Classification - Visual Recognition

1. Introduction
2. Neural Nets
3. Supervised learning
4. SVM classifiers
5. Datasets and evaluation
Supervised learning

Loss functions
Optimization framework: Risk&ERM
Generalization
Constraints for optimization
Gradient descent formal algo

=> all done in course
Classification pipeline

To summarize:

• Theory: Risk minimization, Regularization, Generalization
• Supervised Learning, Learning from examples: ERM
  – To be explained: training/validation/test sets
• Algos:
  – Neural Nets, Deep architectures
  – (linear/kernel-based) SVM classifiers
  – k Nearest Neighbors
  – Learning binary / Multiclass classifiers
Classification- Visual Recognition

1. Introduction
2. NNs
3. Supervised learning
4. SVM classifiers
5. Datasets and evaluation
SVM

Notations:

• Image/Patterns $\mathbf{x} \in \mathbf{X}$

• $\Phi$: function transforming the patterns into feature vectors $\Phi(x)$

• $\langle \cdot, \cdot \rangle$ dot product in the feature space endowed by $\Phi(\cdot)$

• Classes $y = \pm 1$

Early kernel classifiers derived from the perceptron [Rosenblatt58]:

• taking the sign of a linear discriminant function:

$$f(\mathbf{x}) = \langle \mathbf{w}, \Phi(\mathbf{x}) \rangle + b$$

• Classifiers called $\Phi$-machines
SVM

• **Question**: how to find/estimate \( f \)?
  
  – Feature function \( \Phi \) usually hand-chosen for each problem
  
  – Several \( \Phi \) for image processing like BoW
  
  – \( w \) and \( b \): parameters to be determined

\[
f(x) = \langle w, \Phi(x) \rangle + b
\]

• Learning algorithm on a set of training examples:
\[
\mathcal{A} = (x_1, y_1) \cdots (x_n, y_n)
\]
Which hyperplane? $w \cdot x + b$?
SVM optimization: maximizing the margin between + and -

Def.: Margin = distance between the hyperplanes $f(x) = 1$ and $f(x) = -1$ (dashed lines in Figure).

Intuitively, a classifier with a larger margin is more robust to fluctuations.

Hard Margin

Final expression for the Hard Margin SVM optimization:

$$\min_{w,b} \ P(w, b) = \frac{1}{2} \|w\|^2 \quad \text{with} \quad \forall \ i \quad y_i f(x_i) \geq 1$$
SVM

• Hard Margin: OK if data are linearly separated

• Otherwise: noisy data (in red) disrupt the optim.

• Solution: Soft SVM
SVM: Soft Margin

Introducing the slack variables $\xi_i$, one usually gets rid of the inconvenient max of the loss and rewrite the problem as

$$\min_{w,b} P(w, b) = \frac{1}{2}\|w\|^2 + C \sum_{i=1}^{n} \xi_i \quad \text{with} \quad \left\{ \begin{array}{l} \forall i \quad y_i f(x_i) \geq 1 - \xi_i \\ \forall i \quad \xi_i \geq 0 \end{array} \right.$$  

For very large values of the hyper-parameter $C$, **Hard Margin** case:
- Minimization of $\|w\|$ (ie margin maximization) under the constraint that all training examples are correctly classified with a loss equal to zero.

**Smaller values of C relax this constraint: Soft Margin** case
- SVMs that produces markedly better results on noisy problems.
SVM learning scheme

Equivalently, minimizing the following objective function in feature space with the hinge loss function:

$$\ell(y_i f(x_i)) = \max(0, 1 - y_i f(x_i))$$

$$\min_{w,b} P(w, b) = \frac{1}{2}\|w\|^2 + C \sum_{i=1}^{n} \ell(y_i f(x_i))$$
Learning SVMs: Primal/Dual

• In practice: Convex optimization problem
  
  – Primal optimization: $f(x) = \langle w, \Phi(x) \rangle + b$
  
  – Dual optimization: learning SVMs can be achieved by solving the dual of this convex optimization problem

