Outline

- Linear Support Vector Machines
- Nonlinear SVM, Kernel methods
- Multiclass classification
Support Vector Machines

- Given training examples $(x_1, y_1), \cdots, (x_n, y_n)$
  - Consider binary classification: $y_i \in \{+1, -1\}$
- Linear Support Vector Machine (SVM):
  \[
  \arg \min_w C \sum_{i=1}^{n} \max(1 - y_i w^T x_i, 0) + \frac{1}{2} w^T w
  \]
  (hinge loss with L2 regularization)
Support Vector Machines

- Goal: Find a hyperplane to separate these two classes of data:
  \[ y_i = 1, \quad w^T x_i \geq 1; \quad \text{if } y_i = -1, \quad w^T x_i \leq -1. \]

- Prefer a hyperplane with maximum margin.
Support Vector Machines

- Goal: Find a hyperplane to separate these two classes of data:
  \[ \text{if } y_i = 1, \ w^T x_i \geq 1; \quad \text{if } y_i = -1, \ w^T x_i \leq -1. \]

Prefer a hyperplane with \text{maximum margin}
Size of margin

- minimum of $||x||$ such that $w^T x = 1$
Size of margin

- minimum of $\|x\|$ such that $w^T x = 1$
- clearly, $x = \alpha \frac{w}{\|w\|}$ for some $\alpha$ (half margin)
Size of margin

- minimum of $\|x\|$ such that $w^T x = 1$
- clearly, $x = \alpha \frac{w}{\|w\|}$ for some $\alpha$ (half margin)
- $\alpha = \frac{1}{\|w\|}$
Size of margin

- minimum of $||x||$ such that $w^T x = 1$
- clearly, $x = \alpha \frac{w}{||w||}$ for some $\alpha$ (half margin)
- $\alpha = \frac{1}{||w||}$
- Maximize margin $\Rightarrow$ minimize $||w||$
Support Vector Machines (hard constraints)

- SVM primal problem (with hard constraints):

\[
\min_w \frac{1}{2} w^T w \\
\text{s.t. } y_i(w^T x_i) \geq 1, \ i = 1, \ldots, n,
\]
Support Vector Machines (hard constraints)

- SVM primal problem (with hard constraints):

\[
\min_w \frac{1}{2} w^T w \\
\text{s.t. } y_i(w^T x_i) \geq 1, \ i = 1, \ldots, n,
\]

- What if there are outliers?
Support Vector Machines

- Given training data $x_1, \cdots, x_n \in \mathbb{R}^d$ with labels $y_i \in \{+1, -1\}$.
- SVM primal problem:

$$\min_{w, \xi} \frac{1}{2} w^T w + C \sum_{i=1}^{n} \xi_i$$

s.t. $y_i(w^T x_i) \geq 1 - \xi_i, i = 1, \ldots, n,$

$\xi_i \geq 0$
Support Vector Machines

- SVM primal problem:

\[
\min_{\mathbf{w}, \xi} \frac{1}{2} \mathbf{w}^T \mathbf{w} + C \sum_{i=1}^{n} \xi_i \\
\text{s.t. } y_i(\mathbf{w}^T \mathbf{x}_i) \geq 1 - \xi_i, i = 1, \ldots, n, \\
\xi_i \geq 0
\]

- Equivalent to

\[
\min_{\mathbf{w}} \underbrace{\frac{1}{2} \mathbf{w}^T \mathbf{w}}_{\text{L2 regularization}} + C \sum_{i=1}^{n} \max(0, 1 - y_i \mathbf{w}^T \mathbf{x}_i) \\
\text{hinge loss}
\]

- Non-differentiable when \( y_i \mathbf{w}^T \mathbf{x}_i = 1 \) for some \( i \)
Stochastic Subgradient Method for SVM

- A subgradient of $\ell_i(w) = \max(0, 1 - y_i w^T x_i)$:

$$
\begin{cases}
-y_i x_i & \text{if } 1 - y_i w^T x_i > 0 \\
0 & \text{if } 1 - y_i w^T x_i < 0 \\
0 & \text{if } 1 - y_i w^T x_i = 0
\end{cases}
$$

- Stochastic Subgradient descent for SVM:

For $t = 1, 2, \ldots$

Randomly pick an index $i$

If $y_i w^T x_i < 1$, then

$$w \leftarrow (1 - \eta_t) w + \eta_t nC y_i x_i$$

Else (if $y_i w^T x_i \geq 1$):

$$w \leftarrow (1 - \eta_t) w$$
Kernel SVM
Non-linearly separable problems

What if the data is not linearly separable?

