Machine Learning in Computer Vision: Concepts and Applications

Matteo Munaro
Intelligent Autonomous Systems Lab (IAS-Lab)
Department Of Information Engineering
University of Padova

(contains slides from D. Hoiem and L. Lazebnik)
How does our brain solve this problem?

1. Analyze the given images
2. Try to find some similarities within each group
3. Given a query image, compare it with the training groups
4. Classify query image as belonging to the closest group
What is Machine Learning?

- Machine Learning (ML) provides algorithms that **improve** with experience
- Solves problems which cannot be solved by **enumerative** methods or **calculus-based** techniques
- When the **relationships** between all input/output system variables is completely **understood**, ML is **not needed**
Advantages of Machine Learning

- **Learning and writing an algorithm**
  - Easy for human brain, but tough for a machine
  - It takes some time and good amount of training data for a machine to accurately classify objects

- **Classification and scalability**
  - Easy for a machine. Once learnt, a machine can process one million images without any fatigue whereas human brain cannot
  - ML with big data is a deadly combination

- Most of the algorithm complexity can be moved to the learning stage and leave classification as simple and fast as possible.
Using machine learning to detect spam emails.

To: you@gmail.com
GET YOUR DIPLOMA TODAY!
If you are looking for a fast and cheap way to get a diploma, this is the best way out for you. Choose the desired field and degree and call us right now: For US: 1.845.709.8044 Outside US: +1.845.709.8044 "Just leave your NAME & PHONE NO. (with CountryCode)" in the voicemail. Our staff will get back to you in next few days!

ALGORITHM
Naïve Bayes
Rule mining
Using machine learning to recommend books.

ALGORITHMS

Collaborative Filtering
Nearest Neighbour
Clustering
Machine Learning applications

- Using machine learning to **identify faces** and expressions.

**ALGORITHMS**
- Decision Trees
- Adaboost
Using machine learning to identify vocal patterns

ALGORITHMS
- Feature Extraction
- Probabilistic Classifiers
- Support Vector Machines
+ many more….
Recognising spam emails
Recommended books
Reading handwriting
Recognising speech, faces
Driving a car

How would you write these programs?
Many applications are immensely hard to program directly. These almost always turn out to be “pattern recognition” tasks.

1. Program the computer to do the pattern recognition task directly. 

2. Provide “training” data.
Types of Machine Learning algorithms

- Supervised Learning Algorithms
- Unsupervised Learning Algorithms
- Semi-supervised Learning Algorithms
- Instance-based Algorithms
- Clustering Algorithms
- Dimensional Reduction Algorithms
- Bayesian Algorithms
- Artificial Neural Network Algorithms
- Decision Tree Algorithms
- Ensemble Algorithms
- Deep Learning Algorithms
Machine Learning problems

- Supervised Learning
  - Discrete: classification or categorization
  - Continuous: regression

- Unsupervised Learning
  - clustering
  - dimensionality reduction
Clustering: group together similar points and represent them with a single token.

Key challenges:
1) What makes two points/images/patches similar?
2) How do we compute an overall grouping from pairwise similarities?
Clustering techniques
Discrete & Unsupervised

- **K-means**
  - Iteratively re-assign points to the nearest cluster center

- **Agglomerative clustering**
  - Start with each point as its own cluster and iteratively merge the closest clusters

- **Mean-shift clustering**
  - Estimate modes of pdf (no need to know number of clusters)

- **Gaussian Mixture Models (GMM)**
  - Maximize likelihood of the data with regards to parameters of GMMs

As we go down this chart, the clustering strategies have more tendency to transitively group points even if they are not nearby in feature space
Clustering application: Image segmentation

How many clusters? Mean-shift

Machine Learning problems

Supervised Learning

<table>
<thead>
<tr>
<th>Continuous</th>
<th>Discrete</th>
</tr>
</thead>
<tbody>
<tr>
<td>regression</td>
<td>classification or categorization</td>
</tr>
</tbody>
</table>

Unsupervised Learning

<table>
<thead>
<tr>
<th>Continuous</th>
<th>Discrete</th>
</tr>
</thead>
<tbody>
<tr>
<td>dimensionality reduction</td>
<td>clustering</td>
</tr>
</tbody>
</table>
- As data dimension grows, the learning problem becomes more difficult.
- The number of configurations of the variables of interest increases exponentially with the dimensionality.
Mapping from higher-dimensional space to lower-dimensional space

