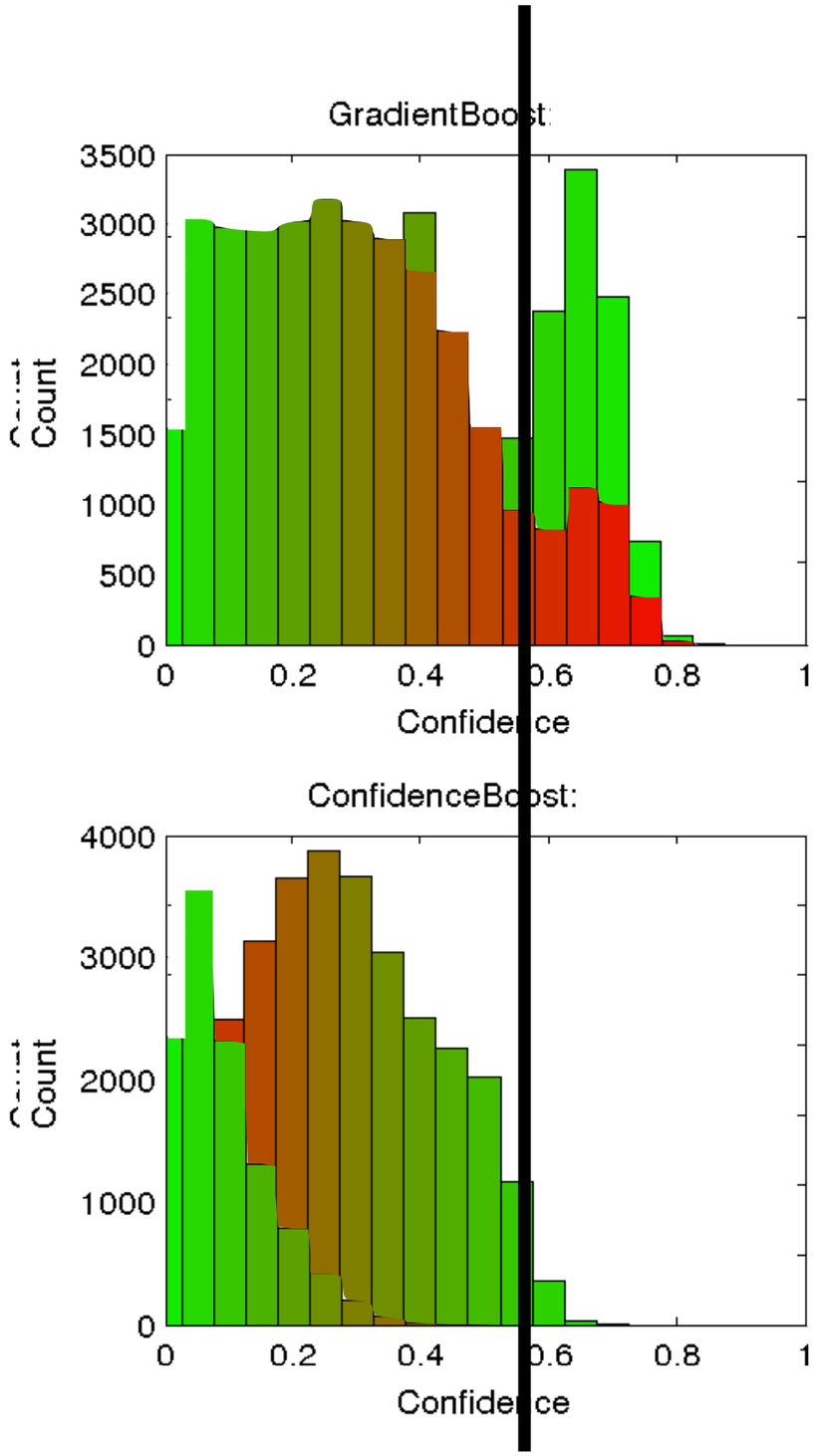


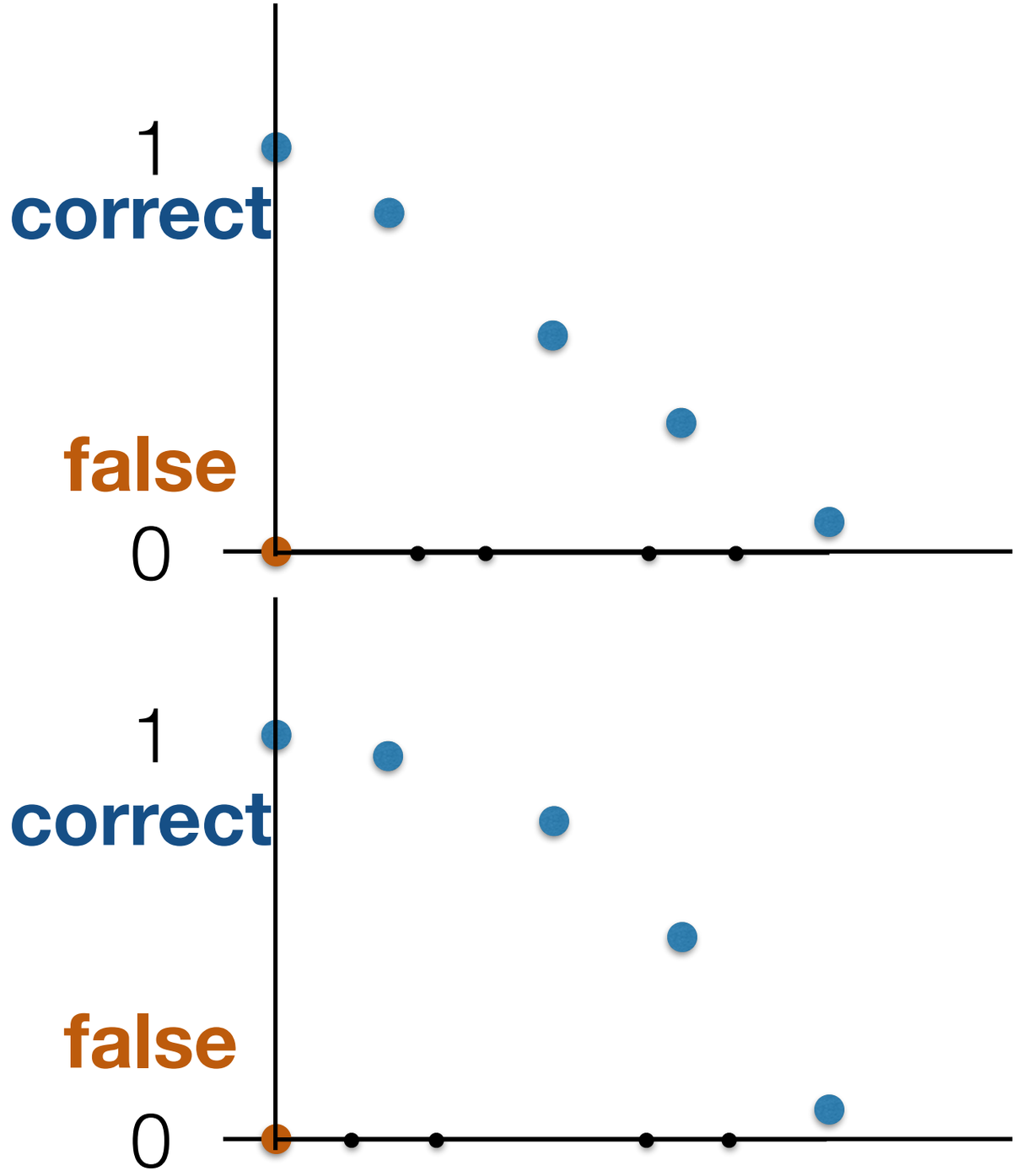
How Can We Find a Good Threshold?