Solution: map data $x_i$ to higher dimensional (maybe infinite) feature space $\varphi(x_i)$, where they are linearly separable.
SVM with nonlinear mapping

- SVM with nonlinear mapping \( \varphi(\cdot) \):

\[
\min_{w,\xi} \frac{1}{2} w^T w + C \sum_{i=1}^{n} \xi_i \\
\text{s.t. } y_i(w^T \varphi(x_i)) \geq 1 - \xi_i, \quad \xi_i \geq 0, \quad i = 1, \ldots, n,
\]

Hard to solve if \( \varphi(\cdot) \) maps to very high or infinite dimensional space.
SVM with nonlinear mapping

- SVM with nonlinear mapping $\varphi(\cdot)$:

$$\min_{\mathbf{w}, \xi} \frac{1}{2} \mathbf{w}^T \mathbf{w} + C \sum_{i=1}^{n} \xi_i$$

s.t. $y_i(\mathbf{w}^T \varphi(\mathbf{x}_i)) \geq 1 - \xi_i, \quad \xi_i \geq 0, \quad i = 1, \ldots, n,$

- Hard to solve if $\varphi(\cdot)$ maps to very high or infinite dimensional space
Support Vector Machines (dual)

- Primal problem:
  \[
  \min_{w, \xi} \frac{1}{2} \| w \|^2 + C \sum_i \xi_i \\
  \text{s.t. } y_i w^T \varphi(x_i) - 1 + \xi_i \geq 0, \text{ and } \xi_i \geq 0 \quad \forall i = 1, \ldots, n
  \]

- Equivalent to:
  \[
  \min_{w, \xi} \max_{\alpha \geq 0, \beta \geq 0} \frac{1}{2} \| w \|^2 + C \sum_i \xi_i - \sum_i \alpha_i (y_i w^T \varphi(x_i) - 1 + \xi_i) - \sum_i \beta_i \xi_i
  \]

- Under certain condition (e.g., slater’s condition), exchanging min, max will not change the optimal solution:
  \[
  \max_{\alpha \geq 0, \beta \geq 0} \min_{w, \xi} \frac{1}{2} \| w \|^2 + C \sum_i \xi_i - \sum_i \alpha_i (y_i w^T \varphi(x_i) - 1 + \xi_i) - \sum_i \beta_i \xi_i
  \]
Support Vector Machines (dual)

- Reorganize the equation:

\[
\max_{\alpha \geq 0, \beta \geq 0} \min_{w, \xi} \frac{1}{2} \|w\|^2 - \sum_i \alpha_i y_i w^T \varphi(x_i) + \sum_i \xi_i (C - \alpha_i - \beta_i) + \sum_i \alpha_i
\]

- Now, for any given \( \alpha, \beta \), the minimizer of \( w \) will satisfy

\[
\frac{\partial L}{\partial w} = w - \sum_i \alpha_i y_i \varphi(x_i) = 0 \Rightarrow w^* = \sum_i y_i \alpha_i \varphi(x_i)
\]

- Also, we have \( C = \alpha_i + \beta_i \), otherwise \( \xi_i \) can make the objective function \(-\infty\)

- Substitute these two equations back we get

\[
\max_{\alpha \geq 0, \beta \geq 0, C = \alpha + \beta} -\frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \varphi(x_i)^T \varphi(x_j) + \sum_i \alpha_i
\]
Support Vector Machines (dual)

Therefore, we get the following dual problem

\[
\max_{C \geq \alpha \geq 0} \left\{ -\frac{1}{2} \alpha^T Q \alpha + e^T \alpha \right\} := D(\alpha),
\]

where \( Q \) is an \( n \) by \( n \) matrix with \( Q_{ij} = y_i y_j \varphi(x_i)^T \varphi(x_j) \).

Based on the derivations, we know

1. Primal minimum = dual maximum (under slater’s condition)
2. Let \( \alpha^* \) be the dual solution and \( w^* \) be the primal solution, we have

\[
w^* = \sum_{i} y_i \alpha^*_i \varphi(x_i)
\]

We can solve the dual problem instead of the primal problem.
Kernel Trick

- Do not directly define $\varphi(\cdot)$
Kernel Trick

- Do not directly define $\varphi(\cdot)$
- Instead, define “kernel”

$$K(x_i, x_j) = \varphi(x_i)^T \varphi(x_j)$$

This is all we need to know for Kernel SVM!
Kernel Trick

- Do not directly define $\varphi(\cdot)$
- Instead, define “kernel”

$$K(x_i, x_j) = \varphi(x_i)^T \varphi(x_j)$$

This is all we need to know for Kernel SVM!

- Examples:
  - Gaussian kernel: $K(x_i, x_j) = e^{-\gamma \|x_i - x_j\|^2}$
  - Polynomial kernel: $K(x_i, x_j) = (\gamma x_i^T x_j + c)^d$
  - Other kernels for specific problems:
    - Graph kernels
      (Vishwanathan et al., “Graph Kernels”, JMLR, 2010)
    - Pyramid kernel for image matching
      (Grauman and Darrell, “The Pyramid Match Kernel: Discriminative Classification with Sets of Image Features”. In ICCV, 2005)
    - String kernel
Support Vector Machines (dual)

- Training: compute \( \alpha = [\alpha_1, \cdots, \alpha_n] \) by solving the quadratic optimization problem:

\[
\min_{0 \leq \alpha \leq C} \frac{1}{2} \alpha^T Q \alpha - e^T \alpha
\]

where \( Q_{ij} = K(x_i, x_j) \)

- Prediction: for a test data \( x \),

\[
\hat{w}^T \phi(x) = \sum_{i=1}^{n} y_i \alpha_i \phi(x_i)^T \phi(x) = \sum_{i=1}^{n} y_i \alpha_i K(x_i, x)
\]
Support Vector Machines (dual)

- Training: compute $\alpha = [\alpha_1, \cdots, \alpha_n]$ by solving the quadratic optimization problem:

$$\min_{0 \leq \alpha \leq C} \frac{1}{2} \alpha^T Q \alpha - e^T \alpha$$

where $Q_{ij} = K(x_i, x_j)$

- Prediction: for a test data $x$,

$$w^T \varphi(x) = \sum_{i=1}^{n} y_i \alpha_i \varphi(x_i)^T \varphi(x)$$

$$= \sum_{i=1}^{n} y_i \alpha_i K(x_i, x)$$
Kernel Ridge Regression

- Actually, this “kernel method” works for many different losses
Kernel Ridge Regression

• Actually, this “kernel method” works for many different losses
• Example: ridge regression

\[
\min_w \frac{1}{2} \| w \|^2 + \frac{1}{2} \sum_{i=1}^{n} (w^T \varphi(x_i) - y_i)^2
\]

• Dual problem:

\[
\min_\alpha \alpha^T Q \alpha + \| \alpha \|^2 - 2 \alpha^T y
\]
Scalability

- Challenge for solving kernel SVMs (for dataset with $n$ samples):
  - **Space**: $O(n^2)$ for storing the $n$-by-$n$ kernel matrix (can be reduced in some cases);
  - **Time**: $O(n^3)$ for computing the exact solution.

Good packages available:
- LIBSVM (can be called in scikit-learn)
- LIBLINEAR (for linear SVM, can be called in scikit-learn)
Scalability

- Challenge for solving kernel SVMs (for dataset with \( n \) samples):
  - **Space**: \( O(n^2) \) for storing the \( n \)-by-\( n \) kernel matrix (can be reduced in some cases);
  - **Time**: \( O(n^3) \) for computing the exact solution.