• Dual (using Lagragian and $k(x_i, x_j) = \langle \Phi(x_i), \Phi(x_j) \rangle$):
  
  – For Hard Margin:
    
    $$\max_{\alpha} \mathcal{L}(\alpha) = \sum_{i=1}^{\alpha_i} \alpha_i - \frac{1}{2} \sum_{i,j} y_i y_j \alpha_i \alpha_j k(x_i, x_j) \quad \text{with} \quad \begin{cases} \sum_i \alpha_i y_i = 0 \\ 0 \leq \alpha_i \end{cases}$$

  – For Soft Margin:
    
    $$\max_{\alpha} \mathcal{L}(\alpha) = \sum_{i=1}^{\alpha_i} \alpha_i - \frac{1}{2} \sum_{i,j} y_i y_j \alpha_i \alpha_j k(x_i, x_j) \quad \text{with} \quad \begin{cases} \sum_i \alpha_i y_i = 0 \\ 0 \leq \alpha_i \leq C \end{cases}$$
SVM optimization

Standard equivalent formulation without enforcing $\alpha_i$ to be positive:

- Optimization on coefficients $\alpha_i$ of the SVM kernel expansion $f(x) = \sum_{i=1}^{n} \alpha_i k(x, x_i) + b$ by defining the dual objective function:

$$D(\alpha) = \sum_i \alpha_i y_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j k(x_i, x_j)$$

- and solving the SVM dual *Quadratic Programming* (QP) problem.

$$\max_{\alpha} D(\alpha) \text{ with } \begin{cases} \sum_i \alpha_i = 0 \\ A_i \leq \alpha_i \leq B_i \\ A_i = \min(0, C y_i) \\ B_i = \max(0, C y_i) \end{cases}$$
Classification pipeline

To summarize on SVM:
Support Vector Machines (SVM) defined by three incremental steps:

1. [Vapnik63]: linear classifier / separates the training examples with the **widest** margin => Optimal Hyperplane
Support Vector Machines (SVM) defined by three incremental steps:

1. [Vapnik63]: linear classifier / separates the training examples with the widest margin => Optimal Hyperplane

2. [Guyon93] Optimal Hyperplane built in the feature space induced by a kernel function
Support Vector Machines (SVM) defined by three incremental steps:

1. [Vapnik63]: linear classifier / separates the training examples with the widest margin => Optimal Hyperplane

2. [Guyon93] Optimal Hyperplane built in the feature space induced by a kernel function

3. [Cortes95] soft version: noisy problems addressed by allowing some examples to violate the margin constraint
Extra: Solving SVM

• Min P or Max D
  => QP (Quadratic programing) family optimization
• Good news: efficient batch numerical algorithms have been developed to solve the specific SVM QP problem (hinge loss, convex objective,…)

• Some strategies (exploiting specif.):
  – Conjugate Gradient method [Vapnik]
  – Sequential Minimal Optimization (SMO) [platt].
• In both methods successive searches along well chosen directions

• Some famous SVM solvers like SVMLight [Joachims] or SVMTorch propose to use decomposition algorithms to define such directions
• SVMstruct (for structured outputs)
• State-of-the-art implementation of SMO: [libsvm] => used in tutorials
• LibLinear bib for primal optim (with MATLAB)
Extra: SMO algo for SVM optimization

1. Set $\alpha \leftarrow 0$ and compute the initial gradient $g$ of $D(\alpha)$

2. Choose a $\tau$-violating pair(*) $(i, j)$ Stop if no such pair exists

3. $\lambda \leftarrow \min \left\{ \frac{g_i - g_j}{k_{ii} + k_{jj} - 2k_{ij}}, B_i - \alpha_i, \alpha_j - A_j \right\}$

4. $\alpha_i \leftarrow \alpha_i + \lambda$, $\alpha_j \leftarrow \alpha_j - \lambda$

5. $g_s \leftarrow g_s - \lambda(k_{is} - k_{js}) \quad \forall s \in \{1 \ldots n\}$

6. Return to step 2

(*) pairs in +1/-1 with significant diff of gradients
A ways to easily satisfy the null sum coeff constraint
Classification pipeline

Image → Local descriptors → Visual codes → Image signature → SVM → Class label

- Feature extraction
- Feature coding
- Pooling
Classification- Visual Recognition

1. Introduction
2. NNs
3. Supervised learning
4. SVM classifiers
5. Datasets and evaluation
Datasets for learning/testing

• How to define a category?
  – Bicycle
  – Paintings with women
  – Portraits
...