- Feature selection
- Feature extraction (mapping)

Advantages of dimensionality reduction

- It reduces the time and storage space required.
- Removal of multi-collinearity (ill-conditioning) improves the performance of the machine learning model.
- It becomes easier to visualize the data when reduced to very low dimensions such as 2D or 3D.
**PCA:**
component axes that maximize the variance

**LDA:**
maximizing the component axes for class-separation

Typically used *before* classification or regression.
PCA application: Face recognition with EigenFaces
Continuous & Unsupervised

Training:

![Training Image](image1)

Testing:

![Testing Image](image2)

### Machine Learning problems

<table>
<thead>
<tr>
<th>Supervised Learning</th>
<th>Unsupervised Learning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Continuous</td>
<td></td>
</tr>
<tr>
<td>regression</td>
<td>dimensionality reduction</td>
</tr>
<tr>
<td>Discrete</td>
<td></td>
</tr>
<tr>
<td>classification or categorization</td>
<td>clustering</td>
</tr>
</tbody>
</table>
Regression looks for a pattern to build a model of the data.

- Predict a continuous value given some input.
- The best model is found minimizing the errors.
- “Best fit line” (linear regression).
- The error is the distance between the actual data and the model data.

\[ \varepsilon_i = y_i - (\hat{\beta}_0 + \hat{\beta}_1 x_i) \]

- error
- output prediction
- real output
Assumptions:

- Data represent whole population
- Error is a random variable
- Constant variance

Other types of regression:

- Logarithmic regression
  \[ Y = a + b \ln(x) \]
- Quadratic regression
  \[ Y = a x^2 + b x + c \]
- Power regression
  \[ Y = a x^b \]
- Exponential regression
  \[ Y = a b^x \]
- Gaussian Mixture Regression (GMR)
  \[ p(x) = \sum_{j=1}^{K} w_j \cdot N(x | \mu_j, \Sigma_j) \]
Regression application: Autonomous Driving

Supervised Learning Regression Problem

- Learning algorithm: gradient descent to minimize errors

**Training:**
- Digitizes the road ahead and records the person’s steering directions
- Regression on relation between road and steering variables -> Neural Network

**Testing:**
- Digitizes the road
- Feeds the image to its neural network
- Measure each steering direction’s confidence

Alvin-system of artificial neural networks
Machine Learning problems

<table>
<thead>
<tr>
<th>Supervised Learning</th>
<th>Unsupervised Learning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Discrete</td>
<td></td>
</tr>
<tr>
<td>classification or categorization</td>
<td>clustering</td>
</tr>
<tr>
<td>Continuous</td>
<td></td>
</tr>
<tr>
<td>regression</td>
<td>dimensionality reduction</td>
</tr>
</tbody>
</table>
A classifier maps from the feature space to a label

- Input: feature vector
- Output: class label

Training goals:
1. Accurate classification of training data
2. Correct classifications have high confidence
3. Classification function is simple
Types of classification

- **Instance-based**: transfer category labels from examples with most similar features
  - *What similarity function?*

- **Linear** classifier: confidence in positive label is a weighted sum of features
  - *What are the weights?*

- **Non-linear** classifier: predictions based on more complex function of features
  - *What form does the classifier take? What parameters?*

- **Generative** classifier: assign to the label that best explains the features (makes features most likely)
  - *What is the probability function and its parameters?*
Components of generalization error

- **Bias**: how much the average model over all training sets differ from the true model?
  - Error due to inaccurate assumptions/simplifications made by the model
- **Variance**: how much models estimated from different training sets differ from each other

- **Underfitting**: model is too “simple” to represent all the relevant class characteristics
  - High bias and low variance
  - High training error and high test error

- **Overfitting**: model is too “complex” and fits irrelevant characteristics (noise) in the data
  - Low bias and high variance
  - Low training error and high test error
Classification issues

- Models with **too few** parameters are inaccurate because of a large bias (not enough flexibility)

- Models with **too many** parameters are inaccurate because of a large variance (too much sensitivity to the sample)
Classification error

\[ E(\text{MSE}) = \text{noise}^2 + \text{bias}^2 + \text{variance} \]

- Unavoidable error
- Error due to incorrect assumptions
- Error due to variance of training samples