- Good packages available:
  - LIBSVM (can be called in scikit-learn)
  - LIBLINEAR (for linear SVM, can be called in scikit-learn)
Multiclass classification
Multiclass Learning

- $n$ data points, $L$ labels, $d$ features
- Input: training data $\{x_i, y_i\}_{i=1}^n$:
  - Each $x_i$ is a $d$ dimensional feature vector
  - Each $y_i \in \{1, \ldots, L\}$ is the corresponding label
  - Each training data belongs to one category
- Goal: find a function to predict the correct label

$$f(x) \approx y$$
Multi-label Problems

- $n$ data points, $L$ labels, $d$ features
- Input: training data $\{x_i, y_i\}^n_{i=1}$:
  - Each $x_i$ is a $d$ dimensional feature vector
  - Each $y_i \in \{0, 1\}^L$ is a label vector (or $Y_i \in \{1, 2, \ldots, L\}$)
    - Example: $y_i = [0, 0, 1, 0, 0, 1, 1]$ (or $Y_i = \{3, 6, 7\}$)
  - Each training data can belong to multiple categories
- Goal: Given a testing sample $x$, predict the correct labels

<table>
<thead>
<tr>
<th>Document 1</th>
<th>{Sports, Politics}</th>
</tr>
</thead>
<tbody>
<tr>
<td>Document 2</td>
<td>{Science, Politics}</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Document n</td>
<td>{Environment}</td>
</tr>
</tbody>
</table>
Multiclass: each row of $L$ has exact one “1”
Multilabel: each row of $L$ can have multiple ones
Reduction to binary classification

- Many algorithms for binary classification
- Idea: transform multi-class or multi-label problems to multiple binary classification problems
- Two approaches:
  - One versus All (OVA)
  - One versus One (OVO)
One Versus All (OVA)

- Multi-class/multi-label problems with \( L \) categories
- Build \( L \) different binary classifiers
- For the \( t \)-th classifier:
  - Positive samples: all the points in class \( t \) \( \{x_i : t \in y_i\} \)
  - Negative samples: all the points not in class \( t \) \( \{x_i : t \notin y_i\} \)
  - \( f_t(x) \): the decision value for the \( t \)-th classifier
    (larger \( f_t \) \( \Rightarrow \) higher probability that \( x \) in class \( t \))
- Prediction:
  \[
  f(x) = \arg \max_t f_t(x)
  \]
- Example: using SVM to train each binary classifier.
One Versus One (OVO)

- Multi-class/multi-label problems with $L$ categories
- Build $L(L - 1)$ different binary classifiers
- For the $(s, t)$-th classifier:
  - Positive samples: all the points in class $s$ ($\{x_i : s \in y_i\}$)
  - Negative samples: all the points in class $t$ ($\{x_i : t \in y_i\}$)
  - $f_{s,t}(x)$: the decision value for this classifier
    (larger $f_{s,t}(x) \Rightarrow$ label $s$ has higher probability than label $t$)
  - $f_{t,s}(x) = -f_{s,t}(x)$
- Prediction:
  $$f(x) = \arg \max_s \left( \sum_t f_{s,t}(x) \right)$$
- Example: using SVM to train each binary classifier.
OVA vs OVO

- Prediction accuracy: depends on datasets
- Computational time:
  - OVA needs to train $L$ classifiers
  - OVO needs to train $L(L - 1)/2$ classifiers
- Is OVA always faster than OVO?
OVA vs OVO

- Prediction accuracy: depends on datasets
- Computational time:
  - OVA needs to train $L$ classifiers
  - OVO needs to train $L(L - 1)/2$ classifiers
- Is OVA always faster than OVO?
  NO, depends on the time complexity of the binary classifier
  - If the binary classifier requires $O(n)$ time for $n$ samples:
    OVA and OVO have similar time complexity
  - If the binary classifier requires $O(n^{1.xx})$ time:
    OVO is faster than OVA
OVA vs OVO

- Prediction accuracy: depends on datasets
- Computational time:
  - OVA needs to train $L$ classifiers
  - OVO needs to train $L(L - 1)/2$ classifiers
- Is OVA always faster than OVO?
  - NO, depends on the time complexity of the binary classifier
    - If the binary classifier requires $O(n)$ time for $n$ samples:
      - OVA and OVO have similar time complexity
    - If the binary classifier requires $O(n^{1.xx})$ time:
      - OVO is faster than OVA
- LIBSVM (kernel SVM solver): OVO
- LIBLINEAR (linear SVM solver): OVA
Another approach for multi-class classification

- OVA and OVO: decompose the problem by labels
  But good binary classifiers may not imply good multi-class prediction.
Another approach for multi-class classification

- OVA and OVO: decompose the problem by labels
  
  But good binary classifiers may not imply good multi-class prediction.

- Design a **multi-class loss function** and solve a single optimization problem
Another approach for multi-class classification

- OVA and OVO: decompose the problem by labels
  But good binary classifiers may not imply good multi-class prediction.
- Design a multi-class loss function and solve a single optimization problem
- Minimize the regularized training error:

\[
\min_{w_1, \ldots, w_L} \sum_{i=1}^{n} \text{loss}(x_i, y_i) + \lambda \sum_{j=1, \ldots, L} w_j^T w_j
\]
Loss functions for multi-class classification

- Ranking based approaches: directly minimizes the ranking loss:
  - For multiclass classification, the score of \( y_i \) should be larger than other labels

Soft-max loss: measure the probability of predicting correct class
Loss functions for multi-class classification

- Ranking based approaches: directly *minimizes the ranking loss*:
  - For multiclass classification, the score of $y_i$ should be larger than other labels

- Soft-max loss:
  - measure the *probability* of predicting correct class
For simplicity, we assume a linear model

Model parameters: \( w_1, \ldots, w_L \)

For each data point \( x \), compute the decision value for each label:

\[
w_1^T x, \quad w_2^T x, \ldots, \quad w_L^T x
\]

Prediction:

\[
y = \arg \max_t w_t^T x
\]

For training data \( x_i, y_i \) is the true label, so we want

\[
y_i \approx \arg \max_t w_t^T x_i \quad \forall i
\]
Softmax

- The predicted score for each class:

\[ w_1^T x_i, \ w_2^T x_i, \ldots \]

- Loss for the \( i \)-th data is defined by

\[
- \log \left( \frac{e^{w_{y_i}^T x_i}}{\sum_j e^{w_j^T x_i}} \right)
\]

(Probability of choosing the correct label)

- Solve a single optimization problem

\[
\min_{w_1, \ldots, w_L} \sum_{i=1}^n - \log \left( \frac{e^{w_{y_i}^T x_i}}{\sum_j e^{w_j^T x_i}} \right) + \lambda \sum_j w_j^T w_j
\]
Weston-Watkins Formulation

- Proposed in Weston and Watkins, “Multi-class support vector machines”. In ESANN, 1999.

\[
\begin{align*}
\min_{\{w_t\},\{\xi_i^t\}} & \quad \frac{1}{2} \sum_{t=1}^{L} \|w_t\|^2 + C \sum_{i=1}^{n} \sum_{t=1}^{L} \xi_i^t \\
\text{s.t.} & \quad w_{y_i}^T x_i - w_t^T x_i \geq 1 - \xi_i^t, \quad \xi_i^t \geq 0 \quad \forall t \neq y_i, \quad \forall i = 1, \ldots, n
\end{align*}
\]

- If point \( i \) is in class \( y_i \), for any other labels \( (t \neq y_i) \), we want

\[
w_{y_i}^T x_i - w_t^T x_i \geq 1
\]

or we pay a penalty \( \xi_i^t \)

- Prediction:

\[
f(x) = \arg \max_t w_t^T x_i
\]
Crammer-Singer Formulation


\[
\min_{\{w_t\}, \{\xi_i^t\}} \frac{1}{2} \sum_{t=1}^{L} \|w_t\|^2 + C \sum_{i=1}^{n} \xi_i \\
\text{s.t. } w_{y_i}^T x_i - w_t^T x_i \geq 1 - \xi_i, \quad \forall t \neq y_i, \forall i = 1, \ldots, n \\
\xi_i \geq 0 \quad \forall i = 1, \ldots, n
\]

- If point \(i\) is in class \(y_i\), for any other labels \((t \neq y_i)\), we want

\[
w_{y_i}^T x_i - w_t^T x_i \geq 1
\]

- For each point \(i\), we only pay the largest penalty

- Prediction:

\[
f(x) = \arg \max_t w_t^T x_i
\]
Conclusions

- SVM, Kernel SVM, Kernel methods

Questions?