  Concepts, semantics, ontologies ...
Image/video datasets for training/testing

Choice of the categories (objects, concepts)
- Number of categories
- Number of images per category
- Hierarchical structure ?
- Mono-label/multi-labels
- Pre-processing
  - Color, resolution, centered ...
Example: ImageNet dataset

- Large Scale Visual Recognition Challenge (ILSVRC)
  - 1,2 Million images, 1000 classes

- Paper:
  - ImageNet: A Large-Scale Hierarchical Image Database, Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li and Li Fei-Fei, CVPR 2009
Classes of ImageNet

- Based on WordNet
  - Each node is depicted by images

- A knowledge ontology
  - Taxonomy
  - Partonomy

Website: ImageNet

ImageNet is an image database organized according to the WordNet hierarchy (currently only the nouns), in which each node of the hierarchy is depicted by hundreds and thousands of images. Currently we have an average of over five hundred images per node. We hope ImageNet will become a useful resource for researchers, educators, students and all of you who share our passion for pictures.

Click here to learn more about ImageNet, Click here to join the ImageNet mailing list.
Constructing ImageNet

• 2–step process

Step 1:
Collect candidate images Via the Internet

Step 2:
Clean up candidate Images by humans

• Still a lot of pbs, biases => ImageNetv2, ...
Benchmarks and evaluation

• Train / test / validation sets
  • Cross-validation
  • Learning hyper parameters

• Evaluation
  • Test Error
  • Accuracy, MAP, confusion matrix, Per-class averaging
  • Significance of the comparison, statistical tests, ...

• Dataset building, concepts and semantics
  • Data pre-processing, data augmentation
Image/video datasets for training/testing

- Training classifiers on A
- Testing on B: error evaluation
- A and B disjoints!
Training: Cross-validation

Training Set A

Testing Set B

Boat
Street
Protest

Training
Validation
Extra:

Beyond BoW representation
Work on local descriptors

\[ \mathbf{H} = \begin{bmatrix} C_1 & \alpha_{1,1} & \ldots & \alpha_{1,j} & \ldots & \alpha_{1,N} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ C_m & \alpha_{m,1} & \ldots & \alpha_{m,j} & \ldots & \alpha_{m,N} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ C_M & \alpha_{M,1} & \ldots & \alpha_{M,j} & \ldots & \alpha_{M,N} \end{bmatrix} \Rightarrow g: pooling

f: coding

Work on dico

Work on pooling

Work on coding
Pooling: Aggregating projections => global image index

Sum pooling alternative:

- **Max pooling**: keep the max value for the projection for each visual word
  - Relevant for sparse / soft coding: limit noise effect
  - (Partially) Justify by bio-inspired models (cortex)

\[
\alpha_m = g(\alpha_m) = \max_{j=1..N} \alpha_{m,j}
\]

\[
H = \begin{bmatrix}
\alpha_{1,1} & \cdots & \alpha_{1,j} & \cdots & \alpha_{1,N} \\
\vdots & & \vdots & & \vdots \\
\alpha_{m,1} & \cdots & \alpha_{m,j} & \cdots & \alpha_{m,N} \\
\vdots & & \vdots & & \vdots \\
\alpha_{M,1} & \cdots & \alpha_{M,j} & \cdots & \alpha_{M,N}
\end{bmatrix}
\Rightarrow g: pooling

f: coding
Coding: Projection $\Rightarrow$ dictionary

- **soft assignment**
  - Kernel codebook: absolute weight
  - Uncertainty: relative weight
  - Plausibility: absolute weight to 1-nn

