CS 1674: Intro to Computer Vision
Visual Recognition

Prof. Adriana Kovashka
University of Pittsburgh
March 1, 2022
Plan for this lecture

• What is recognition?
  – a.k.a. classification, categorization

• Support vector machines
  – Separable case / non-separable case
  – Linear / non-linear (kernels)

• Example approach for scene classification

• The importance of generalization
  – The bias-variance trade-off (applies to all classifiers)
Classification

- Given a feature representation for images, how do we learn a model for distinguishing features from different classes?
Classification

- Assign input vector to one of two or more classes
- Input space divided into *decision regions* separated by *decision boundaries*
Examples of image classification

• Two-class (binary): Cat vs Dog
Examples of image classification

- Multi-class (often): Object recognition
Examples of image classification

• Fine-grained recognition

Generalist  Insect catching  Grain eating  Coniferous-seed eating  Nectar feeding
Chiseling  Dip netting  Surface skimming  Scything  Probing
Aerial fishing  Pursuit fishing  Scavenging  Raptorial  Filter feeding

Visipedia Project
Examples of image classification

- Place recognition

Places Database [Zhou et al. NIPS 2014]
Examples of image classification

• Material recognition

[Bell et al. CVPR 2015]
Examples of image classification

- Dating historical photos

1940 1953 1966 1977

[Palermo et al. ECCV 2012]
Examples of image classification

• Image style recognition

Flickr Style: 80K images covering 20 styles.

[Karayev et al. BMVC 2014]
Recognition: A machine learning approach
The machine learning framework

- Apply a prediction function to a feature representation of the image to get the desired output:

  \[
  f(\text{apple}) = \text{“apple”} \\
  f(\text{tomato}) = \text{“tomato”} \\
  f(\text{cow}) = \text{“cow”}
  \]
The machine learning framework

\[ y^* = f(x) \]

- **Training**: given a training set of labeled examples \( \{(x_1, y_1), \ldots, (x_N, y_N)\} \), estimate the prediction function \( f \) by minimizing the prediction error on the training set, e.g. \( |f(x_i) - y_i| \)
  - Evaluate multiple hypotheses \( f_1, f_2, f_H \ldots \) and pick the best one as \( f \)
- **Testing**: apply \( f \) to a never-before-seen test example \( x \) and output the predicted value \( y^* = f(x) \)
The old-school way

Training

Training Images

Training Labels

Image Features

Training

Learned model

Testing

Test Image

Image Features

Learned model

Prediction

Slide credit: D. Hoiem and L. Lazebnik
The simplest classifier

\[ f(x) = \text{label of the training example nearest to } x \]

- All we need is a distance function for our inputs
- No training required!
**K-Nearest Neighbors classification**

- For a new point, find the $k$ closest points from training data
- Labels of the $k$ points “vote” to classify

If query lands here, the 5 NN consist of 3 positives and 2 negatives, so we classify it as positive.
Where was this image taken?

Nearest Neighbors according to BOW-SIFT + color histogram + a few others
The Importance of Data

The graph shows the percentage of geolocations within 200km as a function of database size (in thousands of images, log scale). The green line represents the first nearest neighbor scene match, while the red line represents chance—random scenes. As the database size increases, the percentage of geolocations within 200km also increases significantly for the first nearest neighbor scene match, whereas the chance-scene match remains relatively constant.
• Find a *linear function* to separate the classes

\[ f(x) = \text{sgn}(w_1x_1 + w_2x_2 + \ldots + w_Dx_D) = \text{sgn}(\mathbf{w} \cdot \mathbf{x}) \]
Linear classifier

- Decision = \text{sign}(\mathbf{w}^T \mathbf{x}) = \text{sign}(w_1 x_1 + w_2 x_2)

- What should the weights be?
Let \( w = \begin{bmatrix} a \\ c \end{bmatrix} \quad x = \begin{bmatrix} x \\ y \end{bmatrix} \)

\[ ax + cy + b = 0 \]

Compare to:

\[ \text{slope} \cdot x + y - \text{intercept} = y \]

\[ ax + b = -cy \]

\[ (-a/c) \cdot x + (-b/c) = y \]

Slope: \(-a/c\)