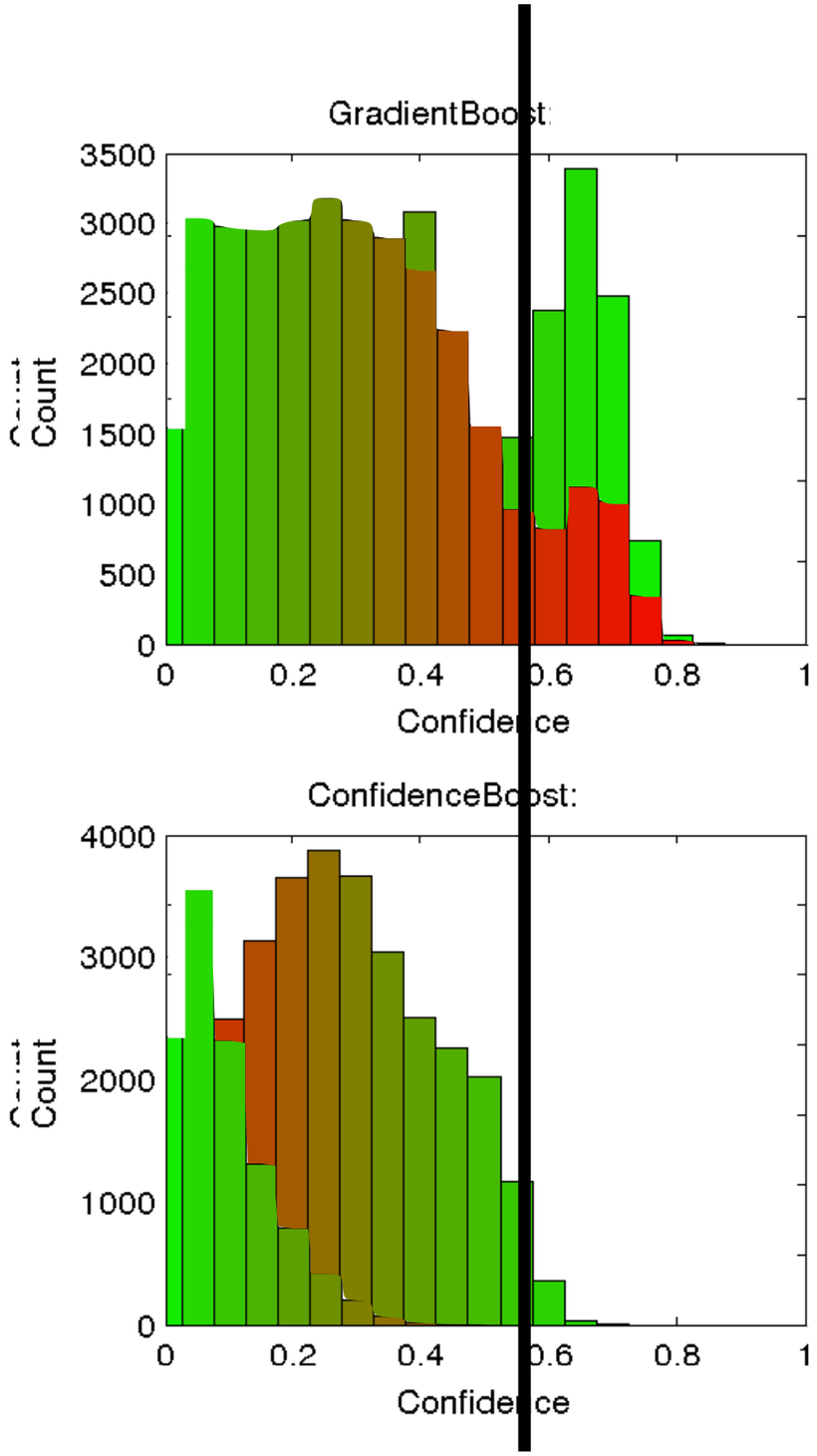


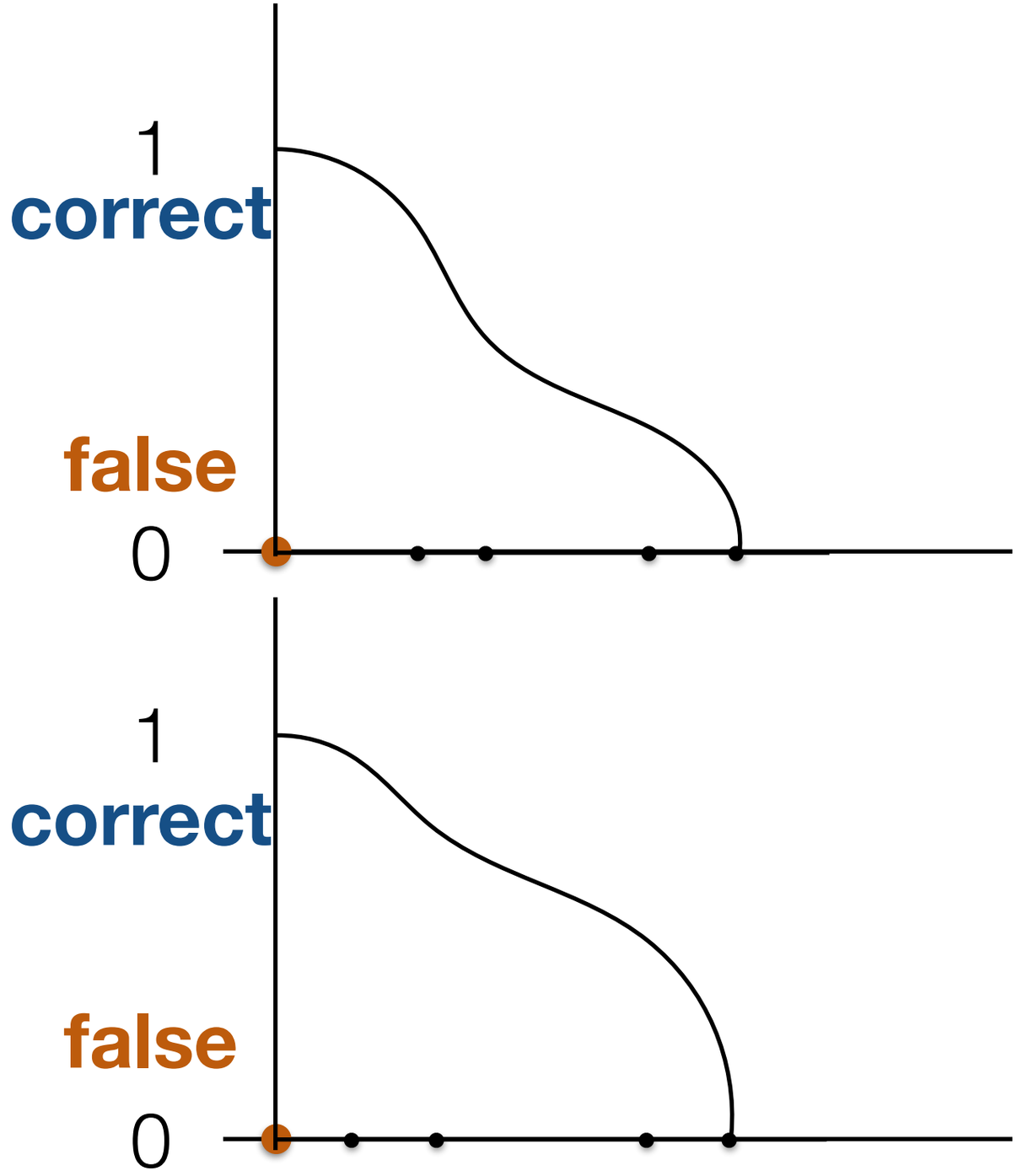
How Can We Find a Good Threshold?