Visual Word Ambiguity

J.C. van Gemert, C.J. Veenman, A.W.M. Smeulders, J.M. Geusebroek

PAMI 2010
Soft Coding: kernel

\[ f_{Kernel}(x_j)[m] = K(d(x_j, c_m)) \]

Ex: \( K(x) = \exp(-ax) \)
Soft Coding: uncertainty

\[ f_{unc}(x_j)[m] = \frac{K(d(x_j, c_m))}{\sum_{k=1}^{M} K(d(x_j, c_k))} \]
Soft Coding : plausibility

\[ f_{\text{Plau}}(x_j)[m] = \begin{cases} 
K(d(x_j, c_m)) & \text{if } m = \arg\min_{k \in \{1; M\}} \|x_j - c_k\|^2 \\
0 & \text{otherwise}
\end{cases} \]
### Soft Coding

- Soft vs hard assignment/coding
  - Not a so big gain soft / hard
  - Uncertainty certainly the best strategy

- Semi-soft : excellent tradeoff

\[
\begin{align*}
\mathbf{H} &= \begin{bmatrix}
\alpha_{1,1} & \cdots & \alpha_{1,j} & \cdots & \alpha_{1,N} \\
\vdots & \ddots & \vdots & \ddots & \vdots \\
\alpha_{m,1} & \cdots & \alpha_{m,j} & \cdots & \alpha_{m,N} \\
\vdots & \ddots & \vdots & \ddots & \vdots \\
\alpha_{M,1} & \cdots & \alpha_{M,j} & \cdots & \alpha_{M,N}
\end{bmatrix} \\
\Rightarrow & g: \text{pooling} \\
f: \text{cooding}
\end{align*}
\]
Sparse Coding

- Other approach: **sparse coding**
  - Approximation of each local feature $x_i$ (SIFT) as a lin. combination of a subset of words from the dictionary: $x_i \sim C\alpha_i$
    - $\alpha_i$ weight vectors, $C$ matrix of vectors of the dictionary

\[
\alpha_i = \text{argmin}_{\alpha} \quad L(\alpha, C) \triangleq \|x_i - C\alpha\|_2^2
\]

- Pb: not sparse, many irrelevant values in $M$
- Each $x_i$ should be represented using only a small nb of visual words => sparsity

- Sparse but no locality

\[
\alpha_i = \text{argmin}_{\alpha} \quad L(\alpha, C) \triangleq \|x_i - C\alpha\|_2^2 + \lambda\|\alpha\|_1
\]
Sparse Coding

- Sparse coding vs VQ (hard assignment)
  - VQ: hard coding
  - SC : Sparse Coding : most of $\alpha_i=0$
  - LLC : Local Linear Coding : words representing the feature must be close (locality)

Are these criteria minimizing reconstruction error relevant for image classification purpose?
Aggregating projections => global image index

Where we are:
- Better represent coding/pooling association: work on the whole matrix of clusters/descriptors dependency => combine spatial pooling and sparse coding

Next:
- Work on new descriptors (bio inspired, learned)
- Dictionaries
  - Train the dico (supervised training)
  - Avoiding dico/clustering
    - Kernel similarity on bag of local features
- Exploit spatial image information
Work on local descriptors

\[ H = \begin{bmatrix}
  c_1 & \cdots & c_m \\
  \vdots & \ddots & \vdots \\
  \alpha_{M,1} & \cdots & \alpha_{M,j} & \cdots & \alpha_{M,N}
\end{bmatrix} \Rightarrow g: pooling

\begin{align*}
\alpha_{1,1} & \cdots & \alpha_{1,j} & \cdots & \alpha_{1,N} \\
\vdots & & \vdots & & \vdots \\
\alpha_{m,1} & \cdots & \alpha_{m,j} & \cdots & \alpha_{m,N}
\end{align*}

Work on dico

Work on pooling

Work on coding
Invariance/robustness in BoW pipeline

Stability of the representation:

– Small deformations/transformations in the input space => similar representations
– Large (or unexpected) transformations in the input space => very dissimilar representations
Invariance/robustness in BoW pipeline