Y-intercept: \(-b/c\)
Let \( w = \begin{bmatrix} a \\ c \end{bmatrix} \quad x = \begin{bmatrix} x \\ y \end{bmatrix} \)

\[
ax + cy + b = 0
\]

\[
w \cdot x + b = 0
\]
Lines in $\mathbb{R}^2$

Let $w = \begin{bmatrix} a \\ c \end{bmatrix}$, $x = \begin{bmatrix} x \\ y \end{bmatrix}$

$$ax + cy + b = 0$$

Slope: $-a/c$

Y-intercept: $-b/c$
Lines in $\mathbb{R}^2$

Let $w = \begin{bmatrix} a \\ c \end{bmatrix}$ and $x = \begin{bmatrix} x \\ y \end{bmatrix}$.

\[ ax + cy + b = 0 \]

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

Distance from point to line:

\[ D = \frac{|ax_0 + cy_0 + b|}{\sqrt{a^2 + c^2}} \]
\( (x_0, y_0) \)

**Lines in \( \mathbb{R}^2 \)**

Let \( w = \begin{bmatrix} a \\ c \end{bmatrix} \) and \( x = \begin{bmatrix} x \\ y \end{bmatrix} \)

\[
ax + cy + b = 0
\]

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

**Distance from point to line**

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

*Adapted from Kristen Grauman*

**Slope:** \(-a/c\)  
**Y-intercept:** \(-b/c\)
Linear classifiers

- Find linear function to separate positive and negative examples

\[ x_i \text{ positive: } x_i \cdot w + b \geq 0 \]
\[ x_i \text{ negative: } x_i \cdot w + b < 0 \]

Which line is best?

Linear classifiers

• Find linear function to separate positive and negative examples

\[ x_i \text{ positive: } x_i \cdot w + b \geq 0 \]
\[ x_i \text{ negative: } x_i \cdot w + b < 0 \]

Which line is best?

Support vector machines

- Discriminative classifier based on optimal separating line (for 2d case)
- Maximize the margin between the positive and negative training examples

Support vector machines

• Want line that maximizes the margin.

\[ \mathbf{x}_i \text{ positive (} y_i = 1) : \quad \mathbf{x}_i \cdot \mathbf{w} + b \geq 1 \]
\[ \mathbf{x}_i \text{ negative (} y_i = -1) : \quad \mathbf{x}_i \cdot \mathbf{w} + b \leq -1 \]

For support vectors, \[ \mathbf{x}_i \cdot \mathbf{w} + b = \pm 1 \]

Support vector machines

- Want line that maximizes the margin.

For support vectors, \( \mathbf{x}_i \cdot \mathbf{w} + b = \pm 1 \)

Distance between point and line:

For support vectors:

\[
\frac{\mathbf{w}^T \mathbf{x} + b}{\| \mathbf{w} \|} = \pm \frac{1}{\| \mathbf{w} \|}
\]

\[
M = \left| \frac{1}{\| \mathbf{w} \|} - \frac{-1}{\| \mathbf{w} \|} \right| = 2
\]
Support vector machines

- Want line that maximizes the margin.

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

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

Therefore, the margin is \( 2 / \|w\| \)

Finding the maximum margin line

1. Maximize margin $2/\|w\|$
2. Correctly classify all training data points:
   \[ x_i \text{ positive } (y_i = 1) : \quad x_i \cdot w + b \geq 1 \]
   \[ x_i \text{ negative } (y_i = -1) : \quad x_i \cdot w + b \leq -1 \]

**Quadratic optimization problem:**

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

Subject to $y_i(w \cdot x_i + b) \geq 1$

One constraint per training point.

Note sign trick:
- $w \cdot x_i + b \geq 1$ (if $y_i = 1$)
- $w \cdot x_i + b \leq -1$ (if $y_i = -1$)
- $(-1) \cdot w \cdot x_i + b \geq 1$

Adapted from C. Burges, *A Tutorial on Support Vector Machines for Pattern Recognition*
Finding the maximum margin line

- Solution: \( \mathbf{w} = \sum_i \alpha_i y_i \mathbf{x}_i \)

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

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

• Notice that it relies on an \textit{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)
Inner product

- The decision boundary for the SVM and its optimization depend on the inner product of two data points (vectors):

\[ f(x) = \text{sign} \left( \mathbf{w} \cdot \mathbf{x} + b \right) = \text{sign} \left( \sum_i \alpha_i y_i \mathbf{x}_i \cdot \mathbf{x} + b \right) \]

- The inner product is equal

\[ (\mathbf{x}_i^T \mathbf{x}) = \| \mathbf{x}_i \|^2 \cos \theta \]

If the angle in between them is 0 then: \[ (\mathbf{x}_i^T \mathbf{x}) = \| \mathbf{x}_i \|^2 \]

If the angle between them is 90 then: \[ (\mathbf{x}_i^T \mathbf{x}) = 0 \]

The inner product measures how similar the two vectors are.

