Robot Learning
General Pipeline

1. Data acquisition (e.g., from 3D sensors)
2. Feature extraction and representation construction
3. **Robot learning**: e.g., classification (recognition) or clustering (knowledge discovery)
4. Decision making or planning
5. Action execution
Robot Learning

• There are generally three types of robot learning:
  
  ➢ Learning from data
  
  ➢ Learning by demonstration
  
  ➢ Reinforcement learning
Robot Learning

- There are generally three types of robot learning:
  - Learning from data
  - Learning by demonstration
  - Reinforcement learning
Learning from data

**CLUSTERING**
- Data is not labeled
- Group points that are “close” to each other
- Identify structure or patterns in data
- Unsupervised learning

**CLASSIFICATION**
- Labeled data points
- Want a “rule” that assigns labels to new points
- Supervised learning
Learning from data

• **Supervised learning** (Classification)
  - Support Vector Machines (SVMs)
  - Decision Trees
  - Ensemble methods

• **Unsupervised learning** (Clustering)
  - K-means
  - Gaussian Mixture Models
Support Vector Machines

• A Support Vector Machine (SVM) is a supervised learning (i.e., classification) model that recognizes patterns using a separating hyperplane.

• Given a set of training data (with semantic labels), an SVM training algorithm builds a model that outputs an optimal hyperplane which assigns new examples into one category or the other.

• SVM is a non-probabilistic binary linear classifier (as our first SVM model for discussion)
Support Vector Machines

• The original SVM algorithm was invented by Vladimir N. Vapnik and Alexey Ya. Chervonenkis in 1963, based on Statistical Learning Theory.

• In 1992, Bernhard E. Boser, Isabelle M. Guyon and Vladimir N. Vapnik suggested a way to create nonlinear classifiers by applying the kernel trick to maximum-margin hyperplanes.

• Empirically good performance: successful applications in many fields (e.g., text processing, image processing, bioinformatics, robotics, etc.)
Support Vector Machines

• For example: We want to recognize human behaviors
  • We have a training dataset of skeletal data for the human activity “jump forward”
  • Now given a new (previously unseen) skeleton sequence, we want to answer the question: is the human performing “jump forward”, or not?
Support Vector Machines

As an illustration, here we deal with lines and points in the Cartesian plane instead of hyperplanes and vectors in a high dimensional space.

How can we choose a line to separate the data?
Support Vector Machines

Is this a good line?
Support Vector Machines

Is this a good line?
Support Vector Machines

Is this a good line?
Support Vector Machines

Which line is the best? …and Why?
Support Vector Machines

H3 can best separate two categories of data, as it maximizes the margin
Support Vector Machines

Nice properties: convex, theoretically motivated, nonlinear with kernels.
Support Vector Machines

• We are given a training dataset of \( n \) instances of the form

\[
(x_1, y_1), \ldots, (x_n, y_n)
\]

• The \( y_i \) are either 1 or \(-1\), each indicating the class to which the point \( x_i \) belongs

• Each \( x_i \) is a \( p \)-dimensional real vector
Support Vector Machines

• We are given a training dataset of $n$ instances of the form

\[(x_1, y_1), \ldots, (x_n, y_n)\]

• We want to find the "maximum-margin hyperplane" that divides the group of instances $x_i$, where $y_i = 1$ if $x_i$ belong to the category and $y_i = -1$ if $x_i$ does not belong to the category

• The hyperplane is defined so that the distance between the hyperplane and the nearest data point from either group is maximized
Support Vector Machines

• Any hyperplane can be written as the set of points \( x \) satisfying

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

where \( w \) is the parameter (normal vector) of the hyperplane.
Support Vector Machines

• The parameter $\frac{b}{\|w\|}$ determines the offset of the hyperplane from the origin along the normal vector $w$. 