See the following for explanations of bias-variance (also Bishop’s “Neural Networks” book):

- [http://www.inf.ed.ac.uk/teaching/courses/mlsc/Notes/Lecture4/BiasVariance.pdf](http://www.inf.ed.ac.uk/teaching/courses/mlsc/Notes/Lecture4/BiasVariance.pdf)
Underfitting / Overfitting

- **Underfitting**
- **Overfitting**

**Graph:**
- **Error** axis
- **Complexity** axis
- Red line: **Test error**
- Blue line: **Training error**

**High Bias**
- Low Variance

**Low Bias**
- High Variance
Number of training examples

- Few training examples
  - Graph showing decreasing test error with increasing complexity
- Many training examples
  - Graph showing decreasing test error with increasing complexity

Axes:
- Test Error
- Complexity
  - X-axis: High Bias, Low Variance to Low Bias, High Variance
  - Y-axis: High Bias, Low Variance to Low Bias, High Variance
Generalization error

Fixed prediction model

Error

Generalization Error

Training

Testing

Number of Training Examples
Many classifiers to choose from

- SVM
- Neural networks
- Naïve Bayes
- Bayesian network
- Logistic regression
- Randomized Forests
- Boosted Decision Trees
- K-nearest neighbor
- RBMs
- Etc.

Which is the best one?
No classifier is inherently better than any other: you need to make assumptions to generalize.

Three kinds of error:
- Inherent: unavoidable
- Bias: due to over-simplifications
- Variance: due to inability to perfectly estimate parameters from limited data.

No free-lunch theorem.
- Assign label of **nearest training data** point to each test data point
- **Instance-based** learning
- **Simplest** classifier (good one to try first)
K-nearest neighbor
1-nearest neighbor
3-nearest neighbor
5-nearest neighbor
Find a *linear function* to separate the classes:

\[ f(x) = \text{sgn}(w \cdot x + b) \]
Find a *linear function* to separate the classes:

\[ f(x) = \text{sgn}(w \cdot x + b) \]
Find a *linear function* to separate the classes:

\[ f(x) = \text{sgn}(w \cdot x + b) \]
Datasets that are linearly separable work out great:

But what if the dataset is just too hard?

We can map it to a higher-dimensional space:
General idea: the original input space can always be mapped to some higher-dimensional feature space where the training set is separable:

$$\Phi: \mathbf{x} \rightarrow \varphi(\mathbf{x})$$
People detection in images
Histogram of Oriented Gradients descriptor + Support Vector Machine

descriptor = [..., ...., ....]

SVM

Person  No person

People detection in images

Histogram of Oriented Gradients descriptor

- **Image gradients** computation

- **Histogram of gradient orientations**

---

Diapositiva 1: People detection in images

Histograme of Oriented Gradients descriptor

Immagine di input

- Finestra di detection
- Normalizzazione gamma
- Calcolo gradienti
- Voto pesato in celle spaziali e delle orientazioni
- Normalizzazione su celle spaziali parzialmente sovraposte
- Raccolta HOGs su tutta la finestra di detection

People detection in images

Sliding window search

Sliding windows

Multi-scale search

- KNN with $K = 15$

- Linear SVM trained with
  - Positive ex: about $10^3$
  - Negative ex: $10^4$
- Multiple instances for the same person
- Non maximum suppression $\rightarrow$ Mean Shift
People detection in images
Histogram of Oriented Gradients descriptor + Support Vector Machine

AVERAGE GRADIENT IMAGE OF TRAINING SET

FALSE POSITIVES

FALSE NEGATIVES
Unfortunately, there is no “definitive” multi-class SVM formulation.

In practice, we have to obtain a multi-class SVM by combining multiple two-class SVMs.

