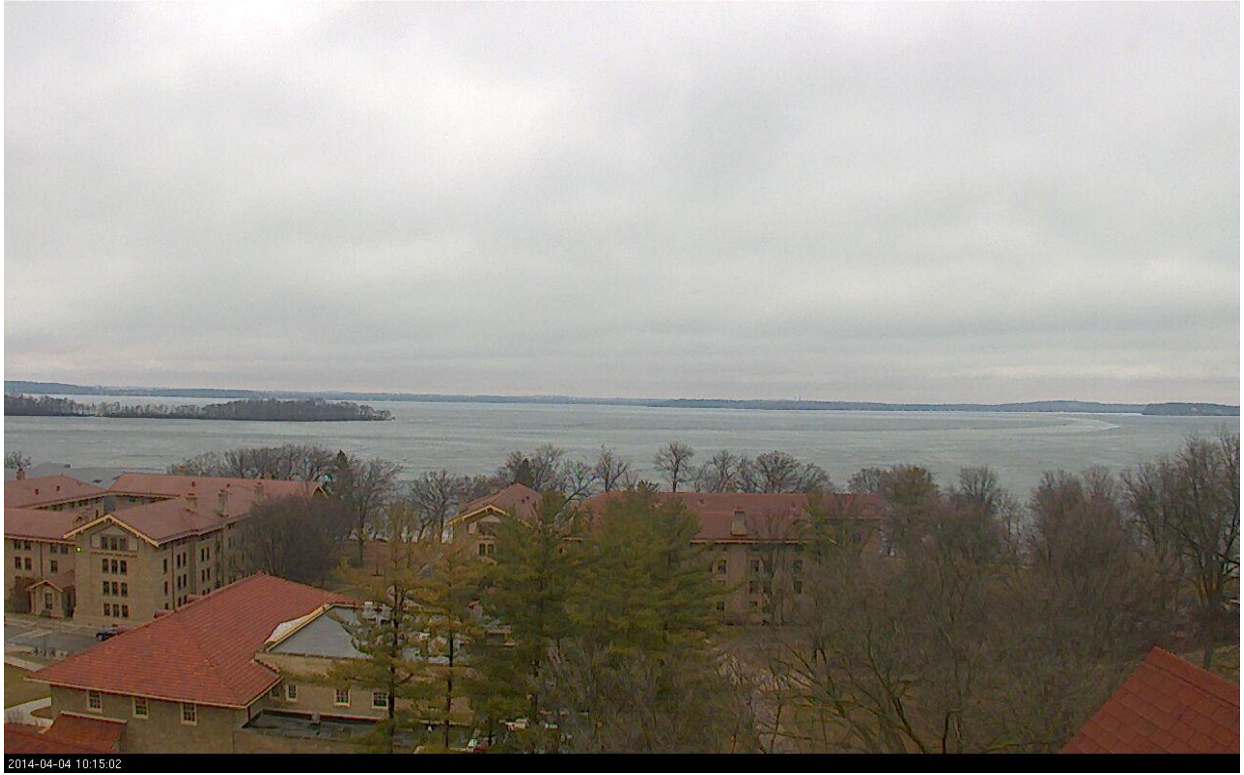
Introduction to Scene Classification

Jia Xu
jiaxu@cs.wisc.edu

Outline

- ➢ Introduction
- ➢ Key Components for Scene Classification
- ➢ Locality-constrained Linear Coding(LLC)
- ➢ Experimental Evaluation
- ➢ Discussion