Detection

Descriptor

Coding/pooling

- Translation, flip, rotation, scale changes

- Depends on detectors
  - Regular dense (multi-scale) Grid
  - Robust to translation, scale

- Depends on descriptors
  - Sift: invariance to local rotation, to radiometric change

- Depends on dico (=> kmeans)
  - Robust to noise
  - Pooling: Invariant to any shift, permutation in the bag

= ?
Beyond BoW

• SPM: Spatial Pyramid (Lazebnik et al)
  *Geometry in BoW: Pyramid in image space*

• Pyramid Match Kernel (Grauman et al)
  *Pyramid in feature space: Kernel similarity*
SPM Algorithm

1. Extract interest point descriptors (dense scan)
2. Construct visual word dictionary
3. Build spatial histograms
4. Train an SVM

OR

Weak (edge orientations)  OR  Strong (SIFT)
Algorithm

1. Extract interest point descriptors (dense scan)
2. Construct visual word dictionary
3. Build spatial histograms
4. Train an SVM

- Vector quantization
- Usually K-means clustering
- Vocabulary size (16 to 400)
Algorithm

1. Extract interest point descriptors (dense scan)
2. Construct visual word dictionary
3. Build spatial histograms
4. Train an SVM (with specific kernels)
Spatial grouping

$H = \begin{bmatrix}
C_1 & \alpha_{1,1} & \cdots & \alpha_{1,j} & \cdots & \alpha_{1,N} \\
& \vdots & & \vdots & & \vdots \\
& \alpha_{m,1} & \cdots & \alpha_{m,j} & \cdots & \alpha_{m,N} \\
& \vdots & & \vdots & & \vdots \\
& \alpha_{M,1} & \cdots & \alpha_{M,j} & \cdots & \alpha_{M,N} \\
C_M & & & & & \\
\end{bmatrix}$

$\Rightarrow g: pooling$

$f: coding$

$\Rightarrow$ Break global invariance because of fixed pyramid
Beyond Bags of Features: Spatial Pyramid Matching for Recognizing Natural Scene Categories, Lazebnik et al, CVPR 06

Pyramid in image space, quantize features

\[ \Rightarrow \text{Limit the global invariance:} \]

\[ S(\text{image 1}, \text{image 2}) \text{ small} \]
Algorithm

1. Extract interest point descriptors (dense scan)
2. Construct visual word dictionary
3. Build spatial histograms
4. Train an SVM

Total weight (value of pyramid match kernel): $I_3 + \frac{1}{2}(I_2 - I_3) + \frac{1}{4}(I_1 - I_2) + \frac{1}{8}(I_0 - I_1)$
Algorithm

1. Extract interest point descriptors (dense scan)
2. Construct visual word dictionary
3. Build spatial histograms
4. **Train an SVM**  
   ... Based on the kernel Similarity PMK
SPM Article: Results

- 3 Datasets
  - Nb images
  - Nb classes
- SVM multiclass !?!
- Eval protocol:
  - Train/test/val
    - 10 folds => average+standard deviation
    - Average per class
  - Nb of images per class in train (from 5 to 30)
- Parameter optimization
- Comparison to others
5. Experiments

In this section, we report results on three diverse datasets: fifteen scene categories [4], Caltech-101 [3], and Graz [14]. We perform all processing in grayscale, even when color images are available. All experiments are repeated ten times with different randomly selected training and test images, and the average of per-class recognition rates\(^2\) is recorded for each run. The final result is reported as the mean and standard deviation of the results from the individual runs. Multi-class classification is done with a support vector machine (SVM) trained using the one-versus-all rule: a classifier is learned to separate each class from the rest, and a test image is assigned the label of the classifier with the highest response.