One vs. others
- Training: learn an SVM for each class vs. the others
- Testing: apply each SVM to test example and assign to it the class of the SVM that returns the highest decision value

One vs. one
- Training: learn an SVM for each pair of classes
- Testing: each learned SVM “votes” for a class to assign to the test example
SVMs: Pros and cons

Pros

- Many publicly available SVM packages: [http://www.kernel-machines.org/software](http://www.kernel-machines.org/software)
- Kernel-based framework is very powerful, flexible
- SVMs work very well in practice, even with very small training sample sizes

Cons

- No “direct” multi-class SVM, must combine two-class SVMs
- Computation, memory
  - During training time, must compute matrix of kernel values for every pair of examples
  - Learning can take a very long time for large-scale problems
What to remember about classifiers

- **No free lunch**: machine learning algorithms are tools, not dogmas
- Try **simple** classifiers first
- Better to have **smart features and simple classifiers** than simple features and smart classifiers
- Use increasingly powerful classifiers with **more training data** (bias-variance tradeoff)
Simple classifiers combined to form a **robust** classifier

\[ F(x) = \alpha_1 f_1(x) + \alpha_2 f_2(x) + \alpha_3 f_3(x) + \ldots \]

- **Weak** classifiers perform slightly better than chance
Boosting

Initial uniform weight on training examples

Incorrect classifications re-weighted more heavily

Final classifier is weighted combination of weak classifiers

\[ H(x) = \text{sign}(\alpha_1 h_1(x) + \alpha_2 h_2(x) + \alpha_3 h_3(x)) \]
The AdaBoost Algorithm

Given: \((x_1, y_1), \ldots, (x_m, y_m); x_i \in \mathcal{X}, y_i \in \{-1, +1\}\)
Initialise weights \(D_1(i) = 1/m\)
For \(t = 1, \ldots, T:\)

- Find \(h_t = \arg \min_{h_j \in \mathcal{H}} \epsilon_j = \sum_{i=1}^{m} D_t(i) [y_i \neq h_j(x_i)]\)
- If \(\epsilon_t \geq 1/2\) then stop
- Set \(\alpha_t = \frac{1}{2} \log \left(\frac{1 - \epsilon_t}{\epsilon_t}\right)\)
- Update

\[
D_{t+1}(i) = \frac{D_t(i) \exp(-\alpha_t y_i h_t(x_i))}{Z_t}
\]

where \(Z_t\) is normalisation factor

Output the final classifier:

\[
H(x) = \text{sign} \left( \sum_{t=1}^{T} \alpha_t h_t(x) \right)
\]
The AdaBoost Algorithm

Given: \((x_1, y_1), \ldots, (x_m, y_m); x_i \in \mathcal{X}, y_i \in \{-1, +1\}\)
Initialise weights \(D_1(i) = 1/m\)
For \(t = 1, \ldots, T:\)

- Find \(h_t = \arg \min_{h_j \in \mathcal{H}} \epsilon_j = \sum_{i=1}^m D_t(i)[y_i \neq h_j(x_i)]\)
- If \(\epsilon_t \geq 1/2\) then stop
- Set \(\alpha_t = \frac{1}{2} \log(\frac{1-\epsilon_t}{\epsilon_t})\)
- Update

\[
D_{t+1}(i) = \frac{D_t(i) \exp(-\alpha_t y_i h_t(x_i))}{Z_t}
\]

where \(Z_t\) is normalisation factor

Output the final classifier:

\[
H(x) = \text{sign} \left( \sum_{t=1}^T \alpha_t h_t(x) \right)
\]
The AdaBoost Algorithm

Given: \((x_1, y_1), \ldots, (x_m, y_m); x_i \in \mathcal{X}, y_i \in \{-1, +1\}\)

Initialise weights \(D_1(i) = 1/m\)

For \(t = 1, \ldots, T:\)

- Find \(h_t = \arg \min_{h_j \in \mathcal{H}} \epsilon_j = \sum_{i=1}^{m} D_t(i)[y_i \neq h_j(x_i)]\)
- If \(\epsilon_t \geq 1/2\) then stop
- Set \(\alpha_t = \frac{1}{2} \log(\frac{1-\epsilon_t}{\epsilon_t})\)
- Update

\[
D_{t+1}(i) = \frac{D_t(i) \exp(-\alpha_t y_i h_t(x_i))}{Z_t}
\]

where \(Z_t\) is normalisation factor

Output the final classifier:

\[
H(x) = \text{sign} \left( \sum_{t=1}^{T} \alpha_t h_t(x) \right)
\]
The AdaBoost Algorithm

Given: \((x_1, y_1), \ldots, (x_m, y_m)\); \(x_i \in \mathcal{X}, y_i \in \{-1, +1\}\)

Initialise weights \(D_1(i) = 1/m\)

For \(t = 1, \ldots, T\):