Human Vision

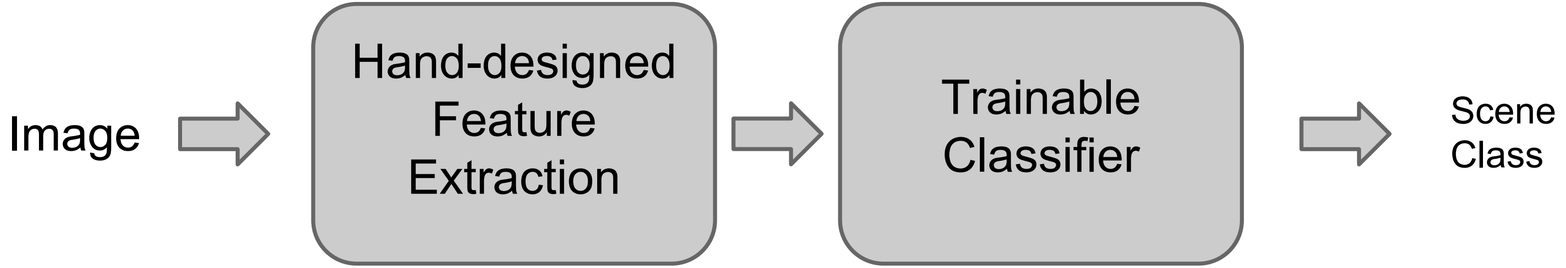


http://alfi.soils.wisc.edu/asig/webcam/big.jpg

What is the scene?
  - office? street? suburb?

How do you recognize this scene?