\(^2\)The alternative performance measure, the percentage of all test images classified correctly, can be biased if test set sizes for different classes vary significantly. This is especially true of the Caltech-101 dataset, where some of the “easiest” classes are disproportionately large.
A. Experimental Setup

To obtain reliable results, we repeat the experimental process 10 times. We select 10 random subsets from the data to create 10 pairs of train and test data. For each of these pairs we create a codeword vocabulary on the train set. The exact same codeword vocabulary is used by both the codebook and the codeword ambiguity approaches to describe the train and the test set. For classification, we use an SVM with a histogram intersection kernel. Specifically, we use libSVM, and use the built in one-versus-one approach for multi-class classification. We use 10-fold cross-validation on the train set to tune parameters of the SVM and the size $K_{\sigma}$ of the codebook kernel. The classification rate we report is the average of the per-category recognition rates which in turn are averaged over the 10 random test sets.

For image features we follow Lazebnik et al. [14], and use a
Multi-class SVM

... By combining multiple two-class SVMs!

- **One vs. All**
  - Training: learn an SVM for each class vs. all others grouped in 1 class
  - 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
**SPM Article: Results on Caltech101**

### Multi-class classification results (30 training images per class)

<table>
<thead>
<tr>
<th>Level</th>
<th>Weak features (16)</th>
<th>Strong features (200)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Single-level</td>
<td>Pyramid</td>
</tr>
<tr>
<td>0</td>
<td>15.5 ±0.9</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>31.4 ±1.2</td>
<td>32.8 ±1.3</td>
</tr>
<tr>
<td>2</td>
<td>47.2 ±1.1</td>
<td>49.3 ±1.4</td>
</tr>
<tr>
<td>3</td>
<td>52.2 ±0.8</td>
<td><strong>54.0 ±1.1</strong></td>
</tr>
</tbody>
</table>
Caltech101 comparison

Zhang, Berg, Maire & Malik, 2006

![Graph showing performance comparison between different methods and 'our method'.]
Scene category confusions

Difficult indoor images

kitchen  living room  bedroom
## Caltech101 challenges

### Top five confusions

<table>
<thead>
<tr>
<th>class 1 / class 2</th>
<th>class 1 misclassified as class 2</th>
<th>class 2 misclassified as class 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>ketch / schooner</td>
<td>21.6</td>
<td>14.8</td>
</tr>
<tr>
<td>lotus / water lily</td>
<td>15.3</td>
<td>20.0</td>
</tr>
<tr>
<td>crocodile / crocodile head</td>
<td>10.5</td>
<td>10.0</td>
</tr>
<tr>
<td>crayfish / lobster</td>
<td>11.3</td>
<td>9.1</td>
</tr>
<tr>
<td>flamingo / ibis</td>
<td>9.5</td>
<td>10.4</td>
</tr>
</tbody>
</table>

### Easiest and hardest classes

- **Minaret (97.6%)**
- **Windsor chair (94.6%)**
- **Joshua tree (87.9%)**
- **Okapi (87.8%)**
- **Cougar body (27.6%)**
- **Beaver (27.5%)**
- **Crocodile (25.0%)**
- **Ant (25.0%)**

**Sources of difficulty:** lack of texture, camouflage, “thin” objects, highly deformable shape
PMK/SIFT Best Categories (1-5)

- Car: 100%
- Leopard: 100%
- Person: 99.7%
- Instrument: 99.1%
- Building: 98.2%
SPM Article

PMK/SIFT Best Categories (6-10)

97.7%

97.4%

95.7%

95.3%

95.2%
SPM Article

PMK/SIFT 5 Worst Categories

1. Ants (7.7%)
2. Beavers (11.2%)
3. Crabs (11.5%)
4. Birds (11.8%)
5. Anchors (12.3%)
schooner
A fore-and-aft rigged sailing vessel having at least two masts, with a foremast that is usually smaller than the other masts.

ketch
A two-masted fore-and-aft-rigged sailing vessel with a mizzenmast stepped aft of a taller mainmast but forward of the rudder.
SPM Article

PMK/SIFT Most Confused Category Pairs

Gerenuk (antilope girafe ou gérénuk)

kangaroo
PMK/SIFT Most Confused Category Pairs

nautilus

brain
Beyond BoW

- Spatial Pyramid (Lazebnik et al)
  
  *Geometry in BoW: Pyramid in image space*

- Pyramid Match Kernel (Grauman et al)
  
  *Pyramid in feature space: Kernel similarity*
How to Compare Sets of Features?