Adapted from Milos Hauskrecht
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:
Nonlinear SVMs

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

\[ \Phi: x \rightarrow \varphi(x) \]
Nonlinear kernel: Example

- Consider the mapping $\varphi(x) = (x, x^2)$

$$\varphi(x) \cdot \varphi(y) = (x, x^2) \cdot (y, y^2) = xy + x^2 y^2$$

$$K(x, y) = xy + x^2 y^2$$
The “Kernel Trick”

- The linear classifier relies on dot product between vectors $K(x_i, x_j) = x_i \cdot x_j$
- If every data point is mapped into high-dimensional space via some transformation $\Phi: x_i \rightarrow \phi(x_i)$, the dot product becomes: $K(x_i, x_j) = \phi(x_i) \cdot \phi(x_j)$
- A kernel function is similarity function that corresponds to an inner product in some expanded feature space
- *The kernel trick*: instead of explicitly computing the lifting transformation $\phi(x)$, define a kernel function $K$ such that: $K(x_i, x_j) = \phi(x_i) \cdot \phi(x_j)$
Examples of kernel functions

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

- **Polynomials of degree up to \( d \):**
  \[ K(x_i, x_j) = (x_i^T x_j + 1)^d \]

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

- **Histogram intersection:**
  \[ K(x_i, x_j) = \sum_k \min(x_i(k), x_j(k)) \]
The benefit of the “kernel trick”

- Example: Polynomial kernel for 2-dim features

\[
k(x, z) = (1 + x^T z)^2 = (1 + x_1 z_1 + x_2 z_2)^2 \\
= 1 + 2x_1 z_1 + 2x_2 z_2 + x_1^2 z_1^2 + 2x_1 x_2 z_2 + x_2^2 z_2^2 \\
= (1, \sqrt{2}x_1, \sqrt{2}x_2, x_1^2, \sqrt{2}x_1 x_2, x_2^2)(1, \sqrt{2}z_1, \sqrt{2}z_2, z_1^2, \sqrt{2}z_1 z_2, z_2^2)^T \\
= \phi(x)^T \phi(z).  \\
\]  

(7.42)

- ... lives in 6 dimensions
- With the kernel trick, we directly compute an inner product in 2-dim space, obtaining a scalar that we add 1 to and exponentiate
Hard-margin SVMs

\[
\min_w \quad \frac{1}{2} \|w\|^2
\]

The \(w\) that minimizes...

Maximize margin

subject to \(y_i w^T x_i \geq 1\),

\forall i = 1, \ldots, N
Soft-margin SVMs

The $w$ that minimizes...

Maximize margin

Minimize misclassification

subject to

$y_i w^T x_i \geq 1 - \xi_i, \quad \forall i = 1, \ldots, N$

The $w$ that minimizes...

$$\min_w \frac{1}{2} \|w\|^2 + C \sum_{i=1}^{N} \xi_i$$

Misclassification cost

# data samples

Slack variable

Figure from Chris Bishop
Example of choosing optimal $w$

- Suppose $w$ is just a scalar, $w$, that can only take values 1, 2, 3
- Let $L(w)$ denote the sum of $\xi_i$ values incurred when using a particular $w$
- Suppose $L(w=1) = 4$, $L(w=2) = 5$, $L(w=3) = 0.5$
- Find the optimal $w$ as $\text{argmin}_w ||w||$
  - What is the optimal $w$?
- Now find the optimal $w$ for $L^\wtilde(w) = ||w|| + L(w)$
  - $L^\wtilde(1) = 1 + 4 = 5$
  - $L^\wtilde(2) = ? L^\wtilde(3) = ?$
  - Now what is the optimal $w$?
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/all**
- 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.
Multi-class problems

One-vs-all (a.k.a. one-vs-others)

- Train K classifiers
- In each, pos = data from class $i$, neg = data from classes other than $i$
- The class with the most confident prediction wins
- Example:
  - You have 4 classes, train 4 classifiers
  - 1 vs others: score 3.5
  - 2 vs others: score 6.2
  - 3 vs others: score 1.4
  - 4 vs other: score 5.5
  - Final prediction: class 2
Multi-class problems

One-vs-one (a.k.a. all-vs-all)

- Train $K(K-1)/2$ binary classifiers (all pairs of classes)
- They all vote for the label
- Example:
  - You have 4 classes, then train 6 classifiers
  - 1 vs 2, 1 vs 3, 1 vs 4, 2 vs 3, 2 vs 4, 3 vs 4
  - Votes: 1, 1, 4, 2, 4, 4
  - Final prediction is class 4
Using SVMs

1. Select a kernel function.

2. Compute pairwise kernel values between labeled examples.

3. Use this “kernel matrix” to solve for SVM support vectors & alpha weights.

4. To classify a new example: compute kernel values between new input and support vectors, apply alpha weights, check sign of output.