- Find \(h_t = \arg \min_{h_j \in \mathcal{H}} \epsilon_j = \sum_{i=1}^{m} D_t(i)[y_i \neq h_j(x_i)]\)
- If \(\epsilon_t \geq 1/2\) then stop
- Set \(\alpha_t = \frac{1}{2} \log\left(\frac{1}{\epsilon_t}\right)\)
- Update

\[
D_{t+1}(i) = \frac{D_t(i) \exp(-\alpha_t y_i h_t(x_i))}{Z_t}
\]

where \(Z_t\) is normalisation factor

Output the final classifier:

\[
H(x) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)
\]
The AdaBoost Algorithm

Given: \((x_1, y_1), \ldots, (x_m, y_m); x_i \in \mathcal{X}, y_i \in \{-1, +1\}\)
Initialise weights \(D_1(i) = 1/m\)
For \(t = 1, \ldots, T:\)
- Find \(h_t = \arg \min_{h_j \in \mathcal{H}} \epsilon_j = \sum_{i=1}^{m} D_t(i)[y_i \neq h_j(x_i)]\)
- If \(\epsilon_t \geq 1/2\) then stop
- Set \(\alpha_t = \frac{1}{2} \log\left(\frac{1-\epsilon_t}{\epsilon_t}\right)\)
- Update
  \[D_{t+1}(i) = \frac{D_t(i) \exp(-\alpha_t y_i h_t(x_i))}{Z_t}\]
  where \(Z_t\) is normalisation factor
Output the final classifier:
\[H(x) = \text{sign} \left( \sum_{t=1}^{T} \alpha_t h_t(x) \right)\]
The AdaBoost Algorithm

Given: \((x_1, y_1), \ldots, (x_m, y_m); x_i \in \mathcal{X}, y_i \in \{-1, +1\}\)
Initialise weights \(D_1(i) = 1/m\)

For \(t = 1, \ldots, T:\)

- Find \(h_t = \arg \min_{h_j \in \mathcal{H}} \epsilon_j = \sum_{i=1}^{m} D_t(i)[y_i \neq h_j(x_i)]\)
- If \(\epsilon_t \geq 1/2\) then stop
- Set \(\alpha_t = \frac{1}{2} \log \left(\frac{1-\epsilon_t}{\epsilon_t}\right)\)
- Update

\[
D_{t+1}(i) = \frac{D_t(i) \exp\left(-\alpha_t y_i h_t(x_i)\right)}{Z_t}
\]

where \(Z_t\) is normalisation factor

Output the final classifier:

\[
H(x) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)
\]
The AdaBoost Algorithm

Given: \((x_1, y_1), \ldots, (x_m, y_m); x_i \in \mathcal{X}, y_i \in \{-1, +1\}\)

Initialise weights \(D_1(i) = 1/m\)

For \(t = 1, \ldots, T:\)

- Find \(h_t = \arg\min_{h_j \in \mathcal{H}} \epsilon_j = \sum_{i=1}^{m} D_t(i)[y_i \neq h_j(x_i)]\)
- If \(\epsilon_t \geq 1/2\) then stop
- Set \(\alpha_t = \frac{1}{2} \log \left( \frac{1-\epsilon_t}{\epsilon_t} \right)\)
- Update

\[
D_{t+1}(i) = \frac{D_t(i) \exp(-\alpha_t y_i h_t(x_i))}{Z_t}
\]

where \(Z_t\) is normalisation factor

Output the final classifier:

\[
H(x) = \text{sign} \left( \sum_{t=1}^{T} \alpha_t h_t(x) \right)
\]
The AdaBoost Algorithm

Given: \((x_1, y_1), \ldots, (x_m, y_m); x_i \in \mathcal{X}, y_i \in \{-1, +1\}\)

Initialise weights \(D_1(i) = 1/m\)

For \(t = 1, \ldots, T:\)

- Find \(h_t = \text{arg min}_{h_j \in \mathcal{H}} \epsilon_j = \sum_{i=1}^{m} D_t(i) \mathbb{1}[y_i \neq h_j(x_i)]\)
- If \(\epsilon_t \geq 1/2\) then stop
- Set \(\alpha_t = \frac{1}{2} \log \left(\frac{1-\epsilon_t}{\epsilon_t}\right)\)
- Update

\[
D_{t+1}(i) = \frac{D_t(i) \exp(-\alpha_t y_i h_t(x_i))}{Z_t}
\]

where \(Z_t\) is normalisation factor

Output the final classifier:

\[
H(x) = \text{sign} \left( \sum_{t=1}^{T} \alpha_t h_t(x) \right)
\]
Boosting application: Face detection

- Haar features
- Adaboost classifier
- Classifiers cascade

- Four basic types
- Easy to compute
- White areas subtracted to black areas
- Integral images allow fast computation of these features

Table of the sums of the intensity values from the top-left corner

The sum of intensity values within one rectangle can be then computed with four accesses to the integral image.