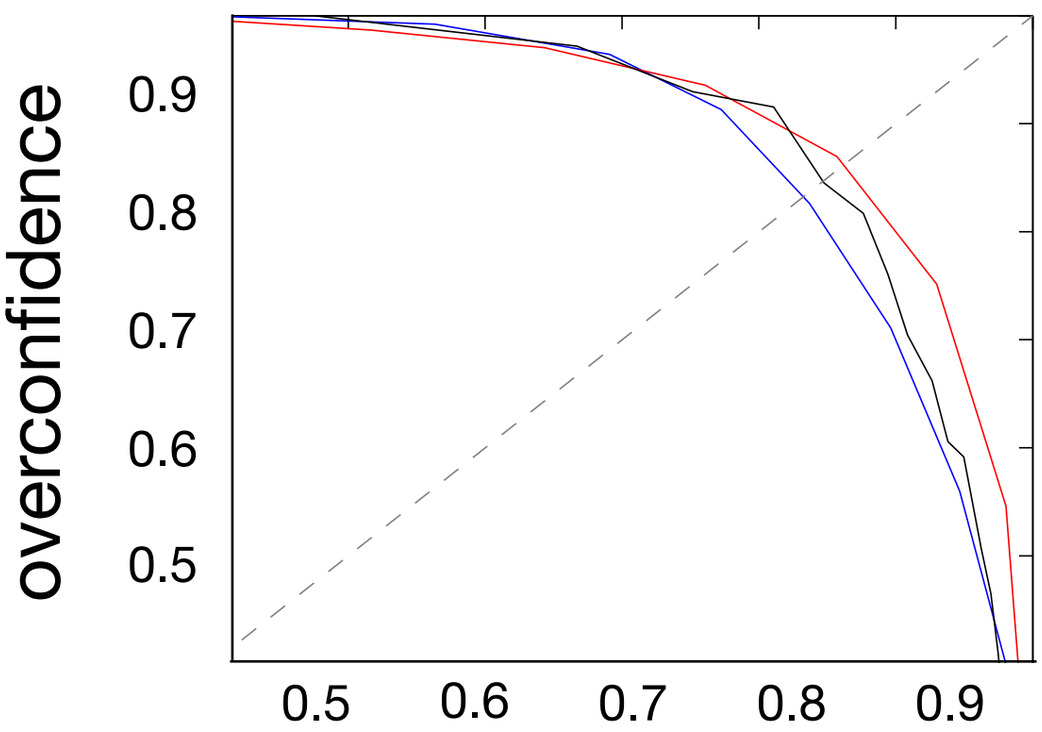


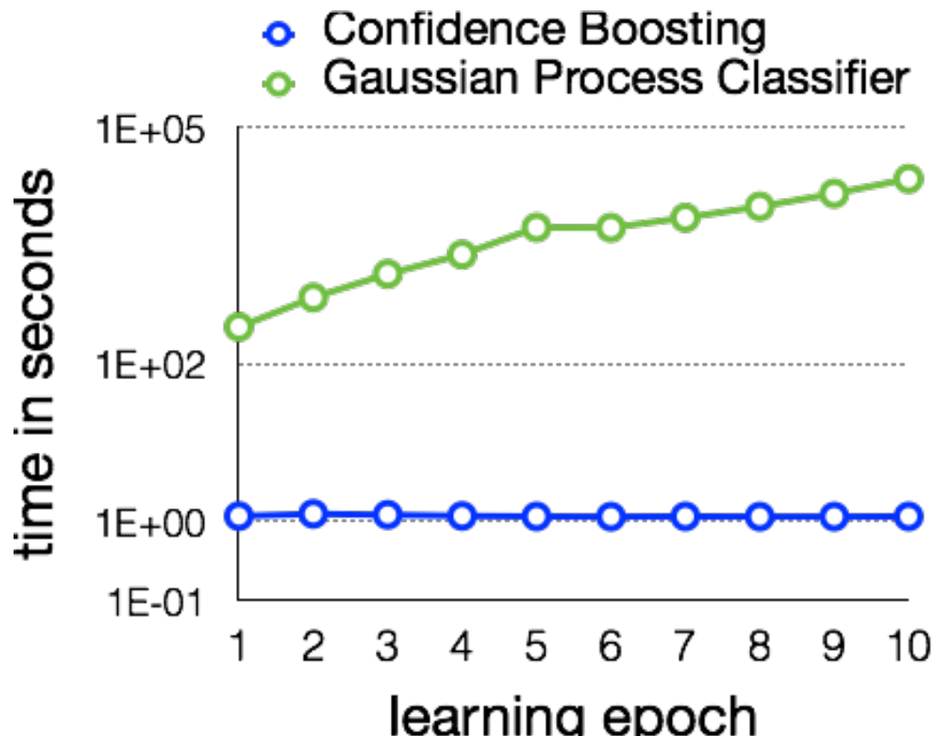
How Can We Find a Good Threshold?





Visualizing Overconfidence





Trade-off curve similar to precision-recall:

- x-axis: fraction of wrong samples below thresholds
- y-axis: fraction of correct samples above thresholds

Result: CB is not more overconfident than GPC

But: Runtime of GPC orders of magnitude larger than Confidence Boost

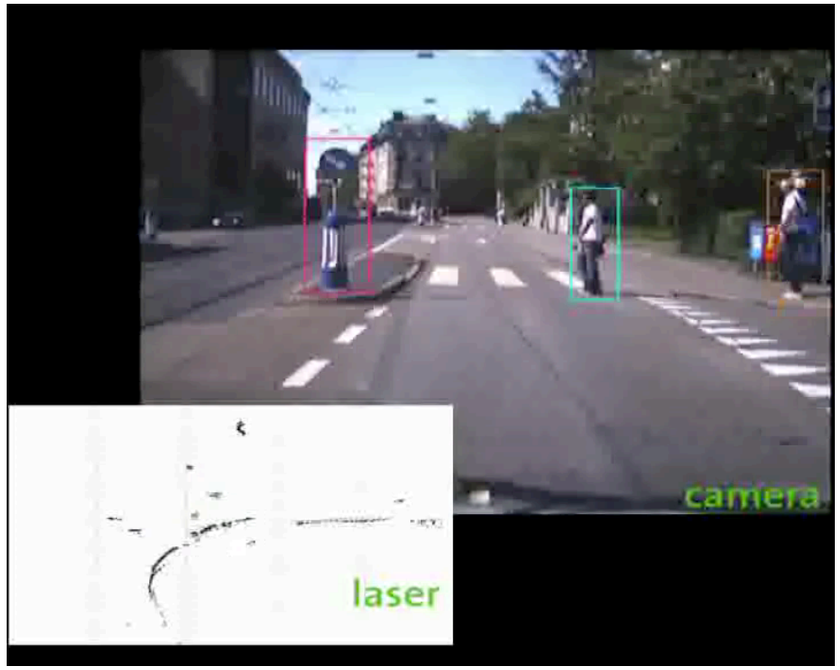
Pool-based and Stream-based AL

Problem: Most often, data occurs in **streams** (online) and not in **pools** (offline)

But: Online Random Forests

- require a balanced class dist.
- depend on the order of the data

Idea: use Mondrian Forests [1]



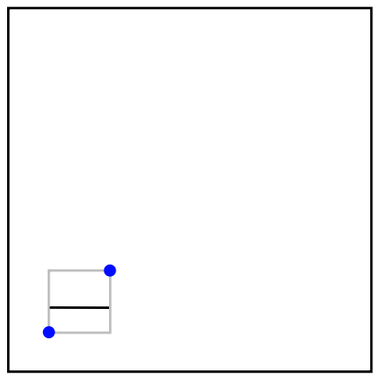
data stream

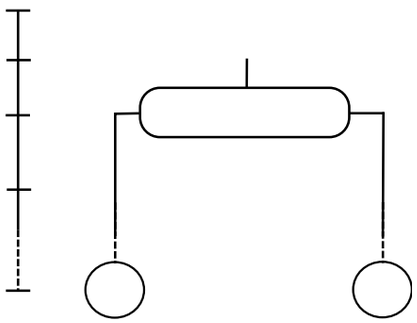
Main differences:

- MFs also store the **range** of the data in each dimension
- MFs are independent on the class labels

[1] B. Lakshminarayanan, D. M. Roy, and Y. W. Teh, "Mondrian Forests: Efficient Online Random Forests," in *Advances in Neural Information Processing Systems (NIPS)*, 2014, pp. 3140–3148.

