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

But: non-adaptive; large training set required

Fused detection and tracking of cars and pedestrians

[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
  \[ f_0(x) = \arg\min_\gamma \sum_{i=1}^{N} L(t_i, \phi(x_i, \gamma)) \]

- for \( m = 1, \ldots, M \)
  - Compute the gradient residual
    \[ r_{im} = - \left[ \frac{\partial L(t_i, f(x_i))}{\partial f(x_i)} \right] \]
  - Use the weak learner to compute \( \gamma_m \) that minimizes
    \[ \sum_{i=1}^{N} (r_{im} - \phi(x_i; \gamma_m))^2 \]
  - Update
    \[ f_m(x) = f_{m-1}(x) + \nu \phi(x, \gamma) \]
  - Return \( f(x) = f_M(x) \)
Online Gradient Boost

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

- For each point, all weak classifiers are updated.

- **Agreement** $\alpha$ between prediction $p_{nm}$ and ground truth $y_n$: $a(p_{nm}, y_n) = p_{nm}^{(y_n)} - 1/C$

- Sample receives low weight if agreement high.

---

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

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

1. Initialize($f_1, \ldots, f_M$)
2. for $n = 1, \ldots, N$ do
   3. $w_n \leftarrow 1$
   4. $g_n \leftarrow 0$
   5. for $m = 1, \ldots, M$ do
      6. $f_m \leftarrow \text{UpdateWeakLearner}(f_m, x_n, y_n, w_n)$
      7. $p_{nm} \leftarrow f_m(x_n)$
      8. $\alpha_{nm} \leftarrow a(p_{nm}, y_n)$
      9. $g_n \leftarrow g_n + \alpha_{nm}$
     10. $w_n \leftarrow -\nabla \ell(g_n)$
   11. end
   12. end
```

Online Random Forest

• As in Online Boosting, the two for loops are switched:
  • outer loop over all 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
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**
  - Optimization
    - Function $f(x)$
  - Prediction
  - Supervisor
    - Uncertainty, Labels
  - Label Query
    - New training data
  - Extend training data

- **New Data**
  - New Data
  - Optimization
  - Prediction
  - Supervisor
  - Label Query
  - Extend training data

- **Uncertainty, Labels**

- **Training data**
  - "Laptop"
  - "Cup"
  - "Present"

- **Function**
  - $f(x)$

- **Graph**
  - X-axis: $x$
  - Y-axis: $y$
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

- Non-linear GPC
- Linear GPC
- Non-linear SVM
- Linear SVM
- Logit Boost

Trained classes
Unseen classes

The GP is less overconfident than the SVM!

[Grimmett, Paul, Triebel, Posner ICRA 2013]
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: \( f(x) \sim \mathcal{GP}(m(x), k(x, x')) \)

2. New function values \( f_* \) can be predicted by \textbf{conditioning} on known values:

\[
p(f_* \mid X, f, x_*) = \mathcal{N}(k_*^T K^{-1} f, k(x_*, x_*) - k_*^T K^{-1} k_*)
\]

3. In the presence of noise we \textbf{marginalize} over the latent functions:

\[
p(f_* \mid X, y, x_*) = \int p(f_* \mid X, x_*, f)p(f \mid X, y)df
\]

4. For classification, we squash the latent function through a \textbf{sigmoid}:

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

5. We get the predictive distribution via \textbf{marginalization}:

\[
p(y_* = +1 \mid X, y, x_*) = \int \Phi(f_*)p(f_* \mid X, y, x_*)df_*
\]

from: Rasmussen and Williams, (2006)
Why is the GPC Less Overconfident?

single model (SVM)

\[ p(y^* = 1 \mid f(x^*)) = \frac{1}{1 + \exp(a f(x^*) + b)} \]

marginalization (GPC)

\[ p(f_* \mid X, y, x^*) = \int p(f_* \mid X, x^*, f)p(f \mid X, y)df \]

• 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

![Graph showing the relationship between run time and memory, and precision and recall over different methods (Camera, Laser, Fusion)]
The 3 Criteria of Learning Algorithms

**Efficiency**
- memory
- run time

**Accuracy**
- precision
- recall

**Confidence**
- overconfidence
- underconfidence
Which Classifier Should We Choose?

Accuracy

Efficiency

- GPC
- SVM
- Boosting
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 $a_c$:

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

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

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

$$h(p) := -\sum_{i=1}^{C} p_i \log_{C}(p_i)$$
Example: AL for 3D Object Recognition

Qualitative Results:

True Label = lime

Epoch 10

Gradient Boost --> lemon
Confidence Boost --> lime

True Label = binder

Epoch 10

Gradient Boost --> calculator
Confidence Boost --> binder

True Label = bowl

Epoch 10

Gradient Boost --> coffeeMug
Confidence Boost --> 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?

Histograms showing the distribution of confidence values for GradientBoost and ConfidenceBoost for both correct and false predictions.
How Can We Find a Good Threshold?

![Graph showing the comparison between GradientBoost and ConfidenceBoost distributions.](image)

- **Correct** points are marked in green, and **false** points are marked in red.

The graphs illustrate the distribution of confidence levels for correct and false predictions, helping to identify a suitable threshold for classification tasks.
How Can We Find a Good Threshold?

![Histograms of Confidence for GradientBoost and ConfidenceBoost](chart)

- **GradientBoost**
  - Count distribution for confidence levels 0, 0.2, 0.4, 0.6, and 0.8.
  - Correct predictions for confidence levels 0.6 and 0.8, but not 0.2.

- **ConfidenceBoost**
  - Count distribution for confidence levels 0, 0.2, 0.4, 0.6, and 0.8.
  - Incorrect predictions for confidence levels 0 and 0.2, but correct for 0.6 and 0.8.

---

*PD Dr. Rudolph Triebel*
*Computer Vision Group*
How Can We Find a Good Threshold?

![Histograms and graphs showing the distribution of confidence scores and their corresponding counts for GradientBoost and ConfidenceBoost.](image)

- **GradientBoost**:
  - Confidence vs. Count graph with bars indicating distribution.
  - Example points marked as 'correct' and 'false'.

- **ConfidenceBoost**:
  - Similar graph with different distribution.
  - Example points marked as 'correct' and 'false'.

Note: The histograms and graphs illustrate the distribution of confidence scores and the corresponding counts, helping in identifying thresholds for classification tasks.
How Can We Find a Good Threshold?

- **GradientBoost**
  - Count vs. Confidence
  - Graph showing data points and trend lines

- **ConfidenceBoost**
  - Count vs. Confidence
  - Graph showing data points and trend lines

The diagrams illustrate the distribution of data points and their corresponding confidence levels, indicating whether the predictions are correct or false.
How Can We Find a Good Threshold?
How Can We Find a Good Threshold?
How Can We Find a Good Threshold?

![Graphs showing ConfidenceBoost and GradientBoost with corresponding correct and false markers.](image-url)
How Can We Find a Good Threshold?

- **GradientBoost**
  - Confidence vs. Count
  - Correct vs. False

- **ConfidenceBoost**
  - Confidence vs. Count
  - Correct vs. False
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]

Main differences:
• MFs also store the range of the data in each dimension
• MFs are independent on the class labels

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

KITTI dataset -- stream based

KITTI dataset -- random

The real data

The data uniformly resampled
Results (no Active Learning)

Comparison -- KITTI dataset -- stream based

The real data

Comparison -- KITTI dataset -- random

The data uniformly resampled

Mondrian Forests require only 5% of the data to reach 90% accuracy

Standard Random Forests can not reach that

### Unknown Number of Classes

<table>
<thead>
<tr>
<th>Classes</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S_1$</td>
<td>124</td>
<td>129</td>
<td>124</td>
<td>129</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$S_2$</td>
<td>124</td>
<td>128</td>
<td>124</td>
<td>128</td>
<td>252</td>
<td>247</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$S_3$</td>
<td>124</td>
<td>128</td>
<td>124</td>
<td>128</td>
<td>126</td>
<td>124</td>
<td>371</td>
<td>378</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$S_4$</td>
<td>124</td>
<td>128</td>
<td>124</td>
<td>128</td>
<td>126</td>
<td>124</td>
<td>124</td>
<td>126</td>
<td>495</td>
<td>504</td>
</tr>
<tr>
<td>$T$</td>
<td>55</td>
<td>57</td>
<td>57</td>
<td>56</td>
<td>55</td>
<td>55</td>
<td>56</td>
<td>55</td>
<td>55</td>
<td>56</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Data sets</th>
<th>$S_1$</th>
<th>$S_2$</th>
<th>$S_3$</th>
<th>$S_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mondrian Forests</td>
<td>39.21 %</td>
<td>58.78 %</td>
<td>78.36 %</td>
<td>95.21 %</td>
</tr>
<tr>
<td>ORF</td>
<td>33.87 %</td>
<td>53.57 %</td>
<td>72.02 %</td>
<td>87.24 %</td>
</tr>
<tr>
<td>OMCGB</td>
<td>29.48 %</td>
<td>34.67 %</td>
<td>59.28 %</td>
<td>60.03 %</td>
</tr>
<tr>
<td>OMCLP</td>
<td>27.02 %</td>
<td>39.44 %</td>
<td>60.78 %</td>
<td>63.10 %</td>
</tr>
</tbody>
</table>

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