Fast computation of pixel sums

Figure 3: The sum of the pixels within rectangle $D$ can be computed with four array references. The value of the integral image at location 1 is the sum of the pixels in rectangle $A$. The value at location 2 is $A + B$, at location 3 is $A + C$, and at location 4 is $A + B + C + D$. The sum within $D$ can be computed as $4 + 1 - (2 + 3)$.

Feature are extracted from image sub-windows

- base dimension of the sub-windows: 24x24 pixels
- the four types of features are scaled and shifted in all possible combinations
  - In a 24x24 sub-window there are ~160,000 possible features

Most important features

Cascade of classifiers

- a degenerate binary tree
- it allows a very fast classification
- for every window, a feature at a time is computed and the window is rejected or the following feature is computed

- Approximation of **discrete functions** by a decision tree.
- **Inner nodes** (among which the root node)
  - A question is asked about data
  - One child node per possible answer
- **Leaf nodes**
  - Correspond to the **decision** to take (or conclusion to make) if reached
- **Ex:** A decision tree for “play tennis”
A RF consists of many decision trees
Output: mode of outputs by individual trees
Very accurate if trained with many examples
Resistance to outliers and over training
Fast body pose estimation from depth images

- DL: NN with **many layers**
- Each layer **combines** patches from previous layers to create robust features
- Each layer applies a **non-linear** function
- NN VS DL: DL has more hidden layers, thus it can learn more complex features
- Input data need to have **neighborhood regularity** (e.g. images)
- The layers are trained one-by-one while freezing the others
- **Long training** time (GPU processing required)
Deep Learning: Convolutional Neural Network
Deep Learning: Does it work?

<table>
<thead>
<tr>
<th>Audio</th>
<th>Accuracy</th>
<th></th>
<th>Audio</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>TIMIT Phone classification</td>
<td>79.6%</td>
<td>Prior art (Clarkson et al., 1999)</td>
<td>TIMIT Speaker identification</td>
<td>99.7%</td>
</tr>
<tr>
<td>Stanford Feature learning</td>
<td>80.3%</td>
<td>Stanford Feature learning</td>
<td>100.0%</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Images</th>
<th>Accuracy</th>
<th>Images</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>CIFAR Object classification</td>
<td>74.5%</td>
<td>NORB Object classification</td>
<td>94.4%</td>
</tr>
<tr>
<td>Prior art (Yu and Zhang, 2010)</td>
<td>75.5%</td>
<td>Prior art (Ranzato et al., 2009)</td>
<td>96.2%</td>
</tr>
<tr>
<td>Stanford Feature learning</td>
<td></td>
<td>Stanford Feature learning</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Video</th>
<th>Accuracy</th>
<th>Video</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>UCF activity classification</td>
<td>86%</td>
<td>Hollywood2 classification</td>
<td>47%</td>
</tr>
<tr>
<td>Prior art (Kalser et al., 2008)</td>
<td></td>
<td>Prior art (Laptev, 2004)</td>
<td></td>
</tr>
<tr>
<td>Stanford Feature learning</td>
<td>87%</td>
<td>Stanford Feature learning</td>
<td>50%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Multimodal (audio/video)</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>AVLetters Lip reading</td>
<td>Accuracy</td>
</tr>
<tr>
<td>Prior art (Zhao et al., 2009)</td>
<td>58.9%</td>
</tr>
<tr>
<td>Stanford Feature learning</td>
<td>63.1%</td>
</tr>
</tbody>
</table>
Different classifiers require different amounts of training data to work well.

Many techniques generate also synthetic training data.
- When problems are difficult and you have lot of training data, try machine learning

- Try more than one classifier

- Good training => good classification

- Simple is better. Do you need more? Combine many simple classifiers (boosting, deep networks, etc.)
Any questions?

matteo.munaro@dei.unipd.it