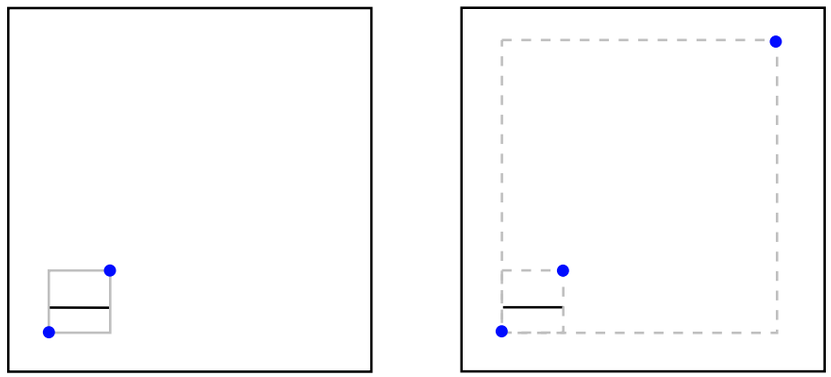
Mondrian Forests

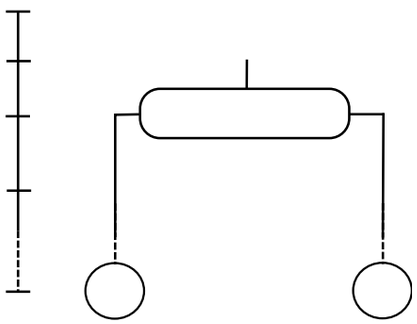




- start with two points
- insert a split

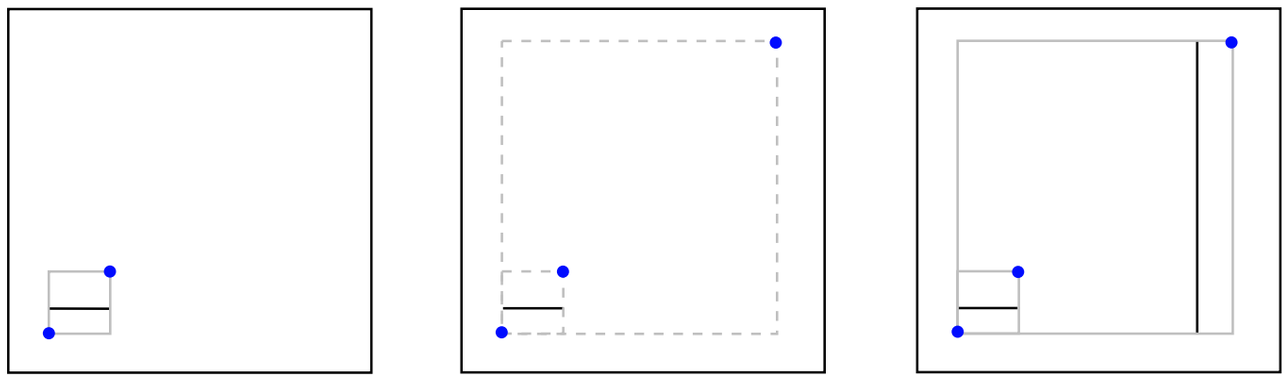
Mondrian Forests

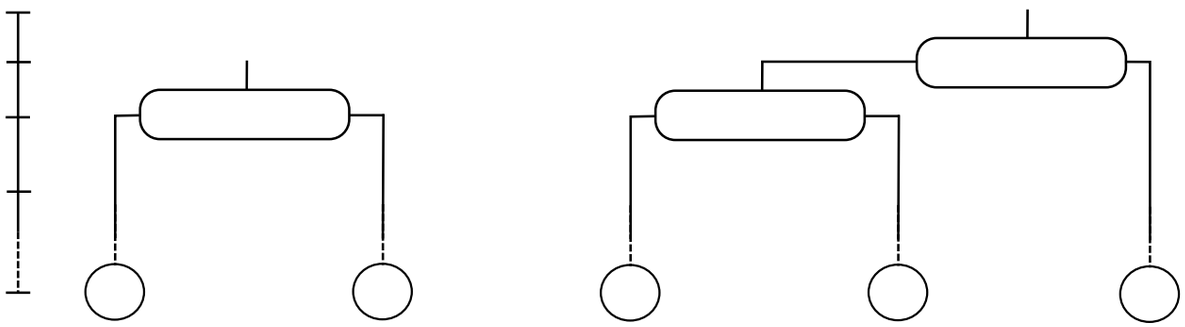




- start with two points
- insert a split
- add a third point

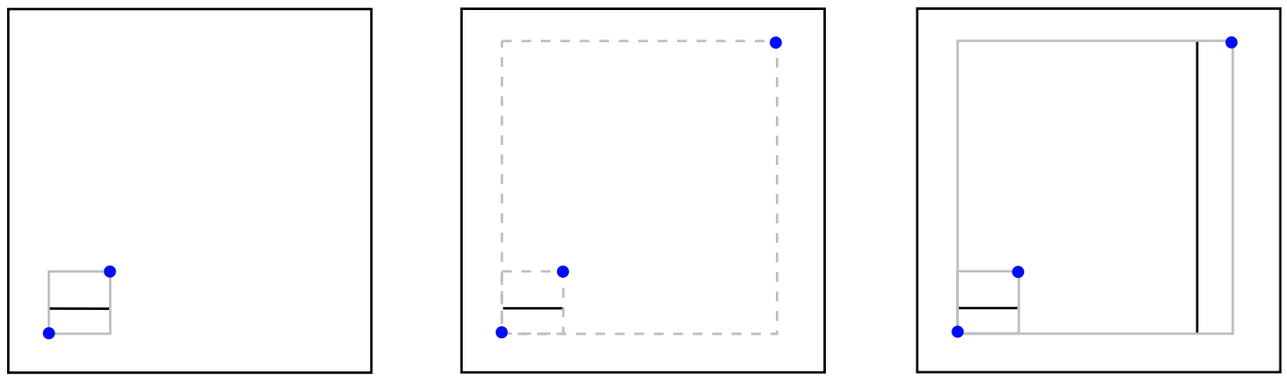
Mondrian Forests

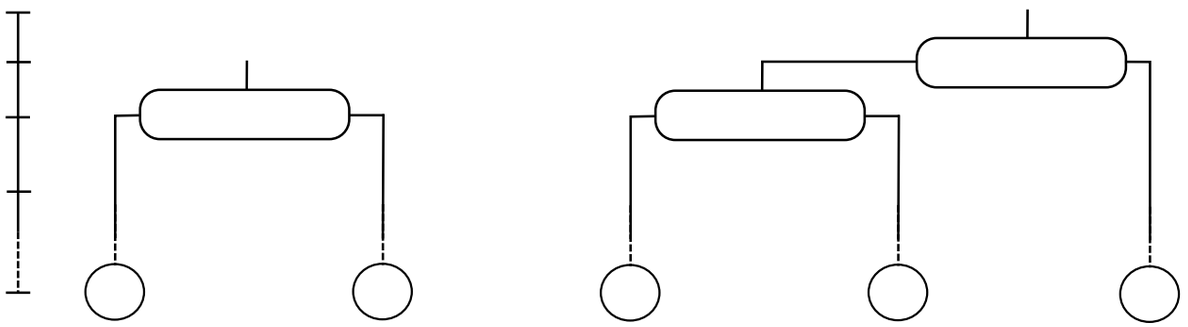


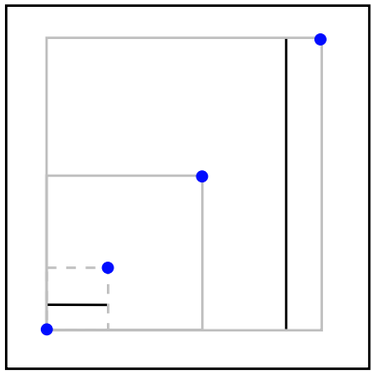


- start with two points
- insert a split
- add a third point
- sample a “time” from an exponential distr.
- insert a node **above** the given node
- insert a random split

Mondrian Forests

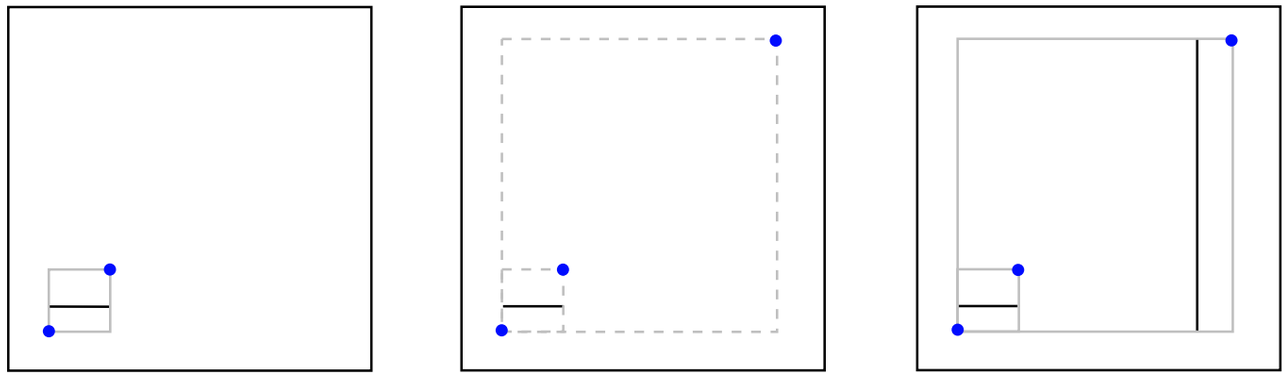


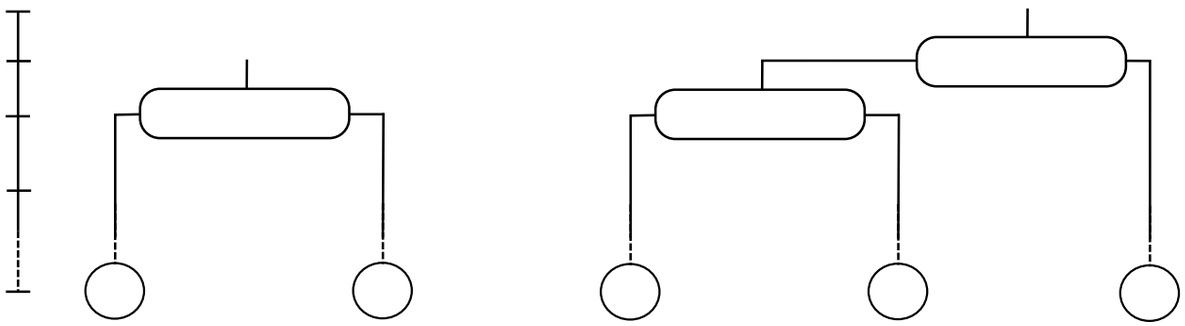


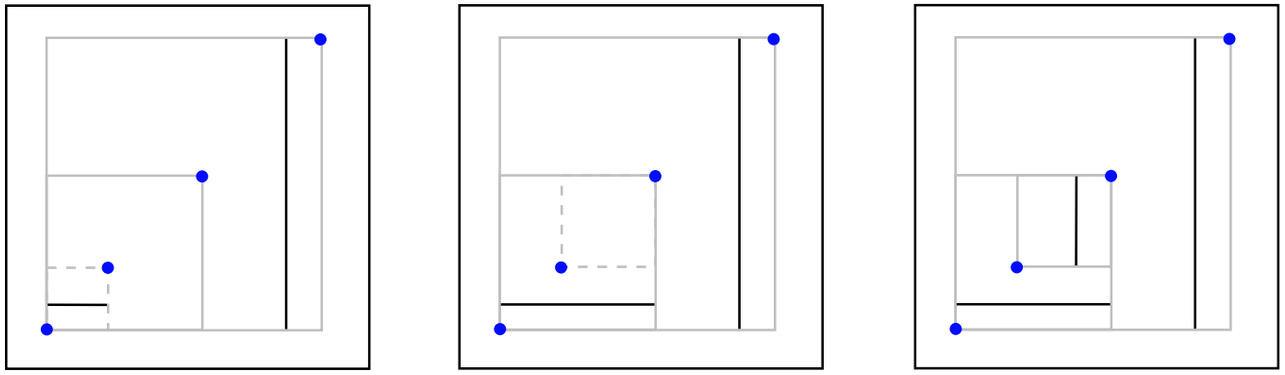


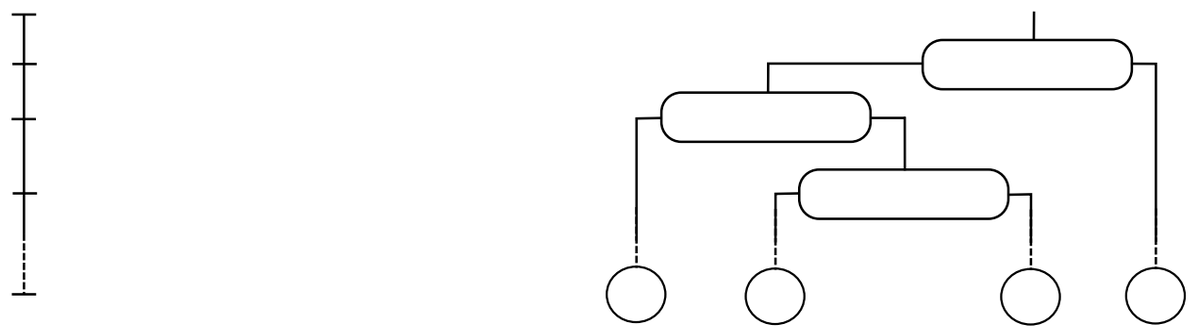
- start with two points
- insert a split
- add a third point
- sample a “time” from an exponential distr.
- insert a node **above** the given node
- insert a random split
- add a new node inside the outer box

Mondrian Forests





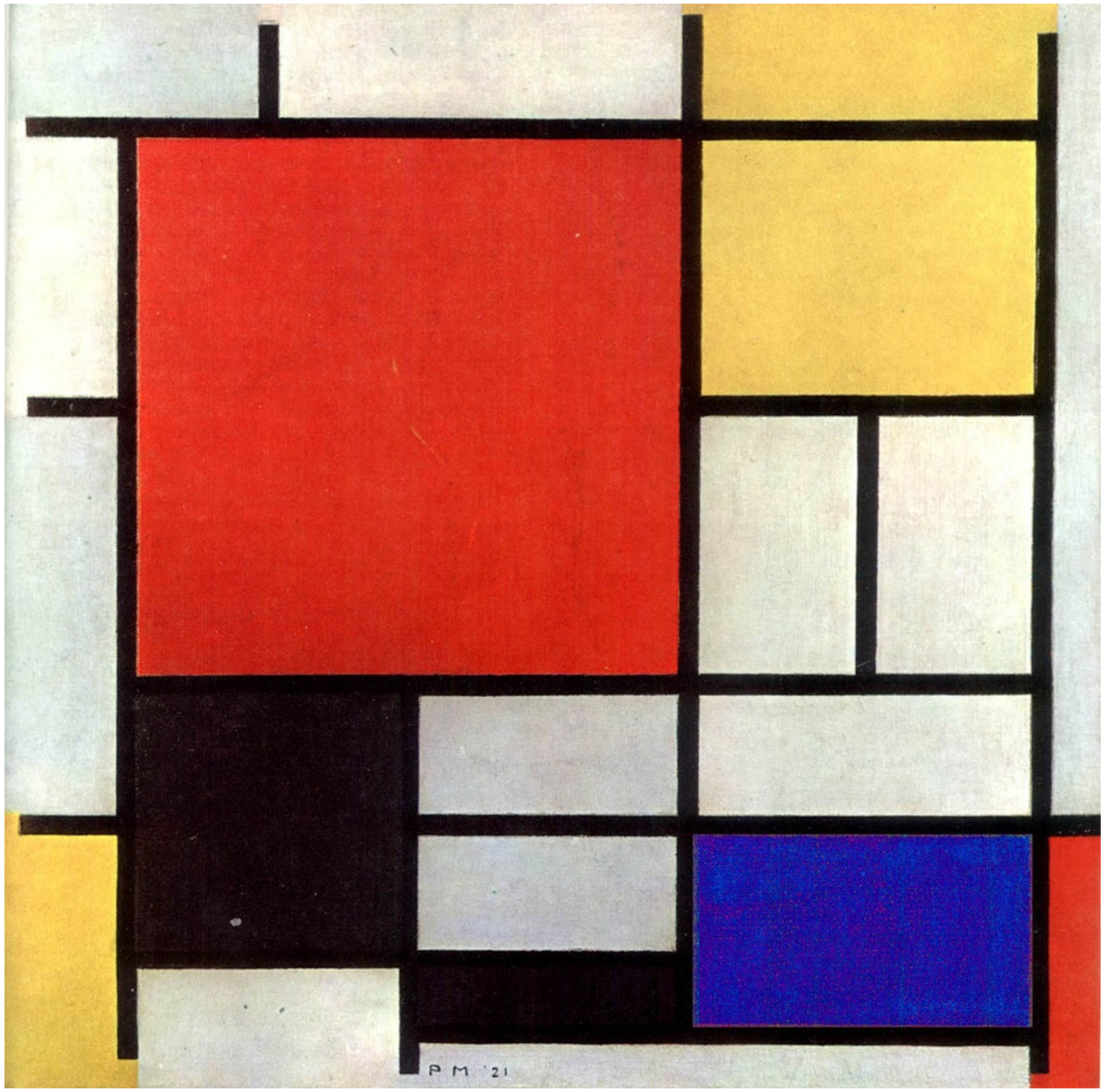




- start with two points
- insert a split
- add a third point
- sample a “time” from an exponential distr.
- insert a node **above** the given node
- insert a random split
- add a new node inside the outer box
- sample again and insert new node at the end
- this requires extending a split

Why the Name “Mondrian” Forests?

Why the Name “Mondrian” Forests?



Piet Mondrian: “*Composition with Red, Yellow, Blue, and Black*” 1926

Gemeentemuseum, Den Haag

Application to Real Data

KiTTI benchmark data set:

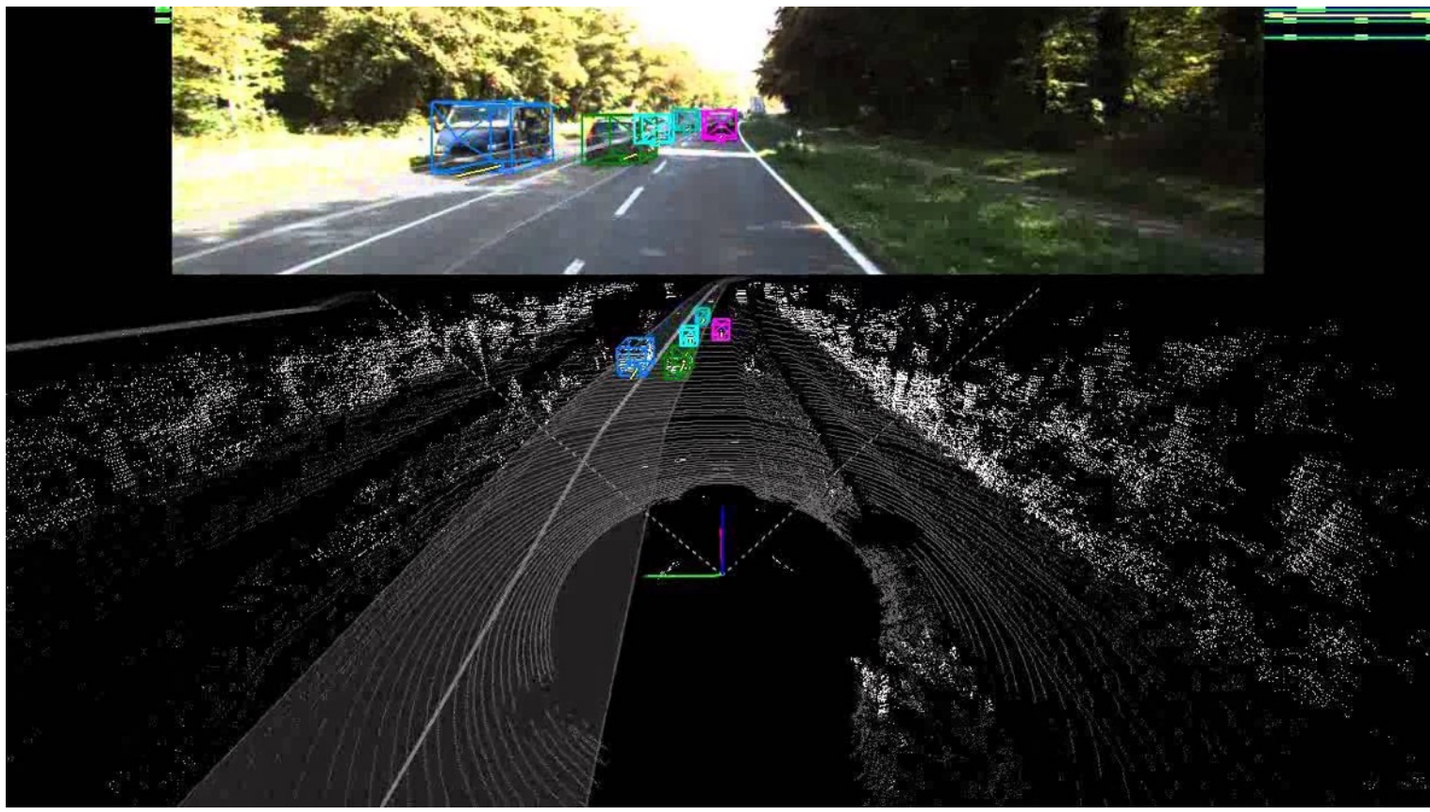
- 3D point clouds
- cars, pedestrians, bikes, trucks, etc.
- segmentation given (“tracklets”)

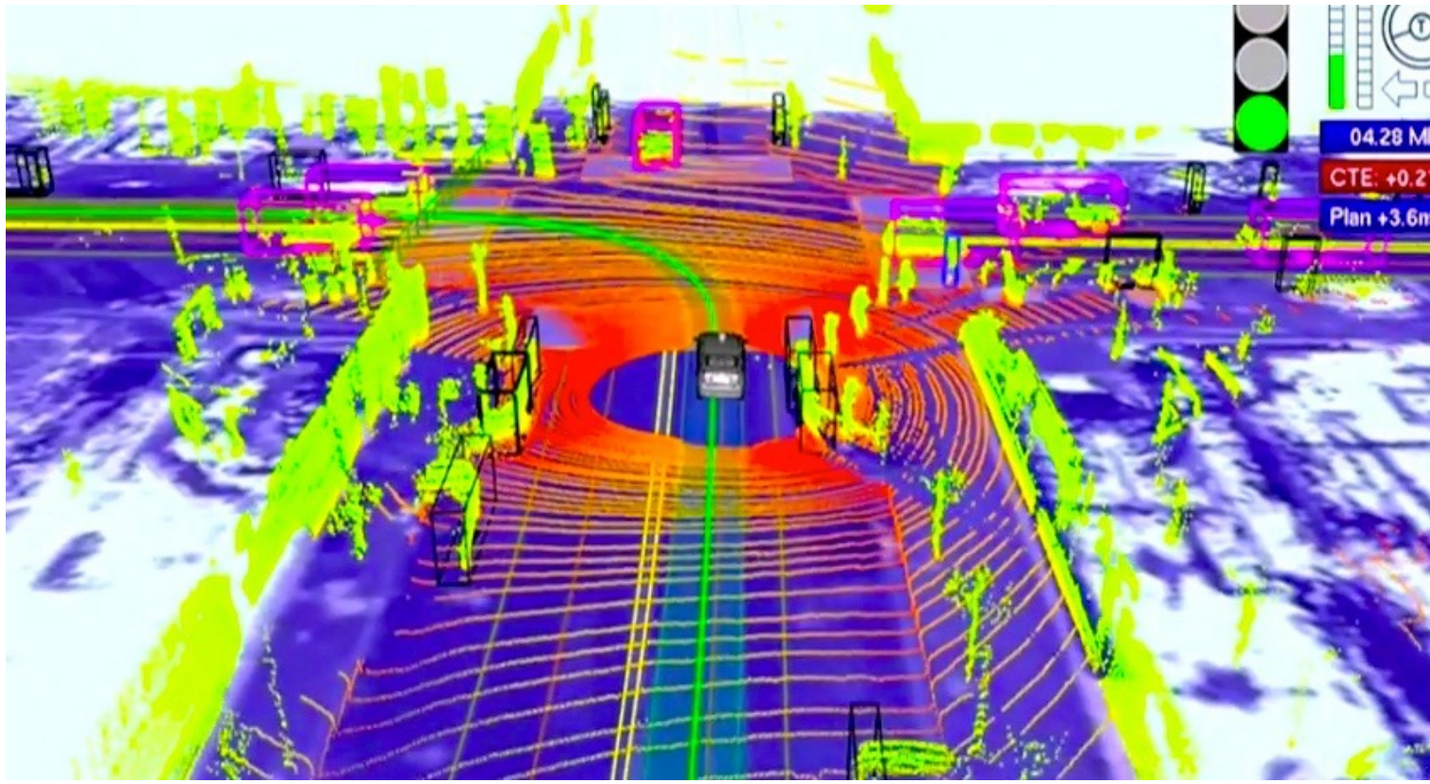
Goal:

- online classification
- use active learning

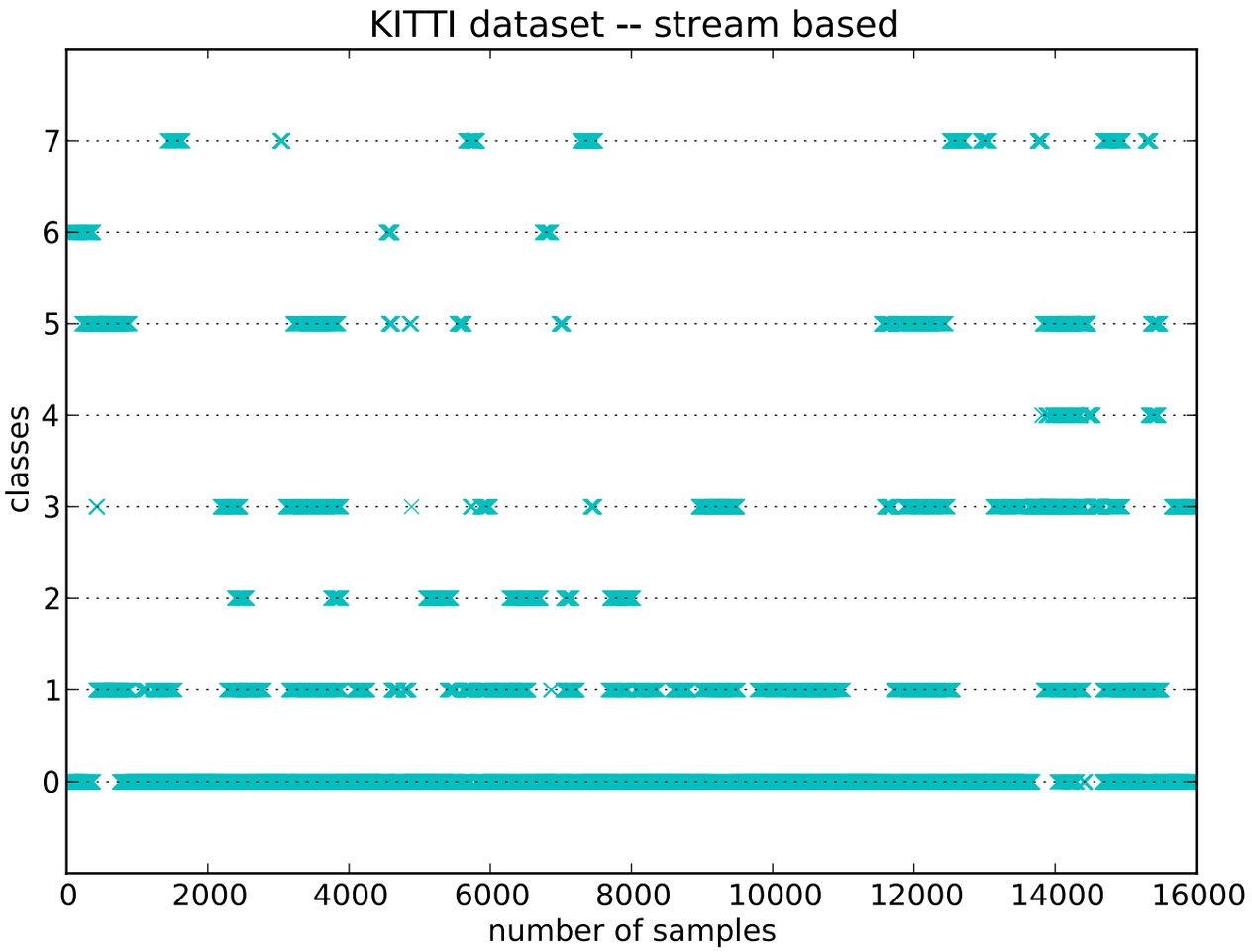
Major problems:

