General recipe

• Fix hypothesis class
  \[ h(x; w, b) = \sigma(w^T \phi(x) + b) \]

• Define loss function
  \[ L(h(x; w, b), y) = -y \log h(x; w, b) + (1 - y) \log(1 - h(x; w, b)) \]

• Minimize average loss on the training set using SGD
  \[ \min_{w, b} \frac{1}{N} \sum_{i=1}^{N} L(h(x_i, w, b), y_i) \]
Risk

• Given:
  • Distribution $\mathcal{D}$ over (x,y) pairs
  • A hypothesis $h \in H$ from hypothesis class H
  • Loss function $L$

• We are interested in Expected Risk:

$$R(h) = \mathbb{E}_{(x,y) \sim \mathcal{D}} L(h(x), y)$$

• Given training set $S$, and a particular hypothesis $h$, Empirical Risk:

$$\hat{R}(S, h) = \frac{1}{|S|} \sum_{(x,y) \in S} L(h(x), y)$$
Generalization

\[ R(h) = \mathbb{E}_{(x,y) \sim \mathcal{D}} L(h(x), y) \]

\[ \hat{R}(S, h) = \frac{1}{|S|} \sum_{(x,y) \in S} L(h(x), y) \]

\[ R(h) = \hat{R}(S, h) + (R(h) - \hat{R}(S, h)) \]

Training error

Generalization error
Controlling generalization error

• How do we know we are overfitting?
  • Use a held-out “validation set”
  • To be an unbiased sample, must be completely unseen
Controlling generalization error

• Variance of empirical risk inversely proportional to size of \( S \)
  • Choose very large \( S \)!

• *Larger* the hypothesis class \( H \), *Higher* the chance of hitting bad hypotheses with low training error and high generalization error
  • Choose small \( H \)!

• For many models, can *bound* generalization error using some property of parameters
  • Regularize during optimization!
  • Eg. L2 regularization
Controlling the size of the hypothesis class

\[ h(x; \mathbf{w}, b) = \sigma(\mathbf{w}^T \phi(x) + b) \]

• How many parameters \((\mathbf{w}, b)\) are there to find?
• Depends on dimensionality of \(\phi\)
• Large dimensionality = large number of parameters
  = more chance of overfitting
• Rule of thumb: size of training set should be at least
  10x number of parameters
• Often training sets are much smaller
Regularization

• Old objective

\[ \min_{w,b} \sum_{i=1}^{N} L(h(x_i; w, b), y_i) \]

• New objective

\[ \min_{w,b} \sum_{i=1}^{N} L(h(x_i; w, b), y_i) + \lambda \| w \|^2 \]

• Why does this help?
Regularization

\[
\min_{w,b} \sum_{i=1}^{N} L(h(x_i; w, b), y_i) + \lambda \|w\|^2
\]

- Ensures classifier does not weigh any one feature too highly
- Makes sure classifier scores \textit{vary slowly} when image changes
  \[
  |w^T \phi(x_1) - w^T \phi(x_2)| \leq \|w\| \|\phi(x_1) - \phi(x_2)\|
  \]
- Prevents “crazy hypotheses” that are unlikely
Generalization error and priors

• Regularization can be thought of as introducing prior knowledge into the model
  • L2-regularization: model output varies slowly as image changes
  • *Biases* the training to consider some hypotheses more than others

• What if bias is wrong?
Bias and variance

• Two things characterize a learning algorithm

• **Variance**
  • How sensitive is the algorithm to the training set?
  • High variance = learnt model varies a lot depending on training set
  • High variance = *overfitting*, i.e., high *generalization error*

• **Bias**
  • How much prior knowledge has been put in?
  • If prior knowledge is wrong, model learnt will not be able to achieve low loss (favors bad hypotheses in general)
  • High bias = *underfitting*, i.e., high *training error*
Bias and variance

Decreasing regularization

Training error
Test error

High bias = underfitting
High variance = overfitting

Decreasing regularization →
Putting it all together

• Want model with least expected risk = expected loss
• But expected risk hard to evaluate
• Empirical Risk Minimization: minimize empirical risk in training set
• Might end up picking special case: overfitting
• Avoid overfitting by:
  • Constructing large training sets
  • Reducing size of model class
  • Regularization
Putting it all together

• Collect training set and validation set
• Pick hypothesis class
• Pick loss function
• Minimize empirical risk (+ regularization)
• Measure performance on held-out validation set
• Profit!
Loss functions and hypothesis classes