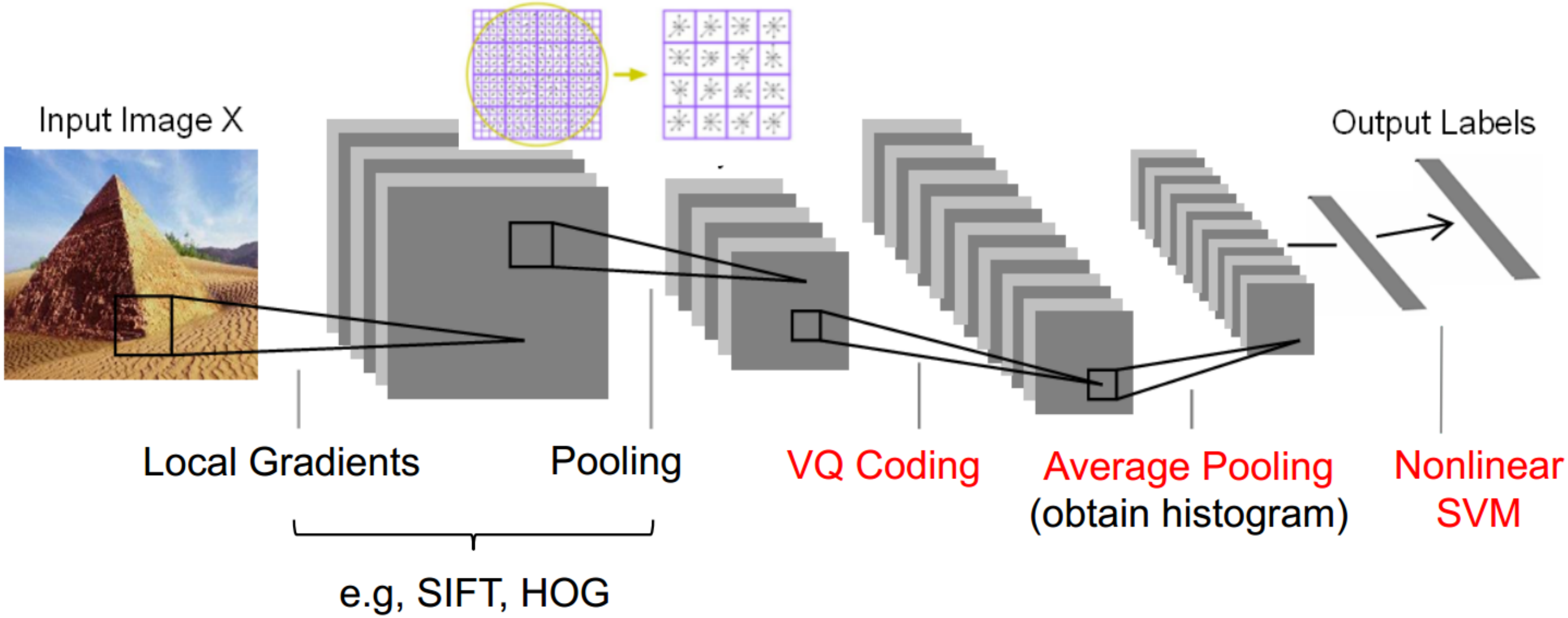
Computer Recognition/Classification



- ➢ Low level features
  - Color histogram, SIFT, HOG, Object Bank, e.t.c
- ➢ Feature engineering
  - Bag of Words, Spatial Pyramid Matching (SPM), Sparse Coding(SC), Locality-constrained Linear Coding(LLC)
- ➢ Classifier
  - SVM (linear, nonlinear, multiple kernel), linear/logistic regression, random forest

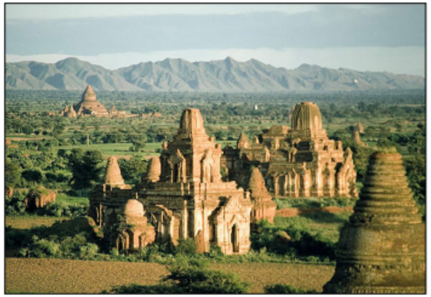
Image Classification Overview

urtasun

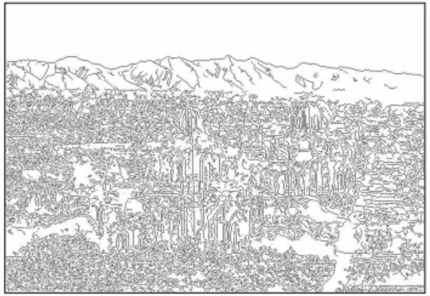


[Source: K. Yu, R. Urtasun]

Feature Extraction



Weak features



Edge points at 2 scales and 8 orientations (vocabulary size 16)

Strong features

SIFT descriptors of 16x16 patches sampled on a regular grid, quantized to form visual vocabulary (size 200, 400)

For each image, we get N feature points: $\mathbf{x}_i$, i=1, ..., N.

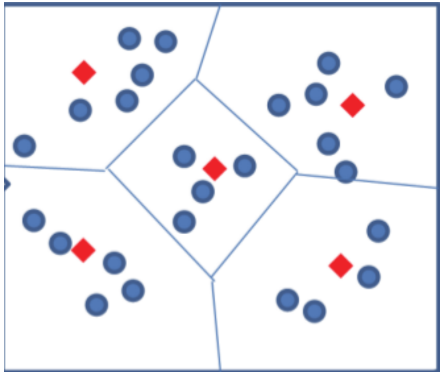
Depending on sample scheme, N can be different for different images

[Source: S. Lazebnik]

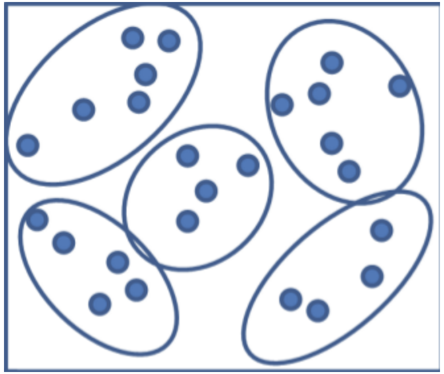
Codebook Generation

[Source: X. Wang et al.]

- K-means
  - partition N features $\{\mathbf{x}_1,\ldots\mathbf{x}_N\}$ into K clusters $\{\mathbf{b}_1,\ldots,\mathbf{b}_K\}$, where $\mathbf{b}_k$ is the center of k-th cluster
  - hard assignment



- Gaussian Mixture Model(GMM)
  - learn K Gaussian mixtures from the feature set $\{\mathbf{x}_1,\ldots\mathbf{x}_N\}$
  - soft assignment



Feature Encoding

- Vector Quantization (VQ)

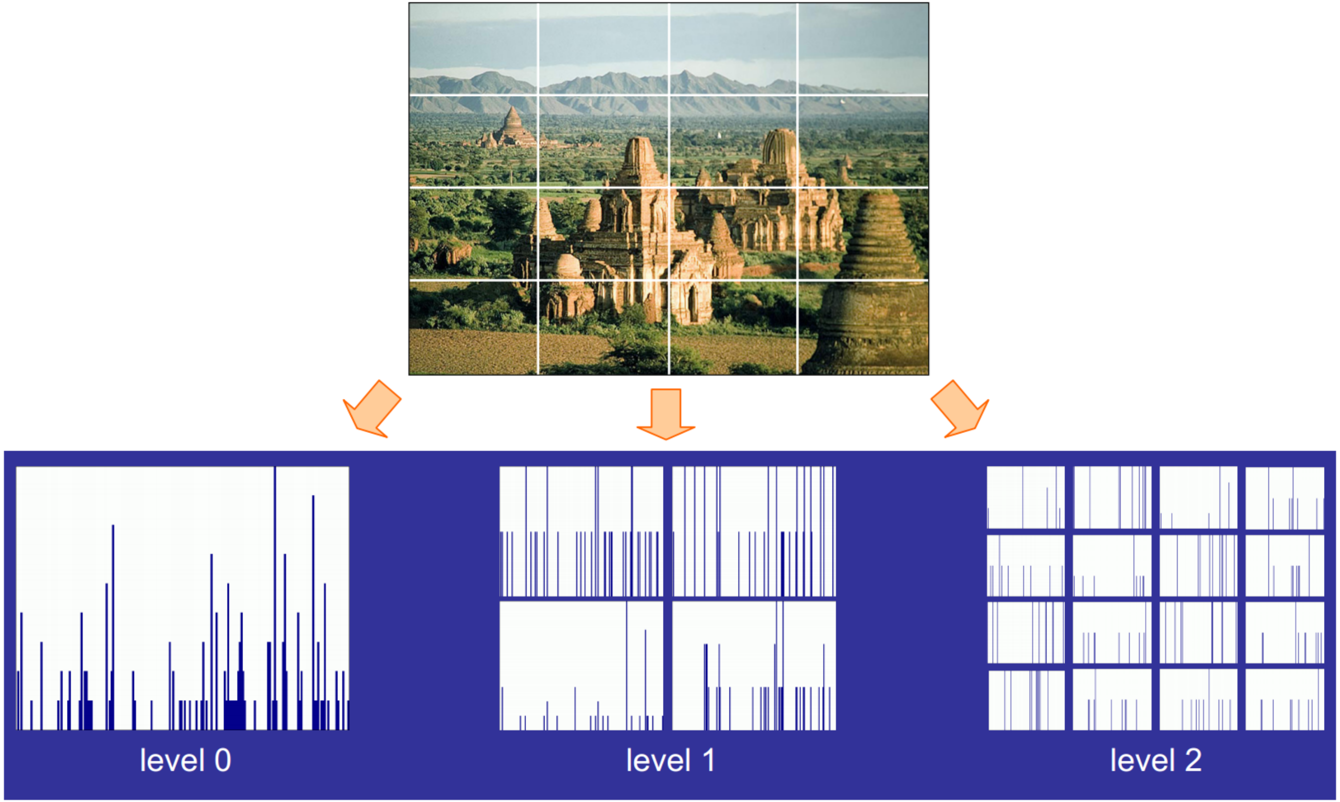
$$c_{nk} = \begin{cases} 1, & \text{if } k = \operatorname{argmin}_k \|\mathbf{x}_n - \mathbf{b}_k\|^2 \\ 0, & \text{otherwise} \end{cases}$$

- Soft-assignment Encoding

$$c_{nk} = \frac{\exp(-\beta \|\mathbf{x}_n - \mathbf{b}_k\|^2)}{\sum_{j=1}^{K} \exp(-\beta \|\mathbf{x}_n - \mathbf{b}_j\|^2)}$$

Spatial context completely ignored!

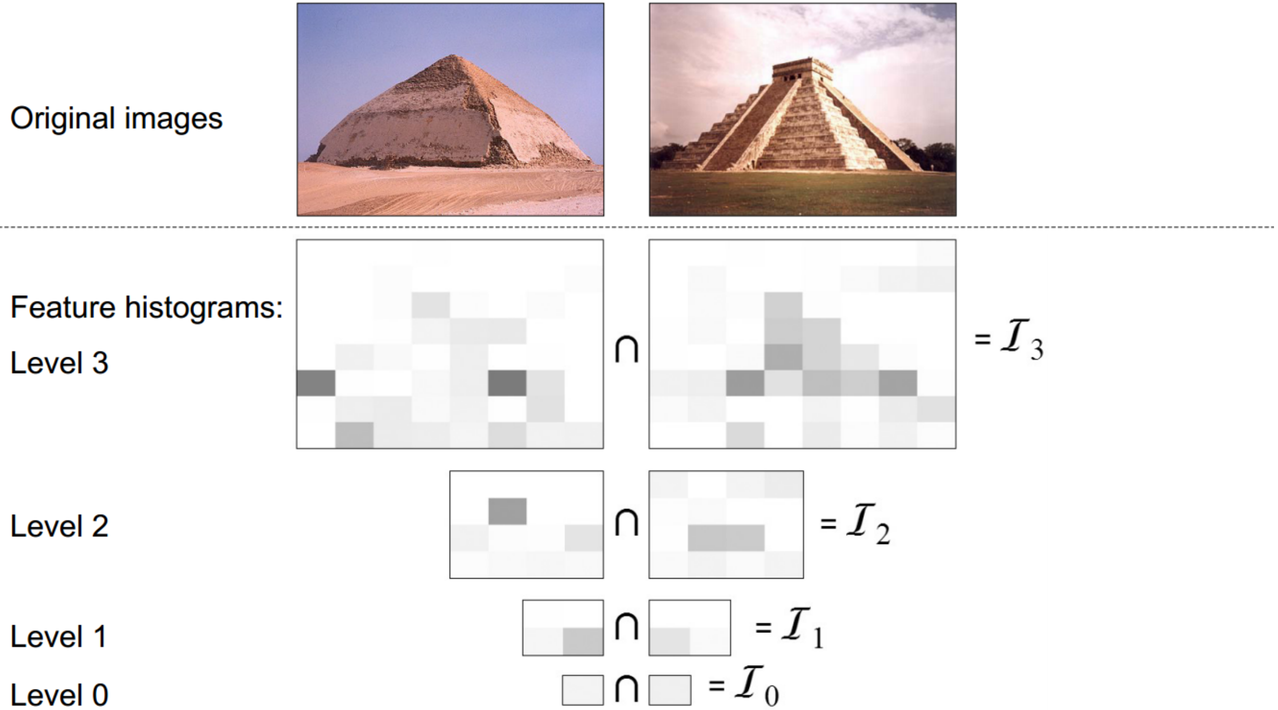
Spatial Pyramid Representation



[Source: S. Lazebnik]

Pyramid Matching

[Source: S. Lazebnik]



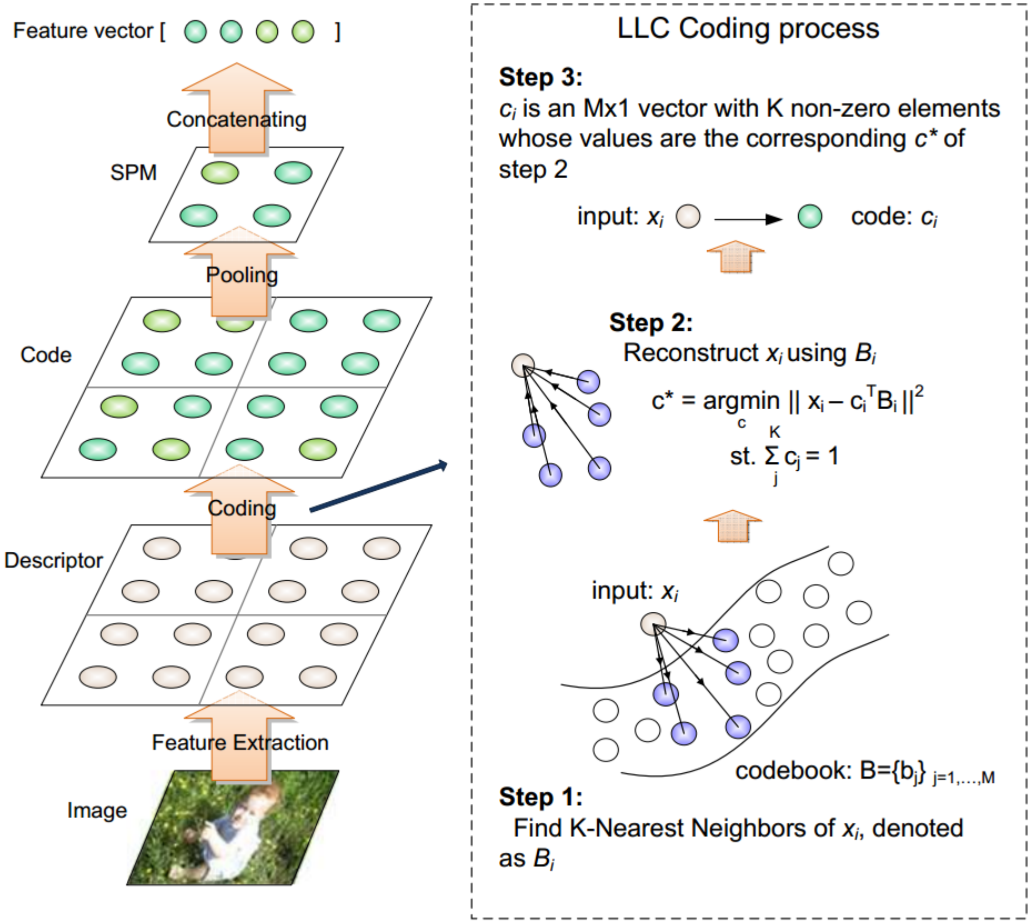
Total weight (value of *pyramid match kernel*): $\mathcal{I}_3 + \frac{1}{2}(\mathcal{I}_2 - \mathcal{I}_3) + \frac{1}{4}(\mathcal{I}_1 - \mathcal{I}_2) + \frac{1}{8}(\mathcal{I}_0 - \mathcal{I}_1)$

Limitations of SPM

- Non-linear SVM is not scalable
- VQ coding may be too coarse
- Average pooling is not optimal

Why not non-linear coding and linear SVM?

LLC



[Source: J. Wang et al.]

SC vs LLC

- ➢ Sparse Coding(SC)

$$\arg\min_{\mathbf{C}} \sum_{i=1}^{N} ||\mathbf{x}_i - \mathbf{B}\mathbf{c}_i||^2 + \lambda ||\mathbf{c}_i||_{\ell^1}$$

- ➢ Locality-constrained Linear Coding(LLC)

$$\min_{\tilde{\mathbf{C}}} \sum_{i=1}^{N} ||\mathbf{x}_i - \tilde{\mathbf{c}}_i \mathbf{B_i}||^2$$

$$st.\ \mathbf{1}^\top \tilde{\mathbf{c}}_i = 1,\ \forall i.$$

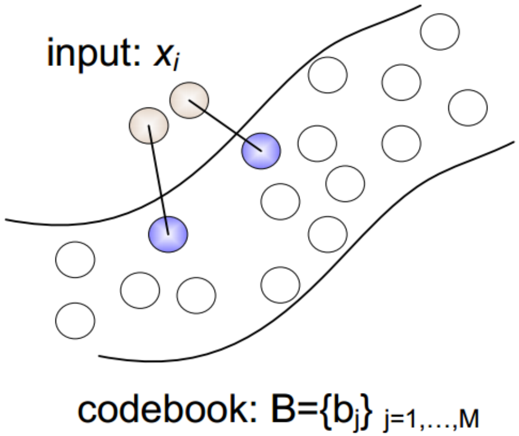
Reconstruct LLC Demo

```
clear; clc; close all;
N = 100;  % feature dimension
K = 5;  % number of nearest neighbours
% construct codebook
B = randn( K, N);
% create truth code
c = randn(K, 1);
c = c /sum(c);
% compute feature
x = B'*c;

% compute data covariance matrix
one = ones(K, 1);
B_1x = B - one *x';
C = B_1x * B_1x';

% reconstruct LLC code
c_hat = C \ one;
c_hat = c_hat /sum(c_hat);
% compute reconstruction error
diff = norm(c-c_hat)
```

Properties of LLC



➢ Better reconstruction
➢ Local smooth sparsity
➢ Analytical solution

[Source: J. Wang et al.]

Pooling and Normalization

- Pooling
  - sum pooling: $\mathbf{c}_{out} = \mathbf{c}_{in1}+,\ldots,+\mathbf{c}_{in2}$
  - max pooling: $\mathbf{c}_{out} = \max(\mathbf{c}_{in1},\ldots,\mathbf{c}_{in2})$
- Normalization
  - sum normalization: $\mathbf{c}_{out} = \mathbf{c}_{in} / \mathrm{sum}(\mathbf{c}_{in})$
  - L2 normalization: $\mathbf{c}_{out} = \mathbf{c}_{in} / \ ||\mathbf{c}_{in}||_2$

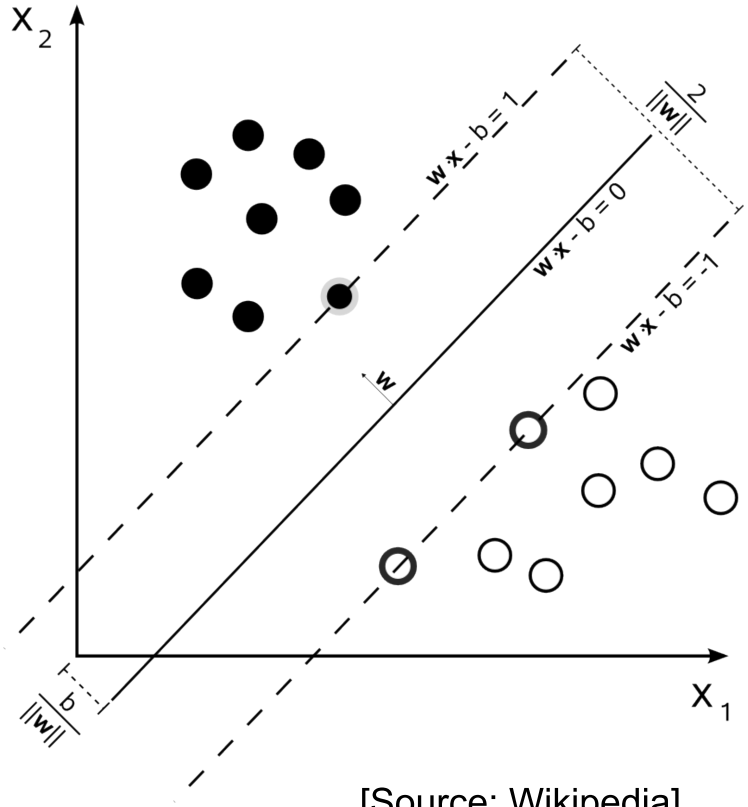
Classification

- Linear SVM
  - Given training data D={ ($\mathbf{x}_i$,y_i) | $\mathbf{x}_i$ is a p dimensional feature vector and y_i = -1 or 1, i=1,...,n}
  - Solve for

    $\text{argmin}_{\mathbf{w},b}\ 1/2\ ||\mathbf{w}||_2$

    subject to (for any i=1,…,n)

    $y_i\ (\mathbf{w}*\mathbf{x}_i - b) >= 1$
  - Various setting are implemented in liblinear



[Source: Wikipedia]

Multi-Class SVM

- Crammer and Singer algorithm
  - already implemented in liblinear
- One-vs-all
  - train C binary classifier ($\mathbf{w}_c$,b_c) for each category c
  - final label for image i is $\text{argmax}_c\ \mathbf{w}_c\mathbf{x}_i+b_c$

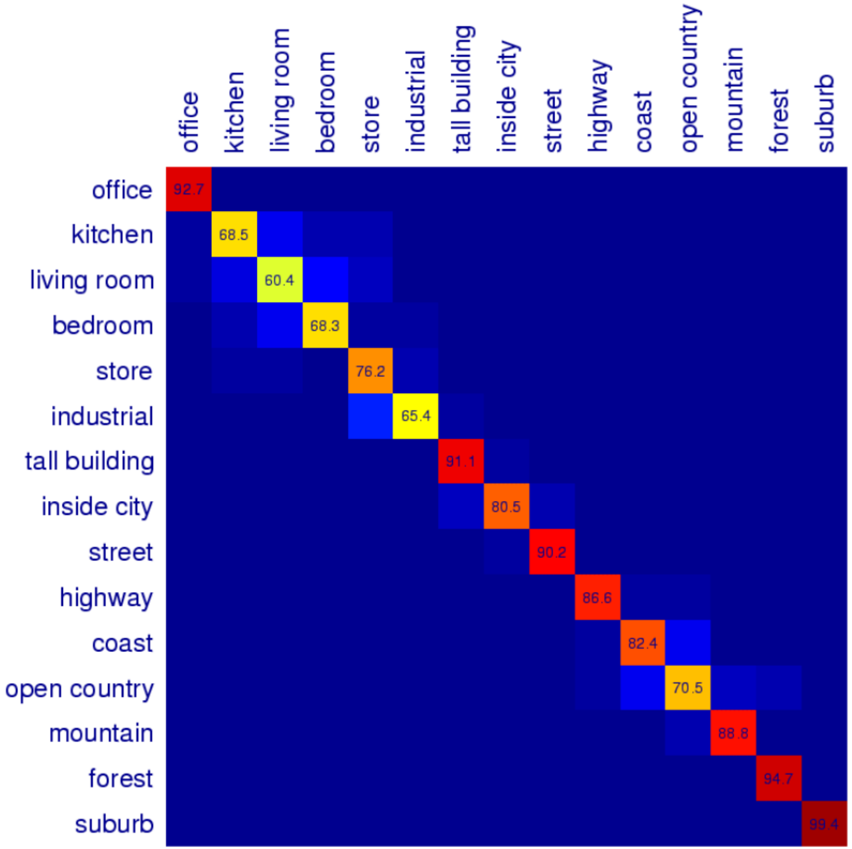
Dataset

- ➢ Scene category dataset from Lazebnik et al.
- ➢ 15 categories
- ➢ Each category has 200 to 400 images
- ➢ Average image size is 300*250

Experimental Evaluation Protocol

- Training with 100 images per class
  - codebook generation and linear SVM
- Testing with the rest images
- Report confusion matrix
- Report mean accuracy (mean of diagonal elements in your confusion matrix)

Confusion Matrix



[Source: S. Lazebnik]

Evaluation Tips

- ➢ Only change one parameter each time, and save your results in a table
- ➢ Important parameters: # of codebook size (start with 1024), # of nearest neighbors (start with 5), -s -C -w in liblinear
- ➢ Grid search

Extensions

- ➢ Codebook Optimization
  - ○ Algorithm 4.1
- ➢ Evaluate your implementation on other applications, e.g. action recognition.
- ➢ Combine LLC with Object Bank

Discussion

- Q & A