- un-balanced classes
- order of appearance

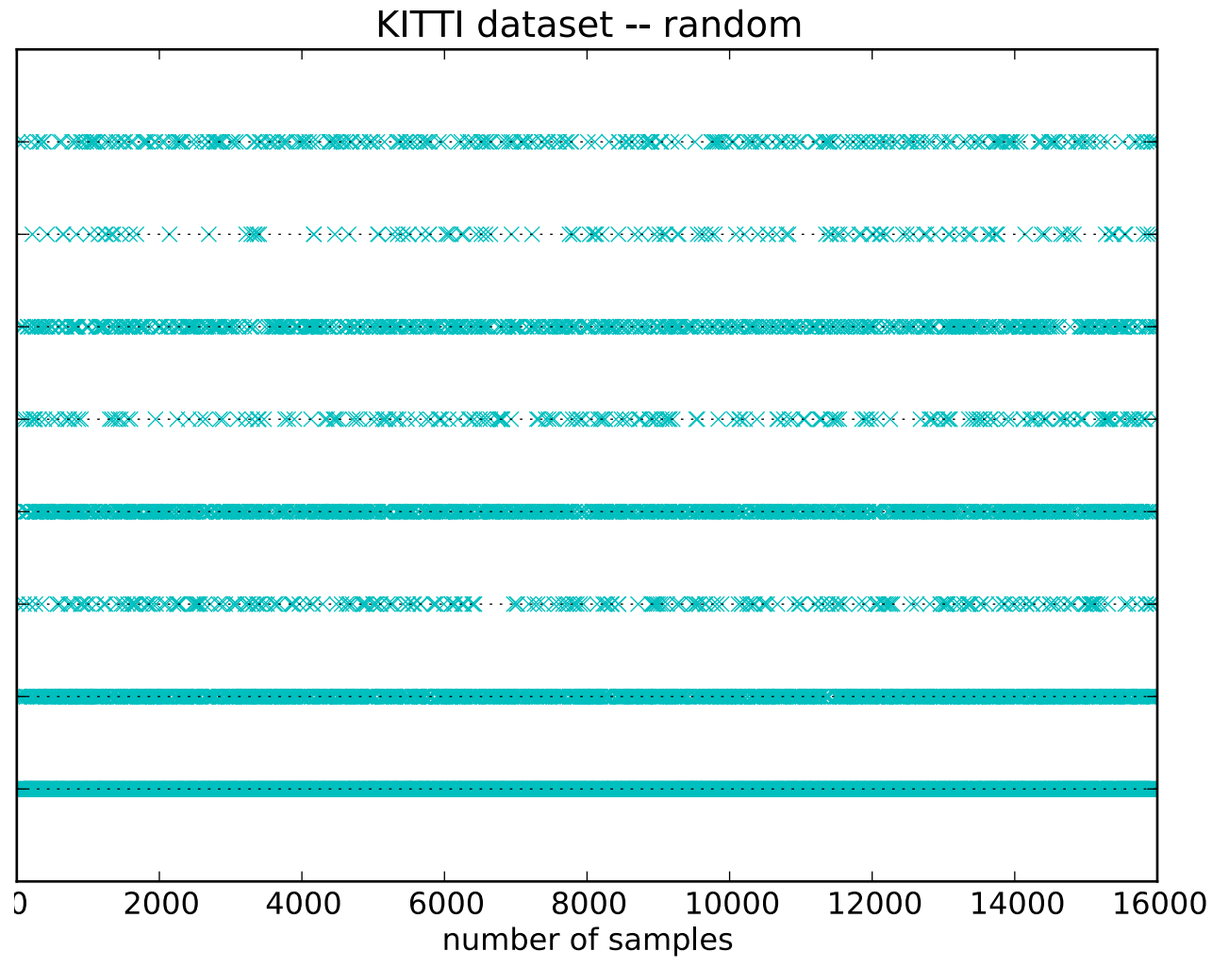




The Data Set

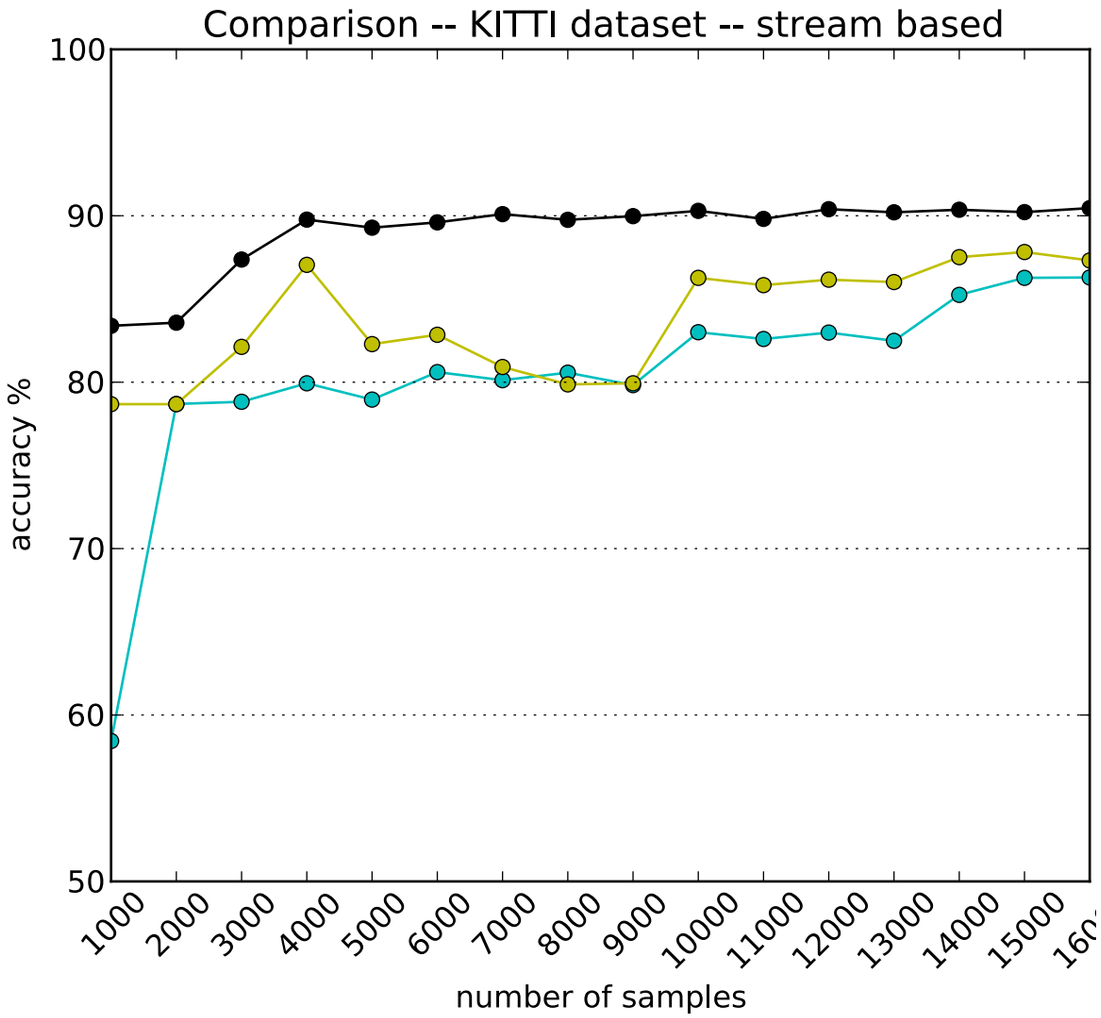


The real data

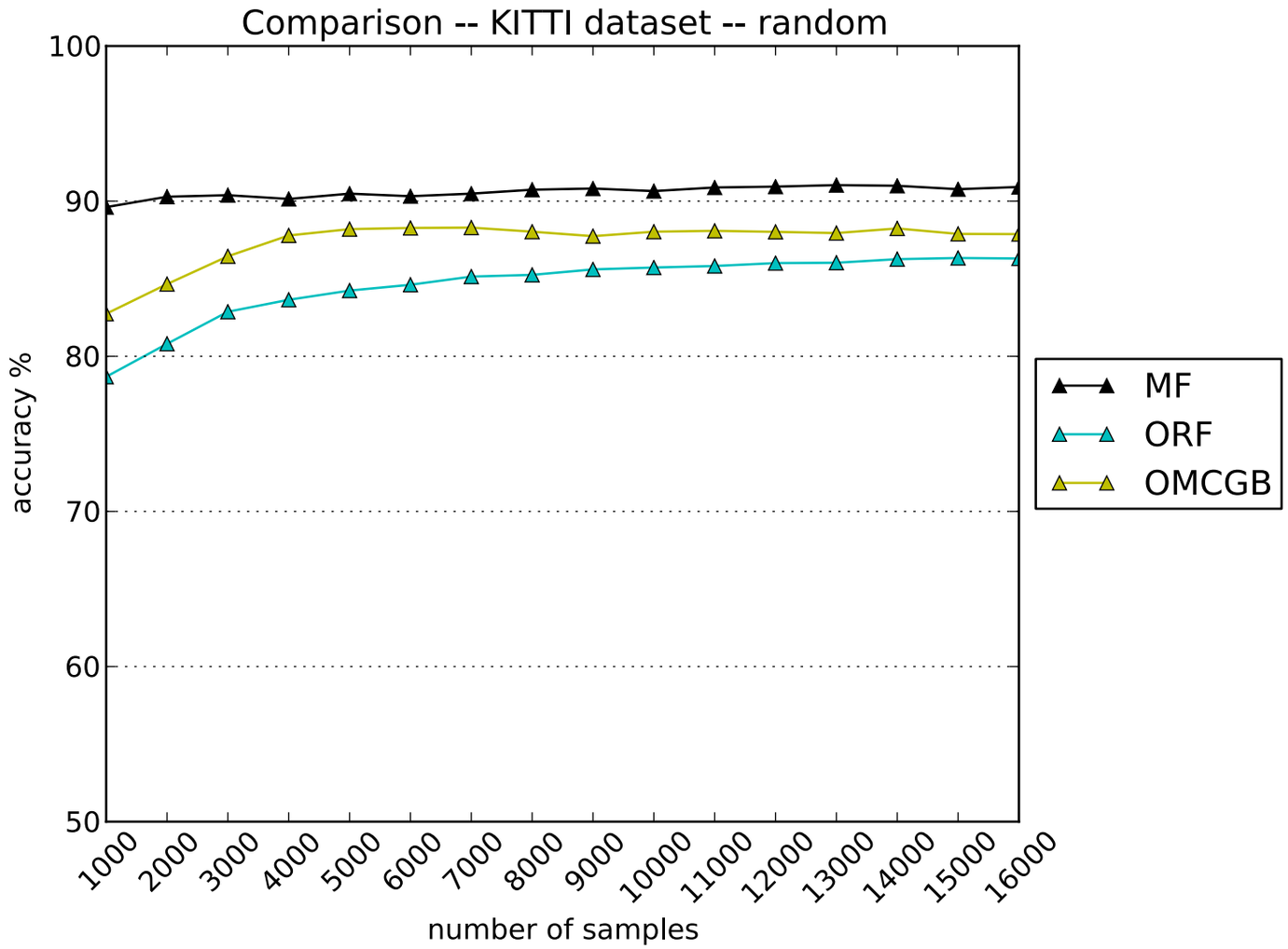


The data uniformly resampled

Results (no Active Learning)



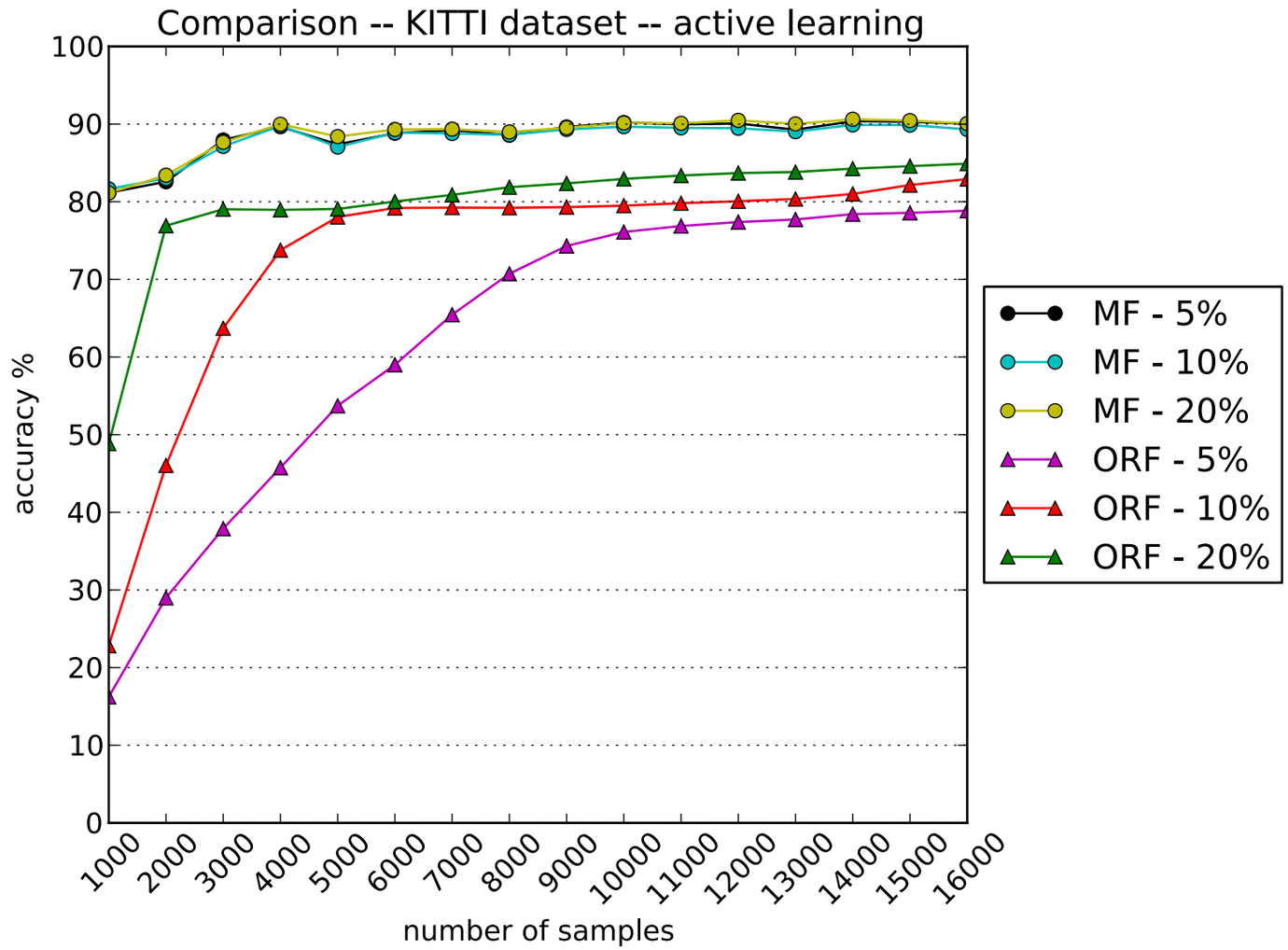
The real data



The data uniformly resampled

A. Narr, R. Triebel, D. Cremers “Stream-based Active Learning for Efficient and Adaptive Classification of 3D Objects” in: *ICRA* 2016

Results Using Active Learning



- Mondrian Forests require only 5% of the data to reach 90% accuracy
- Standard Random Forests can not reach that

A. Narr, R. Triebel, D. Cremers “Stream-based Active Learning for Efficient and Adaptive Classification of 3D Objects” in: *ICRA* 2016

Unknown Number of Classes

	Classes	0	1	2	3	4	5	6	7	8	9
Data sets	S_1	124	129	124	129	0	0	0	0	0	0
	S_2	124	128	124	128	252	247	0	0	0	0
	S_3	124	128	124	128	126	124	371	378	0	0
	S_4	124	128	124	128	126	124	124	126	495	504
	T	55	57	57	56	55	55	56	55	55	56

- Data sets have increasing number of classes
- Mondrian Forests can deal much better with unknown classes

Data sets	S_1	S_2	S_3	S_3
Mondrian Forests	39.21 %	58.78 %	78.36 %	95.21 %
ORF	33.87 %	53.57 %	72.02 %	87.24 %
OMCGB	29.48 %	34.67 %	59.28 %	60.03 %
OMCLP	27.02 %	39.44 %	60.78 %	63.10 %

A. Narr, R. Triebel, D. Cremers “Stream-based Active Learning for Efficient and Adaptive Classification of 3D Objects” in: *ICRA* 2016

Summary

- Online learning has important advantages over offline learning: **efficiency** and **adaptivity**
- In active learning the algorithm **selects** the data to learn from (human in the loop)
- This requires good **confidence** estimates
- The GP classifiers tends to be less **overconfident,** but inefficient
- A faster alternative method is **Confidence Boosting**, it is very efficient and not much overconfident
- For stream-based Active Learning **Mondrian Forests** are better suited