Adapted from Kristen Grauman
Some SVM packages

- LIBLINEAR [https://www.csie.ntu.edu.tw/~cjlin/liblinear/](https://www.csie.ntu.edu.tw/~cjlin/liblinear/)
Linear classifiers vs nearest neighbors

- **Linear pros:**
  - Low-dimensional *parametric* representation
  - Very fast at test time

- **Linear cons:**
  - Can be tricky to select best kernel function for a problem
  - Learning can take a very long time for large-scale problem

- **NN pros:**
  - Works for any number of classes
  - Decision boundaries not necessarily linear
  - *Nonparametric* method
  - Simple to implement

- **NN cons:**
  - Slow at test time (large search problem to find neighbors)
  - Storage of data
  - Especially need good distance function (but true for all classifiers)

Adapted from L. Lazebnik
Beyond Bags of Features: Spatial Pyramid Matching for Recognizing Natural Scene Categories

CVPR 2006
Winner of 2016 Longuet-Higgins Prize

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
Scene category dataset
Fei-Fei & Perona (2005), Oliva & Torralba (2001)
http://www-cvr.ai.uiuc.edu/ponce_grp/data
Bag-of-words representation

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”
Image categorization with bag of words

Training
1. Compute bag-of-words representation for training images
2. Train classifier on labeled examples using histogram values as features
3. Labels are the scene types (e.g. mountain vs field)

Testing
1. Extract keypoints/descriptors for test images
2. Quantize into visual words using the clusters computed at training time
3. Compute visual word histogram for test images
4. Compute labels on test images using classifier obtained at training time
5. **Evaluation only, do only once:** Measure accuracy of test predictions by comparing them to ground-truth test labels (obtained from humans)

Adapted from D. Hoiem
Feature extraction (on which BOW is based)

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)

Slide credit: L. Lazebnik
What about spatial layout?

All of these images have the same color histogram
Spatial pyramid

Compute histogram in each spatial bin
Spatial pyramid

level 0

level 1

level 2

[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): \( \mathcal{I}_3 + \frac{1}{2}(\mathcal{I}_2 - \mathcal{I}_3) + \frac{1}{4}(\mathcal{I}_1 - \mathcal{I}_2) + \frac{1}{8}(\mathcal{I}_0 - \mathcal{I}_1) \)
Scene category dataset
Fei-Fei & Perona (2005), Oliva & Torralba (2001)

[Link to dataset](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><strong>66.8 ±0.6</strong></td>
</tr>
</tbody>
</table>

Fei-Fei & Perona: 65.2%
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
Training vs Testing

• What do we want?
  – High accuracy on training data?
  – No, high accuracy on unseen/new/test data!
  – Why is this tricky?

• Training data
  – Features (x) and labels (y) used to learn mapping f

• Test data
  – Features (x) used to make a prediction
  – Labels (y) only used to see how well we’ve learned f!!!

• Validation data
  – Held-out set of the training data
  – Can use both features (x) and labels (y) to tune parameters of the model we’re learning
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**
  - **Noise** in our observations: unavoidable
  - **Bias**: due to inaccurate assumptions/simplifications by model
  - **Variance**: models estimated from different training sets differ greatly 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
Generalization

Dataset subset 1
Dataset subset 2
Dataset subset 3
Dataset subset 4
Dataset subset 5

Model 1
Model 2
Model 3
Model 4
Model 5

Average model

Variance

Bias

True model?
Generalization

- 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).