\[ \frac{b}{\|w\|} \]
Support Vector Machines

- If the data instances are linearly separable, we can select two parallel hyperplanes (i.e., as decision boundaries) that separate the two categories of data.
- The distance (i.e., margin) between these two hyperplanes is as large as possible.
- The maximum-margin hyperplane is the one that lies halfway between them.
Support Vector Machines

• These hyperplanes can be described by the equations
  \[ w \cdot x - b = 1 \]
  and
  \[ w \cdot x - b = -1 \]
Support Vector Machines

• Geometrically, the distance between these two planes is
  \[ \frac{2}{\|w\|} \]

• Thus, maximizing the distance between the planes is equivalent to minimize \( \|w\| \)
Support Vector Machines

• We also have to prevent data points from falling into the margin, we add the following constraint: for each $i$ either

$$w \cdot x_i - b \geq 1, \text{ if } y_1 = 1$$

or

$$w \cdot x_i - b \leq -1, \text{ if } y_1 = -1$$

• These constraints state that each data point must lie on the correct side of the margin
Support Vector Machines

• We also have to prevent data points from falling into the margin, we add the following constraint: for each $i$ either

\[ \mathbf{w} \cdot \mathbf{x}_i - b \geq 1, \text{ if } y_i = 1 \]

or

\[ \mathbf{w} \cdot \mathbf{x}_i - b \leq -1, \text{ if } y_i = -1 \]

• This can be rewritten as:

\[ y_i (\mathbf{w} \cdot \mathbf{x}_i - b) \geq 1 \]
Support Vector Machines

- The final optimization problem becomes:

  \[ \text{minimize } \|w\| \]

  \[ \text{subject to: } \]

  \[ y_i (w \cdot x_i - b) \geq 1 \]

  \[ \text{for all } 1 \leq i \leq n \]

- After solving this optimization problem, given a new measurement \( x \), we classify it as:

  \[ x \rightarrow \text{sgn}(w \cdot x - b) \]
Support Vector Machines

Lagrange multiplier

From Wikipedia, the free encyclopedia

In mathematical optimization, the method of Lagrange multipliers (named after Joseph Louis Lagrange\(^1\)) is a strategy for finding the local maxima and minima of a function subject to equality constraints.

For instance (see Figure 1), consider the optimization problem

maximize \( f(x, y) \)

subject to \( g(x, y) = 0 \).

We need both \( f \) and \( g \) to have continuous first partial derivatives. We introduce a new variable (\( \lambda \)) called a Lagrange multiplier and study the Lagrange function (or Lagrangian) defined by

\[
\mathcal{L}(x, y, \lambda) = f(x, y) + \lambda \cdot g(x, y),
\]

where the \( \lambda \) term may be either added or subtracted. If \( f(x_0, y_0) \) is a maximum of \( f(x, y) \) for the original constrained problem, then there exists \( \lambda_0 \) such that \( (x_0, y_0, \lambda_0) \) is a stationary point for the Lagrange function (stationary points are those points where the partial derivatives of \( \mathcal{L} \) are zero). However, not all stationary points yield a solution of the original problem. Thus, the method of Lagrange multipliers yields a necessary condition for optimality in constrained problems.\(^2\)\(^3\)\(^4\)\(^5\)\(^6\) Sufficient conditions for a minimum or maximum also exist.

Relevant courses: Machine Learning and Convex Optimization
Support Vector Machines

• **Support vectors** are the data points that lie closest to the detection surface
• They are the most difficult to classify
• They have direct bearing on the optimum location of the decision boundaries
Support Vector Machines

How to deal with non-separable data?
(i.e., when some points falling on the wrong side of the decision boundary)
Support Vector Machines

• We can use the “soft” margin (based on the slack variables), and the problem can be rewritten as

Minimize:

$$||w||^2 + C \sum_{i=1}^{m} \xi_i$$

subject to:

$$y_i (w \cdot x_i + b) \geq 1 - \xi_i, \quad \xi_i \geq 0$$
Support Vector Machines

How about when the data are not linear?

Linear classifiers aren’t complex enough sometimes. SVM solution:

Map data into a richer feature space including nonlinear features, then construct a hyperplane in that space so all other equations are the same!

Formally, preprocess the data with:

\[ x \mapsto \Phi(x) \]

and then learn the map from \( \Phi(x) \) to \( y \):

\[ f(x) = w \cdot \Phi(x) + b. \]
Support Vector Machines

Transformation to separate
Support Vector Machines

\[ \Phi : \mathbb{R}^2 \rightarrow \mathbb{R}^3 \]

\[ (x_1, x_2) \mapsto (z_1, z_2, z_3) := (x_1^2, \sqrt{2}x_1x_2, x_2^2) \]
Support Vector Machines

How about when the data are not linear?

Linear classifiers aren’t complex enough sometimes. SVM solution: *Map data into a richer feature space including nonlinear features, then construct a hyperplane in that space so all other equations are the same!* Formally, preprocess the data with:

\[ x \mapsto \Phi(x) \]

and then learn the map from \( \Phi(x) \) to \( y \):

\[ f(x) = w \cdot \Phi(x) + b. \]
Support Vector Machines

Problem: the dimensionality of $\Phi(x)$ can be very large, making $w$ hard to represent explicitly in memory, and hard to solve.

The Representer theorem (Kimeldorf & Wahba, 1971) shows that (for SVMs as a special case):

$$w = \sum_{i=1}^{m} \alpha_i \Phi(x_i)$$

for some variables $\alpha$. Instead of optimizing $w$ directly we can thus optimize $\alpha$.

The decision rule is now:

$$f(x) = \sum_{i=1}^{m} \alpha_i \Phi(x_i) \cdot \Phi(x) + b$$

We call $K(x_i, x) = \Phi(x_i) \cdot \Phi(x)$ the kernel function.

SVMs: the kernel trick
Polynomial-SVMs

The kernel \( K(x, x') = (x \cdot x')^d \) gives the same result as the explicit mapping + dot product that we described before:

\[
\Phi : \mathbb{R}^2 \to \mathbb{R}^3 \quad (x_1, x_2) \mapsto (z_1, z_2, z_3) := (x_1^2, \sqrt{2}x_1x_2, x_2^2)
\]

\[
\Phi((x_1, x_2) \cdot \Phi((x_1', x_2')) = (x_1^2, \sqrt{2}x_1x_2, x_2^2) \cdot (x'_1^2, \sqrt{2}x'_1x'_2, x'_2^2)
\]

\[
= x_1^2x'_1^2 + 2x_1x'_1x_2x'_2 + x_2^2x'_2^2
\]
is the same as:

\[
K(x, x') = (x \cdot x')^2 = ((x_1, x_2) \cdot (x'_1, x'_2))^2
\]

\[
= (x_1x'_1 + x_2x'_2)^2 = x_1^2x'_1^2 + x_2^2x'_2^2 + 2x_1x'_1x_2x'_2
\]

Interestingly, if \( d \) is large the kernel is still only requires \( n \) multiplications to compute, whereas the explicit representation may not fit in memory!
RBF-SVMs (radial basis function)

The RBF kernel \( K(x, x') = \exp(-\gamma \|x - x'\|^2) \) is one of the most popular kernel functions. It adds a "bump" around each data point:

\[
f(x) = \sum_{i=1}^{m} \alpha_i \exp(-\gamma \|x_i - x\|^2) + b
\]

Using this one can get state-of-the-art results.
Support Vector Machines

Lots of SVM software:

- LibSVM (C++)
- SVMLight (C)

As well as complete machine learning toolboxes that include SVMs:

- Torch (C++)
- Spider (Matlab)
- Weka (Java)
Support Vector Machines

Have we solved all problems?
Support Vector Machines

• Multi-class classification: one-versus-all or one-versus-one
Support Vector Machines

• Online, onboard applications: Sliding window techniques
Support Vector Machines

Have we solved all problems?
Support Vector Machines

• Evaluation of a learning algorithm: Training and Testing
Support Vector Machines

- Evaluation of a learning algorithm: Cross-validation during training
Support Vector Machines

• Evaluation of a learning algorithm: Confusion Matrix

<table>
<thead>
<tr>
<th></th>
<th>$p'$ (Predicted)</th>
<th>$n'$ (Predicted)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$p$ (Actual)</td>
<td>True Positive</td>
<td>False Negative</td>
</tr>
<tr>
<td>$n$ (Actual)</td>
<td>False Positive</td>
<td>True Negative</td>
</tr>
</tbody>
</table>
Support Vector Machines

• Evaluation of a learning algorithm: Confusion Matrix

\[
\begin{array}{cccccccccccccc}
 & ae & ah & aw & eh & er & ey & ih & iy & oa & oo & uh & uw \\
ae & 128 & 0 & 0 & 6 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
ah & 0 & 117 & 14 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 4 \\
aw & 0 & 6 & 126 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 \\
eh & 14 & 0 & 0 & 125 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
er & 0 & 0 & 0 & 0 & 119 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
ey & 0 & 0 & 0 & 1 & 0 & 116 & 1 & 6 & 0 & 0 & 0 & 0 \\
ih & 3 & 0 & 0 & 1 & 0 & 3 & 132 & 0 & 0 & 0 & 0 & 0 \\
iy & 0 & 0 & 0 & 0 & 4 & 1 & 121 & 0 & 0 & 0 & 0 & 0 \\
oa & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 133 & 0 & 0 & 3 & 0 \\
oo & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 132 & 2 & 3 & 0 \\
uh & 0 & 5 & 5 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 126 & 0 \\
uw & 0 & 0 & 0 & 0 & 2 & 0 & 5 & 1 & 0 & 128 & 0 & 0 \\
\end{array}
\]
<table>
<thead>
<tr>
<th>Actual ↓</th>
<th>State1 (Predicted)</th>
<th>State2 (Predicted)</th>
<th>State3 (Predicted)</th>
<th>State4 (Predicted)</th>
<th>State5 (Predicted)</th>
<th>State6 (Predicted)</th>
<th>State7 (Predicted)</th>
<th>State8 (Predicted)</th>
</tr>
</thead>
<tbody>
<tr>
<td>State1 (Actual)</td>
<td>90.12 %</td>
<td>0.00 %</td>
<td>9.88 %</td>
<td>0.00 %</td>
<td>0.00 %</td>
<td>0.00 %</td>
<td>0.00 %</td>
<td>0.00 %</td>
</tr>
<tr>
<td>State2 (Actual)</td>
<td>0.00 %</td>
<td>100.00 %</td>
<td>0.00 %</td>
<td>0.00 %</td>
<td>0.00 %</td>
<td>0.00 %</td>
<td>0.00 %</td>
<td>0.00 %</td>
</tr>
<tr>
<td>State3 (Actual)</td>
<td>0.00 %</td>
<td>0.00 %</td>
<td>92.66 %</td>
<td>0.00 %</td>
<td>0.00 %</td>
<td>7.34 %</td>
<td>0.00 %</td>
<td>0.00 %</td>
</tr>
<tr>
<td>State4 (Actual)</td>
<td>0.00 %</td>
<td>0.00 %</td>
<td>0.00 %</td>
<td>100.00 %</td>
<td>0.00 %</td>
<td>0.00 %</td>
<td>0.00 %</td>
<td>0.00 %</td>
</tr>
<tr>
<td>State5 (Actual)</td>
<td>0.00 %</td>
<td>0.00 %</td>
<td>12.69 %</td>
<td>0.00 %</td>
<td>87.31 %</td>
<td>0.00 %</td>
<td>0.00 %</td>
<td>0.00 %</td>
</tr>
</tbody>
</table>