<table>
<thead>
<tr>
<th>Loss function</th>
<th>Problem</th>
<th>Range of $h$</th>
<th>$\mathcal{Y}$</th>
<th>Formula</th>
</tr>
</thead>
<tbody>
<tr>
<td>Log loss</td>
<td>Binary Classification</td>
<td>$\mathbb{R}$</td>
<td>${0, 1}$</td>
<td>$\log(1 + e^{-yh(x)})$</td>
</tr>
<tr>
<td>Negative log likelihood</td>
<td>Multiclass classification</td>
<td>$[0, 1]^k$</td>
<td>${1, \ldots, k}$</td>
<td>$- \log hy(x)$</td>
</tr>
<tr>
<td>Hinge loss</td>
<td>Binary Classification</td>
<td>$\mathbb{R}$</td>
<td>${0, 1}$</td>
<td>$\max(0, 1 - yh(x))$</td>
</tr>
<tr>
<td>MSE</td>
<td>Regression</td>
<td>$\mathbb{R}$</td>
<td>$\mathbb{R}$</td>
<td>$(y - h(x))^2$</td>
</tr>
</tbody>
</table>
Back to images

\[ h(x; \mathbf{w}, b) = \sigma(\mathbf{w}^T \phi(x) + b) \]

- What should \( \phi \) be?
- Simplest solution: string 2D image intensity values into vector
Linear classifiers on pixels are bad

- Solution 1: Better feature vectors
- Solution 2: Non-linear classifiers
Better feature vectors

These must have different feature vectors: *discriminability*

These must have similar feature vectors: *invariance*
SIFT

• Match *pattern of edges*
  • Edge orientation – clue to shape

• Be resilient to *small deformations*
  • Deformations might move pixels around, but slightly
  • Deformations might change edge orientations, but slightly
The SIFT descriptor

SIFT – Lowe IJCV 2004
Same but different: HOG

Histogram of oriented gradients
Same as SIFT but without orientation normalization. Why?
Invariance to large deformations
Invariance to large deformations

- Large deformations can cause objects / object parts to move a lot (much more than single grid cell)
- Yet, object parts themselves have precise appearance

- Idea: want to represent the image as a “bag of object parts”
Last night I dreamt I went to Manderley again. It seemed to me I stood by the iron gate leading to the drive, and for a while I could not enter, for the way was barred to me. There was a padlock and a chain upon the gate. I called in my dream to the lodge-keeper, and had no answer, and peering closer through the rusted spokes of the gate I saw that the lodge was uninhabited....
Bags of visual words
What should be visual words?

• A word is a sequence of letters that commonly occurs
  • cthn is not a word, cotton is
• Typically such a sequence of letters means something
• Visual words are image patches that frequently occur
• How do we get these visual words?
What should be visual words?

• “Image patches that occur frequently”
• ..but obviously under small variations of color and deformations
• Each occurrence of image patch is slightly different
What should be visual words?

• Consider representing each image patch with SIFT descriptors
• Consider plotting them out in feature space
What should be visual words?

• Consider plotting SIFT feature vectors and clustering them using k-means
• Each k-means center is a visual word
Identifying the words in an image

• Given a new patch, we can assign it to the closest center
Identifying the words in an image

- Given an image, take every patch and assign it to the closest $k$-means center
  - Each $k$-means center is a “word”
Identifying the words in an image

• Given an image, take every patch and assign it to the closest k-means center
  • Each k-means center is a “word”
Encoding images as bag of words

- Densely extract image patches from image
- Compute SIFT vector for each patch
- Assign each patch to a visual word
- Compute histogram of occurrence
Too much invariance?

- Object parts appear in somewhat fixed relationships
Idea: Spatial pyramids

- Divide the image into four parts
- Compute separate histogram in each part
- Concatenate into a single feature vector
A pipeline for recognition

1. Compute image gradients
2. Compute SIFT descriptors
3. Assign to k-means centers
4. Compute histogram
5. Linear classifier

Horse
Linear classifiers on pixels are bad

- Solution 1: Better feature vectors
- Solution 2: Non-linear classifiers
Non-linear classifiers

• Suppose we have a feature vector for every image
Non-linear classifiers

• Suppose we have a feature vector for every image
  • Linear classifier
Non-linear classifiers

- Suppose we have a feature vector for every image
  - Linear classifier
  - Nearest neighbor: assign each point the label of the nearest neighbor
Non-linear classifiers

• Suppose we have a feature vector for every image
  • Linear classifier
  • Nearest neighbor: assign each point the label of the nearest neighbor
  • Decision tree: series of if-then-else statements on different features
Non-linear classifiers

- Suppose we have a feature vector for every image
  - Linear classifier
  - Nearest neighbor: assign each point the label of the nearest neighbor
  - Decision tree: series of if-then-else statements on different features
  - Neural networks