Red dots = training data (all that we see before we ship off our model!)
Green curve = true underlying model
Blue curve = our predicted model/fit
Purple dots = possible test points

Adapted from D. Hoiem
Polynomial Curve Fitting

\[ y(x, \mathbf{w}) = w_0 + w_1 x + w_2 x^2 + \ldots + w_M x^M = \sum_{j=0}^{M} w_j x^j \]
Sum-of-Squares Error Function

\[ E(w) = \frac{1}{2} \sum_{n=1}^{N} \left( y(x_n, w) - t_n \right)^2 \]
$0^{th}$ Order Polynomial

![Graph of a 0th order polynomial with points and a horizontal line at M = 0.](image-url)

Slide credit: Chris Bishop
1st Order Polynomial
3rd Order Polynomial

Slide credit: Chris Bishop
9th Order Polynomial

Slide credit: Chris Bishop
Over-fitting

Root-Mean-Square (RMS) Error: 

\[ E_{RMS} = \sqrt{2E(w^*)/N} \]
Data Set Size: $N = 15$

9th Order Polynomial

\[ \text{Slide credit: Chris Bishop} \]
Data Set Size: $N = 100$

9th Order Polynomial

Slide credit: Chris Bishop
Regularization

Penalize large coefficient values

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

(Remember: We want to minimize this expression.)
Regularization: $\ln \lambda = -18$

Slide credit: Chris Bishop
Regularization: $\ln \lambda = 0$
## Polynomial Coefficients

<table>
<thead>
<tr>
<th></th>
<th>$M = 0$</th>
<th>$M = 1$</th>
<th>$M = 3$</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>
<td>0.35</td>
</tr>
<tr>
<td>$w_1^*$</td>
<td></td>
<td>-1.27</td>
<td>7.99</td>
<td>232.37</td>
</tr>
<tr>
<td>$w_2^*$</td>
<td></td>
<td></td>
<td>-25.43</td>
<td>-5321.83</td>
</tr>
<tr>
<td>$w_3^*$</td>
<td></td>
<td></td>
<td>17.37</td>
<td>48568.31</td>
</tr>
<tr>
<td>$w_4^*$</td>
<td></td>
<td></td>
<td></td>
<td>-231639.30</td>
</tr>
<tr>
<td>$w_5^*$</td>
<td></td>
<td></td>
<td></td>
<td>640042.26</td>
</tr>
<tr>
<td>$w_6^*$</td>
<td></td>
<td></td>
<td></td>
<td>-1061800.52</td>
</tr>
<tr>
<td>$w_7^*$</td>
<td></td>
<td></td>
<td></td>
<td>1042400.18</td>
</tr>
<tr>
<td>$w_8^*$</td>
<td></td>
<td></td>
<td></td>
<td>-557682.99</td>
</tr>
<tr>
<td>$w_9^*$</td>
<td></td>
<td></td>
<td></td>
<td>125201.43</td>
</tr>
</tbody>
</table>
## Polynomial Coefficients

<table>
<thead>
<tr>
<th></th>
<th>No regularization</th>
<th>Huge regularization</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\ln \lambda = -\infty$</td>
<td>$w_0^*$ = 0.35</td>
<td>$\ln \lambda = -18$</td>
</tr>
<tr>
<td></td>
<td>$w_1^*$ = 232.37</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$w_2^*$ = -5321.83</td>
<td>$w_2^*$ = -0.77</td>
</tr>
<tr>
<td></td>
<td>$w_3^*$ = 48568.31</td>
<td>$w_3^*$ = -31.97</td>
</tr>
<tr>
<td></td>
<td>$w_4^*$ = -231639.30</td>
<td>$w_4^*$ = -3.89</td>
</tr>
<tr>
<td></td>
<td>$w_5^*$ = 640042.26</td>
<td>$w_5^*$ = 55.28</td>
</tr>
<tr>
<td></td>
<td>$w_6^*$ = -1061800.52</td>
<td>$w_6^*$ = 41.32</td>
</tr>
<tr>
<td></td>
<td>$w_7^*$ = 1042400.18</td>
<td>$w_7^*$ = -45.95</td>
</tr>
<tr>
<td></td>
<td>$w_8^*$ = -557682.99</td>
<td>$w_8^*$ = -91.53</td>
</tr>
<tr>
<td></td>
<td>$w_9^*$ = 125201.43</td>
<td>$w_9^*$ = 72.68</td>
</tr>
</tbody>
</table>

Adapted from Chris Bishop
Regularization: $E_{\text{RMS}}$ vs. $\ln \lambda$
Training vs test error

- Underfitting
- Overfitting

Error vs Complexity

- High Bias
- Low Variance
- Low Bias
- High Variance
The effect of training set size
Choosing the trade-off between bias and variance

- Need validation set (separate from the test set)

![Graph showing the trade-off between bias and variance](image)

Apply this model to test set
Generalization tips

• 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
• As an additional technique for reducing variance, try regularizing the parameters (penalize high magnitude weights)