- Each instance is unordered set of vectors
- Varying number of vectors per instance

\[ X = \{ \vec{x}_1, \ldots, \vec{x}_m \} \]

\[ Y = \{ \vec{y}_1, \ldots, \vec{y}_n \} \]
Correspondence-Based Match

Explicit search for correspondences...

$$\max_{\pi: X \rightarrow Y} \sum_{x_i \in X} S(x_i, \pi(x_i))$$

[Wallraven et al., Lyu, Boughhorbel et al., Belongie et al., Rubner et al., Berg et al., Gold & Rangarajan, Shashua & Hazan,...]
Partial Matching

Compare sets by computing a *partial matching* between their features.

\[
\max_{\pi: X \rightarrow Y} \sum_{x_i \in X} S(x_i, \pi(x_i))
\]
Pyramid Match

Hierarchical fixed dictionary

\[ \mathcal{X} = \{\vec{x}_1, \ldots, \vec{x}_m\} \quad \mathcal{Y} = \{\vec{y}_1, \ldots, \vec{y}_n\} \]

Optimal partial matching

\[ \max_{\pi: \mathcal{X} \to \mathcal{Y}} \sum_{x_i \in \mathcal{X}} S(x_i, \pi(x_i)) \]
Work on dictionary
PMK: Hierarchical dico

\[
\mathbf{H} = \begin{bmatrix}
    c_1 \\
    \vdots \\
    c_m \\
    \vdots \\
    c_M
\end{bmatrix}
\begin{bmatrix}
    x_1 & \cdots & x_j & \cdots & x_N \\
    \alpha_{1,1} & \cdots & \alpha_{1,j} & \cdots & \alpha_{1,N} \\
    \vdots & \vdots & \vdots & \vdots & \vdots \\
    \alpha_{m,1} & \cdots & \alpha_{m,j} & \cdots & \alpha_{m,N} \\
    \vdots & \vdots & \vdots & \vdots & \vdots \\
    \alpha_{M,1} & \cdots & \alpha_{M,j} & \cdots & \alpha_{M,N}
\end{bmatrix}
\Rightarrow g: pooling
\downarrow
f: coding
Pyramid match: main idea

Feature space partitions serve to “match” the local descriptors within successively wider regions.

\( \mathbb{R}^d \)

descriptor space

\( X = \{ \vec{x}_1, \ldots, \vec{x}_m \} \)

\( Y = \{ \vec{y}_1, \ldots, \vec{y}_n \} \)
Pyramid match: main idea

$\mathbb{R}^d$

Histogram intersection counts number of possible matches at a given partitioning.

$$I(H_X, H_Y) = \sum_j \min(H_X(j), H_Y(j))$$

$= 3$

$X = \{x_1, \ldots, x_m\}$

$Y = \{y_1, \ldots, y_n\}$
Pyramid match kernel

\[ K_\Delta(X, Y) = \sum_{i=0}^{L} 2^{-i} \mathcal{I} \left( H_X^{(i)}, H_Y^{(i)} \right) - \mathcal{I} \left( H_X^{(i-1)}, H_Y^{(i-1)} \right) \]

- For similarity, weights inversely proportional to bin size (or may be learned)
- Normalize these kernel values to avoid favoring large sets

Rq: Back to the SPM article: we have the explanation of the SPM kernel too!
Pyramid match kernel

$X = \{\bar{x}_1, \ldots, \bar{x}_m\} \quad Y = \{\bar{y}_1, \ldots, \bar{y}_n\}$

Optimal match: $O(m^3)$
Pyramid match: $O(mL)$

optimal partial matching
Pyramid match recognition on the Caltech-101

Pyramid match kernel (PMK)

Bag-of-words baseline

Recognition accuracy per class (10 runs)

Number of training examples per class
Bow and beyond / similarity

• A Full chain of process from data to labels
  • Geometry via spatial grids and pyramids
  • Coding/pooling: unsupervised learning, one step to hierarchical/deep learning representations
  • Similarity: the kernel side: