CS 1699: Intro to Computer Vision

Bias-Variance Trade-off +
Other Models and Problems

Prof. Adriana Kovashka
University of Pittsburgh
November 3, 2015
Outline

• Support Vector Machines (review + other uses)
• Bias-variance trade-off
• Scene recognition: Spatial pyramid matching
• Other classifiers
  – Decision trees
  – Hidden Markov models
• Other problems
  – Clustering: agglomerative clustering
  – Dimensionality reduction
Let \( w = \begin{bmatrix} a \\ c \end{bmatrix} \) and \( x = \begin{bmatrix} x \\ y \end{bmatrix} \).

\[ ax + cy + b = 0 \]

\[ w \cdot x + b = 0 \]

The distance from point \((x_0, y_0)\) to the line \(ax + cy + b = 0\) is given by:

\[ D = \frac{|ax_0 + cy_0 + b|}{\sqrt{a^2 + c^2}} = \frac{|w^T x + b|}{||w||} \]

\[ \text{distance from point to line} \]
Support vector machines

- Want line that maximizes the margin.

For support vectors, \( x_i \cdot w + b = \pm 1 \)

Distance between point and line:
\[
\frac{|x_i \cdot w + b|}{\|w\|}
\]

For support vectors:
\[
\frac{w^T x + b}{\|w\|} = \frac{\pm 1}{\|w\|} \quad M = \left| \frac{1}{\|w\|} - \frac{-1}{\|w\|} \right| = \frac{2}{\|w\|}
\]

Finding the maximum margin line

1. Maximize margin $2/\|\mathbf{w}\|$
2. Correctly classify all training data points:

   $\mathbf{x}_i$ positive ($y_i = 1$): $\mathbf{x}_i \cdot \mathbf{w} + b \geq 1$

   $\mathbf{x}_i$ negative ($y_i = -1$): $\mathbf{x}_i \cdot \mathbf{w} + b \leq -1$

**Quadratic optimization problem:**

Minimize $\frac{1}{2} \mathbf{w}^T \mathbf{w}$

Subject to $y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1$

One constraint for each training point.

Note sign trick.

Finding the maximum margin line

• Solution: \( w = \sum_i \alpha_i y_i x_i \)
  \[ b = y_i - w \cdot x_i \] (for any support vector)

• Classification function:
  \[ f(x) = \text{sign} \ (w \cdot x + b) \]
  \[ = \text{sign} \left( \sum_i \alpha_i y_i x_i \cdot x + b \right) \]

If \( f(x) < 0 \), classify as negative, otherwise classify as positive.

• Notice that it relies on an inner product between the test point \( x \) and the support vectors \( x_i \)

• (Solving the optimization problem also involves computing the inner products \( x_i \cdot x_j \) between all pairs of training points)

C. Burges, A Tutorial on Support Vector Machines for Pattern Recognition, Data Mining and Knowledge Discovery, 1998
Nonlinear SVMs

- 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:
Examples of kernel functions

- Linear:
  \[ K(x_i, x_j) = x_i^T x_j \]

- Gaussian RBF:
  \[ K(x_i, x_j) = \exp\left(-\frac{||x_i - x_j||^2}{2\sigma^2}\right) \]

- Histogram intersection:
  \[ K(x_i, x_j) = \sum_k \min(x_i(k), x_j(k)) \]
Allowing misclassifications

\[
\min_w \quad \frac{1}{2} \|w\|^2 + C \sum_{i=1}^{N} \xi_i \\
\text{subject to} \quad y_i w^T x_i \geq 1, \quad \forall i = 1, \ldots, N
\]
What about multi-class SVMs?

• 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 the test example, and assign it to 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
Evaluating Classifiers

• Accuracy
  – # correctly classified / # all test examples

• Precision/recall
  – Precision = # predicted true pos / # predicted pos
  – Recall = # predicted true pos / # true pos

• F-measure
  = 2PR / (P + R)

• Want evaluation metric to be in some range, e.g. [0 1]
  – 0 = worst possible classifier, 1 = best possible classifier
Precision / Recall / F-measure

True positives (images *that contain* people)

True negatives (images *do not contain* people)

Predicted positives (images *predicted to contain* people)

Predicted negatives (images *predicted not to contain* people)

- Precision
- Recall
- F-measure

Accuracy: 5 / 10 = 0.5
Support Vector Regression
Regression

- **Regression** is like **classification** except the **labels are real valued**

- **Example applications:**
  - Stock value prediction
  - Income prediction
  - CPU power consumption
Regularized Error Function for Regression

In linear regression, we minimize the error function:

$$\sum_{n=1}^{N} \{ y_n - t_n \}^2 + \frac{\lambda}{2} \|w\|^2$$

An example of $\epsilon$-insensitive error function:

$$E_{\epsilon} = 0 \quad \text{for} \quad |f(x) - y| < \epsilon$$

$$E_{\epsilon} = |f(x) - y| - \epsilon \quad \text{otherwise}$$

Use the $\epsilon$-insensitive error function:

$$C \sum_{n=1}^{N} E_{\epsilon} (y(x_n) - t_n) + \frac{\lambda}{2} \|w\|^2$$

Adapted from Huan Liu
Slack Variables for Regression

For a target point to lie inside the tube:

\[ y_n - \varepsilon \leq t_n \leq y_n + \varepsilon \]

Introduce slack variables to allow points to lie outside the tube:

Subject to:

\[ t_n \leq y(x_n) + \varepsilon + \xi_n \quad \xi_n \geq 0 \]
\[ t_n \geq y(x_n) - \varepsilon - \xi_n^- \quad \xi_n^- \geq 0 \]

Minimize:

\[ C \sum_{n=1}^{N} (\xi_n + \xi_n^-) + \frac{1}{2} \|w\|^2 \]

Adapted from Huan Liu
Next time: Support Vector Ranking
Outline

• Support Vector Machines (review + other uses)
• Bias-variance trade-off
• Scene recognition: Spatial pyramid matching
• Other classifiers
  – Decision trees
  – Hidden Markov models
• Other problems
  – Clustering: agglomerative clustering
  – Dimensionality reduction
Generalization

- How well does a learned model generalize from the data it was trained on to a new test set?
Generalization

• Components of generalization error
  – **Bias**: how much the average model over all training sets differs 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

Slide credit: L. Lazebnik
Bias-Variance Trade-off

- 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).
Fitting a model

\[ y(x, w) = w_0 + w_1 x + w_2 x^2 + \ldots + w_M x^M = \sum_{j=0}^{M} w_j x^j \]

\[ E(w) = \frac{1}{2} \sum_{n=1}^{N} \{ y(x_n, w) - t_n \}^2 \]

Figures from Bishop

**Is this a good fit?**

*Figure 1.4*: Plots of polynomials having various orders \( M \), shown as red curves, fitted to the data set shown in Figure 1.2.
With more training data

Figure 1.6  Plots of the solutions obtained by minimizing the sum-of-squares error function using the $M = 9$ polynomial for $N = 15$ data points (left plot) and $N = 100$ data points (right plot). We see that increasing the size of the data set reduces the over-fitting problem.

Figures from Bishop
Bias-variance tradeoff

- Underfitting
- Overfitting

Error

High Bias
Low Variance

Low Bias
High Variance

Complexity

Slide credit: D. Hoiem
Bias-variance tradeoff

- Many training examples:
  - Low Bias
  - High Variance

- Few training examples:
  - High Bias
  - Low Variance

Test Error

Complexity

Slide credit: D. Hoiem
Choosing the trade-off

- Need validation set
- Validation set is separate from the test set
Effect of Training Size

- Fixed prediction model

![Graph showing error vs. number of training examples with two inset plots for different numbers of training examples (N=15 and N=100).]
How to reduce variance?

• Choose a simpler classifier

• Use fewer features

• Get more training data

• Regularize the parameters
Table 1.1 Table of the coefficients $w^*$ for polynomials of various order. Observe how the typical magnitude of the coefficients increases dramatically as the order of the polynomial increases.

<table>
<thead>
<tr>
<th>$M = 0$</th>
<th>$M = 1$</th>
<th>$M = 6$</th>
<th>$M = 9$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$w_0^*$</td>
<td>0.19</td>
<td>0.82</td>
<td>0.31</td>
</tr>
<tr>
<td>$w_1^*$</td>
<td>-1.27</td>
<td>7.99</td>
<td>232.37</td>
</tr>
<tr>
<td>$w_2^*$</td>
<td>-25.43</td>
<td>-5321.83</td>
<td></td>
</tr>
<tr>
<td>$w_3^*$</td>
<td>17.37</td>
<td>48568.31</td>
<td></td>
</tr>
<tr>
<td>$w_4^*$</td>
<td>-231639.30</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$w_5^*$</td>
<td>640042.26</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$w_6^*$</td>
<td>-1061800.52</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$w_7^*$</td>
<td>1042400.18</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$w_8^*$</td>
<td>-557682.99</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$w_9^*$</td>
<td>125201.43</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 1.2 Table of the coefficients $w^*$ for $M = 9$ polynomials with various values for the regularization parameter $\lambda$. Note that $\ln \lambda = -\infty$ corresponds to a model with no regularization, i.e., to the graph at the bottom right in Figure 1.4. We see that, as the value of $\lambda$ increases, the typical magnitude of the coefficients gets smaller.

<table>
<thead>
<tr>
<th>$\lambda$</th>
<th>$\ln \lambda = -\infty$</th>
<th>$\ln \lambda = -18$</th>
<th>$\ln \lambda = 0$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$w_0^*$</td>
<td>0.35</td>
<td>0.35</td>
<td>0.13</td>
</tr>
<tr>
<td>$w_1^*$</td>
<td>232.37</td>
<td>4.74</td>
<td>-0.05</td>
</tr>
<tr>
<td>$w_2^*$</td>
<td>-5321.83</td>
<td>-0.77</td>
<td>-0.06</td>
</tr>
<tr>
<td>$w_3^*$</td>
<td>48568.31</td>
<td>-31.97</td>
<td>-0.05</td>
</tr>
<tr>
<td>$w_4^*$</td>
<td>-231639.30</td>
<td>-3.89</td>
<td>-0.03</td>
</tr>
<tr>
<td>$w_5^*$</td>
<td>640042.26</td>
<td>55.28</td>
<td>-0.02</td>
</tr>
<tr>
<td>$w_6^*$</td>
<td>-1061800.52</td>
<td>41.32</td>
<td>-0.01</td>
</tr>
<tr>
<td>$w_7^*$</td>
<td>1042400.18</td>
<td>-45.95</td>
<td>-0.00</td>
</tr>
<tr>
<td>$w_8^*$</td>
<td>-557682.99</td>
<td>-91.53</td>
<td>0.00</td>
</tr>
<tr>
<td>$w_9^*$</td>
<td>125201.43</td>
<td>72.68</td>
<td>0.01</td>
</tr>
</tbody>
</table>
Characteristics of vision learning problems

• Lots of continuous features
  – Spatial pyramid may have ~15,000 features

• Imbalanced classes
  – Often limited positive examples, practically infinite negative examples

• Difficult prediction tasks

• Recently, massive training sets became available
  – If we have a massive training set, we want classifiers with low bias (high variance is ok) and reasonably efficient training

Adapted from D. Hoiem
Remember...

• No free lunch: machine learning algorithms are tools
• Three kinds of error
  – Inherent: unavoidable
  – Bias: due to over-simplifications
  – Variance: due to inability to perfectly estimate parameters from limited data
• 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)
Outline

• Support Vector Machines (review + other uses)
• Bias-variance trade-off
• Scene recognition: Spatial pyramid matching
• Other classifiers
  – Decision trees
  – Hidden Markov models
• Other problems
  – Clustering: agglomerative clustering
  – Dimensionality reduction
Beyond Bags of Features: Spatial Pyramid Matching for Recognizing Natural Scene Categories

CVPR 2006

Svetlana Lazebnik (slazebni@uiuc.edu)
Beckman Institute, University of Illinois at Urbana-Champaign

Cordelia Schmid (cordelia.schmid@inrialpes.fr)
INRIA Rhône-Alpes, France

Jean Ponce (ponce@di.ens.fr)
Ecole Normale Supérieure, France

http://www-cvr.ai.uiuc.edu/ponce_grp
Bags of words
Bag-of-features steps

1. Extract local features
2. Learn “visual vocabulary” using clustering
3. Quantize local features using visual vocabulary
4. Represent images by frequencies of “visual words”
Local feature extraction
Learning the visual vocabulary
Learning the visual vocabulary

Clustering

Slide credit: Josef Sivic
Learning the visual vocabulary

Clustering

Visual vocabulary

Clustering

Slide credit: Josef Sivic
Image categorization with bag of words

Training
1. Extract bag-of-words representation
2. Train classifier on labeled examples using histogram values as features

Testing
1. Extract keypoints/descriptors
2. Quantize into visual words using the clusters computed at training time
3. Compute visual word histogram
4. Compute label using classifier
What about spatial layout?

All of these images have the same color histogram
Spatial pyramid

Compute histogram in each spatial bin

Slide credit: D. Hoiem
Spatial pyramid

[LaZebnik et al. CVPR 2006]
Pyramid matching
Indyk & Thaper (2003), Grauman & Darrell (2005)

Matching using pyramid and histogram intersection for some particular visual word:

Original images

Feature histograms:
Level 3

Level 2

Level 1

Level 0

K( x_i, x_j ) (value of pyramid match kernel): \[ I_3 + \frac{1}{2}(I_2 - I_3) + \frac{1}{4}(I_1 - I_2) + \frac{1}{8}(I_0 - I_1) \]
### Scene category dataset

Fei-Fei & Perona (2005), Oliva & Torralba (2001)

[http://www-cvr.ai.uiuc.edu/ponce_grp/data](http://www-cvr.ai.uiuc.edu/ponce_grp/data)

Multi-class classification results (100 training images per class)

<table>
<thead>
<tr>
<th>Level</th>
<th>Weak features (vocabulary size: 16)</th>
<th>Strong features (vocabulary size: 200)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Single-level</td>
<td>Pyramid</td>
</tr>
<tr>
<td>0 (1 × 1)</td>
<td>45.3 ±0.5</td>
<td></td>
</tr>
<tr>
<td>1 (2 × 2)</td>
<td>53.6 ±0.3</td>
<td>56.2 ±0.6</td>
</tr>
<tr>
<td>2 (4 × 4)</td>
<td>61.7 ±0.6</td>
<td>64.7 ±0.7</td>
</tr>
<tr>
<td>3 (8 × 8)</td>
<td>63.3 ±0.8</td>
<td>66.8 ±0.6</td>
</tr>
</tbody>
</table>

Fei-Fei & Perona: 65.2%

Slide credit: L. Lazebnik
Scene category confusions

Difficult indoor images

kitchen
living room
bedroom
### Caltech101 dataset
Fei-Fei et al. (2004)


---

#### 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>

---

*Slide credit: L. Lazebnik*
Outline

• Support Vector Machines (review + other uses)
• Bias-variance trade-off
• Scene recognition: Spatial pyramid matching

• Other classifiers
  – Decision trees
  – Hidden Markov models

• Other problems
  – Clustering: agglomerative clustering
  – Dimensionality reduction
**Decision tree classifier**

**Example problem:** decide whether to wait for a table at a restaurant, based on the following attributes:

1. **Alternate:** is there an alternative restaurant nearby?
2. **Bar:** is there a comfortable bar area to wait in?
3. **Fri/Sat:** is today Friday or Saturday?
4. **Hungry:** are we hungry?
5. **Patrons:** number of people in the restaurant (None, Some, Full)
6. **Price:** price range ($, $$, $$$)
7. **Raining:** is it raining outside?
8. **Reservation:** have we made a reservation?
9. **Type:** kind of restaurant (French, Italian, Thai, Burger)
10. **WaitEstimate:** estimated waiting time (0-10, 10-30, 30-60, >60)
## Decision tree classifier

<table>
<thead>
<tr>
<th>Example</th>
<th>Attributes</th>
<th>Target</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Alt Bar Fri Hun Pat Price Rain Res Type Est Wait</td>
<td></td>
</tr>
<tr>
<td>$X_1$</td>
<td>T F F T</td>
<td>Some $$$$ F T French 0–10 T</td>
</tr>
<tr>
<td>$X_2$</td>
<td>T F F T</td>
<td>Full $ F F Thai 30–60 F</td>
</tr>
<tr>
<td>$X_3$</td>
<td>F T F F</td>
<td>Some $ F F Burger 0–10 T</td>
</tr>
<tr>
<td>$X_4$</td>
<td>T F T T</td>
<td>Full $ F F Thai 10–30 T</td>
</tr>
<tr>
<td>$X_5$</td>
<td>T F T F</td>
<td>Full $$$$ F T French $&gt;60$ F</td>
</tr>
<tr>
<td>$X_6$</td>
<td>F T F T</td>
<td>Some $$ T T Italian 0–10 T</td>
</tr>
<tr>
<td>$X_7$</td>
<td>F T F F</td>
<td>None $ T F Burger 0–10 F</td>
</tr>
<tr>
<td>$X_8$</td>
<td>F F F T</td>
<td>Some $$ T T Thai 0–10 T</td>
</tr>
<tr>
<td>$X_9$</td>
<td>F T T T</td>
<td>Full $ T F Burger $&gt;60$ F</td>
</tr>
<tr>
<td>$X_{10}$</td>
<td>T T T T</td>
<td>Full $$$$ F T Italian 10–30 F</td>
</tr>
<tr>
<td>$X_{11}$</td>
<td>F F F F</td>
<td>None $ F F Thai 0–10 F</td>
</tr>
<tr>
<td>$X_{12}$</td>
<td>T T T T</td>
<td>Full $ F F Burger 30–60 T</td>
</tr>
</tbody>
</table>
Decision tree classifier

Patrons?

- None
- Some
- Full

Wait Estimate?

> 60

Alternate?

- No
- Yes

Reservation?

- No
- Yes

Fri/Sat?

- No
- Yes

Hungry?

- No
- Yes

Alternate?

- No
- Yes

Bar?

- No
- Yes

Raining?

- No
- Yes

Slide credit: L. Lazebnik
Sequence Labeling Problem

• Unlike most computer vision problems, many NLP problems can viewed as sequence labeling.
• Each token in a sequence is assigned a label.
• Labels of tokens are dependent on the labels of other tokens in the sequence, particularly their neighbors (not i.i.d).

Adapted from Ray Mooney
Markov Model / Markov Chain

- A finite state machine with probabilistic state transitions.
- Makes Markov assumption that next state only depends on the current state and independent of previous history.
Sample Markov Model for POS

Ray Mooney
P(PropNoun Verb Det Noun) = 0.4*0.8*0.25*0.95*0.1 = 0.0076

“Mary ate the cake.”

Adapted from Ray Mooney
Hidden Markov Model

- Probabilistic generative model for sequences.
- Assume an underlying set of hidden (unobserved) states in which the model can be (e.g. parts of speech).
- Assume probabilistic transitions between states over time (e.g. transition from POS to another POS as sequence is generated).
- Assume a probabilistic generation of tokens from states (e.g. words generated for each POS).
Sample HMM for POS

The diagram illustrates a Hidden Markov Model (HMM) for part-of-speech (POS) tagging. The model includes states for different parts of speech, such as determiners (Det), noun phrases (Noun), proper nouns (PropNoun), and verbs (Verb), along with transitions and emission probabilities.

- **Determiners (Det)**: States for 'a', 'the', and 'that' are connected to transitions with probabilities.
- **Noun Phrases (Noun)**: States for 'cat', 'dog', 'car', 'bed', 'pen', and 'apple' are connected with transitions and emission probabilities.
- **Proper Nouns (PropNoun)**: States for 'Tom', 'John', 'Mary', 'Alice', and 'Jerry' are connected with transitions.
- **Verbs (Verb)**: States for 'bit', 'ate', 'saw', 'played', 'hit', and 'gave' are connected with transitions and emission probabilities.

The model transitions and emission probabilities are explicitly shown in the diagram, allowing for the tagging of words as they are processed through the model.
Outline

• Support Vector Machines (review + other uses)
• Bias-variance trade-off
• Scene recognition: Spatial pyramid matching
• Other classifiers
  – Decision trees
  – Hidden Markov models
• Other problems
  – Clustering: agglomerative clustering
  – Dimensionality reduction
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>
Clustering Strategies

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

• Mean-shift clustering
  – Estimate modes

• Graph cuts
  – Split the nodes in a graph based on assigned links with similarity weights

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

1. Say “Every point is its own cluster”
Agglomerative clustering

1. Say “Every point is its own cluster”
2. Find “most similar” pair of clusters
Agglomerative clustering

1. Say “Every point is its own cluster”
2. Find “most similar” pair of clusters
3. Merge it into a parent cluster
Agglomerative clustering

1. Say “Every point is its own cluster”
2. Find “most similar” pair of clusters
3. Merge it into a parent cluster
4. Repeat
Agglomerative clustering

1. Say “Every point is its own cluster”
2. Find “most similar” pair of clusters
3. Merge it into a parent cluster
4. Repeat
Agglomerative clustering

How to define cluster similarity?
- Average distance between points, maximum distance, minimum distance

How many clusters?
- Clustering creates a dendrogram (a tree)
- Threshold based on max number of clusters or based on distance between merges

Adapted from J. Hays
Why do we cluster?

• **Summarizing data**
  – Look at large amounts of data
  – Represent a large continuous vector with the cluster number

• **Counting**
  – Histograms of texture, color, SIFT vectors

• **Segmentation**
  – Separate the image into different regions

• **Prediction**
  – Images in the same cluster may have the same labels

Slide credit: J. Hays, D. Hoiem
Machine Learning Problems

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

- **Unsupervised Learning**
  - Clustering
  - Dimensionality reduction

Slide credit: D. Hoiem
Fig. 8 2D visualization of the SUN Attribute dataset. Each image in the dataset is represented by the projection of its 102-dimensional attribute feature vector onto two dimensions using t-Distributed Stochastic Neighbor Embedding (Van der Maaten and Hinton 2008). There are groups of nearest neighbors, each designated by a color. Interestingly, while the nearest-neighbor scenes in attribute space are semantically very similar, for most of these examples (underwater ocean, abbey, coast, ice skating rink, field_wild, bistro, office) none of the nearest neighbors actually fall in the same SUN database category. The colored border lines delineate the approximate separation of images with and without the attribute associated with the border. Figure best viewed in color (Color figure online).