CS260: Machine Learning Algorithms

Lecture 8: Kernel Methods

Cho-Jui Hsieh
UCLA

Feb 4, 2019

Outline

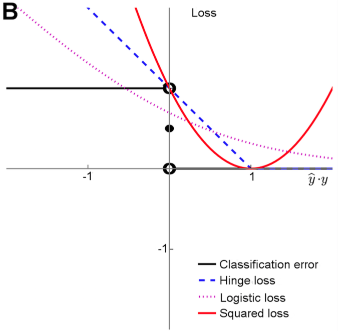
- Linear Support Vector Machines
- Nonlinear SVM, Kernel methods
- Multiclass classification

Support Vector Machines

- Given training examples $(\boldsymbol{x}_1, y_1), \cdots, (\boldsymbol{x}_n, y_n)$
  Consider binary classification: $y_i \in \{+1, -1\}$
- Linear Support Vector Machine (SVM):

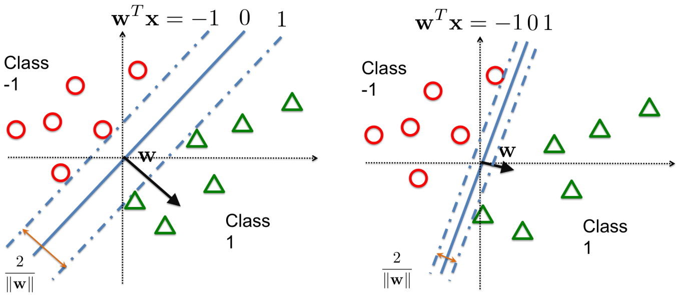
$$\arg\min_{\boldsymbol{w}} C \sum_{i=1}^{n} \max(1 - y_i \boldsymbol{w}^T \boldsymbol{x}_i, 0) + \frac{1}{2} \boldsymbol{w}^T \boldsymbol{w}$$

(hinge loss with L2 regularization)



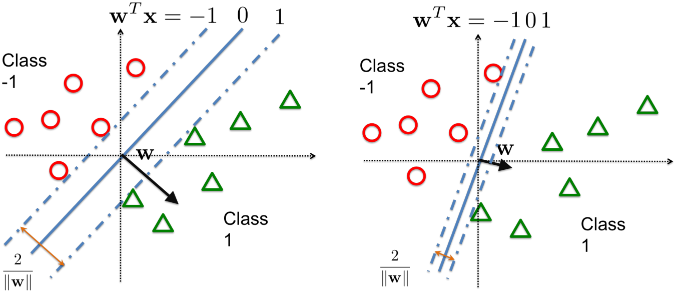
Support Vector Machines

- Goal: Find a hyperplane to separate these two classes of data: if $y_i = 1$, $\boldsymbol{w}^T \boldsymbol{x}_i \geq 1$; if $y_i = -1$, $\boldsymbol{w}^T \boldsymbol{x}_i \leq -1$.



Support Vector Machines

- Goal: Find a hyperplane to separate these two classes of data: if $y_i = 1$, $\boldsymbol{w}^T \boldsymbol{x}_i \geq 1$; if $y_i = -1$, $\boldsymbol{w}^T \boldsymbol{x}_i \leq -1$.



Prefer a hyperplane with maximum margin

Size of margin

- minimum of $\|\boldsymbol{x}\|$ such that $\boldsymbol{w}^T\boldsymbol{x} = 1$

Size of margin

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

Size of margin

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

Size of margin

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

Support Vector Machines (hard constraints)

- SVM primal problem (with hard constraints):

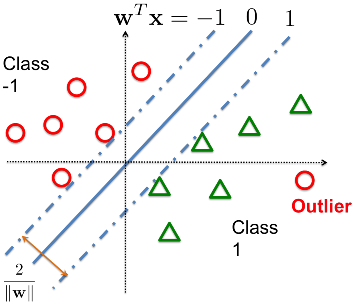
$$\min_{\boldsymbol{w}} \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w}$$
$$\text{s.t. } y_i(\boldsymbol{w}^T\boldsymbol{x}_i) \geq 1, i = 1, \ldots, n,$$

Support Vector Machines (hard constraints)

- SVM primal problem (with hard constraints):

$$\min_{\boldsymbol{w}} \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w}$$
$$\text{s.t. } y_i(\boldsymbol{w}^T\boldsymbol{x}_i) \geq 1, i = 1, \ldots, n,$$

- What if there are outliers?



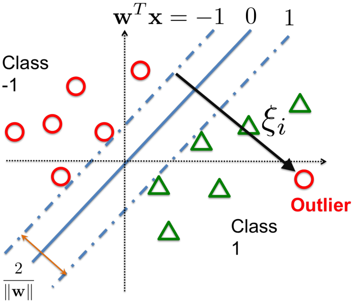
Support Vector Machines

- Given training data $\boldsymbol{x}_1, \cdots, \boldsymbol{x}_n \in \mathbb{R}^d$ with labels $y_i \in \{+1, -1\}$.
- SVM primal problem:

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

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

$$\xi_i \geq 0$$



Support Vector Machines

- SVM primal problem:

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

- Equivalent to

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

- Non-differentiable when $y_i\boldsymbol{w}^T\boldsymbol{x}_i = 1$ for some i

Stochastic Subgradient Method for SVM

- A subgradient of $\ell_i(\boldsymbol{w}) = \max(0, 1 - y_i \boldsymbol{w}^T \boldsymbol{x}_i)$:

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

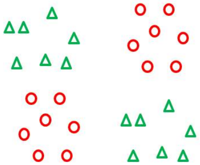
- Stochastic Subgradient descent for SVM:

For $t = 1, 2, \ldots$
  Randomly pick an index i
  If $y_i \boldsymbol{w}^T \boldsymbol{x}_i < 1$, then
    $\boldsymbol{w} \leftarrow (1 - \eta_t)\boldsymbol{w} + \eta_t n C y_i \boldsymbol{x}_i$
  Else (if $y_i \boldsymbol{w}^T \boldsymbol{x}_i \geq 1$):
    $\boldsymbol{w} \leftarrow (1 - \eta_t)\boldsymbol{w}$

Kernel SVM

Non-linearly separable problems

- What if the data is not linearly separable?



$$x \to \varphi(x) = \begin{bmatrix} x_1^2 \\ \sqrt{2}x_1x_2 \\ x_2^2 \end{bmatrix}$$

Solution: map data $\boldsymbol{x}_i$ to higher dimensional(maybe infinite) feature space $\varphi(x_i)$, where they are linearly separable.

SVM with nonlinear mapping

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

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

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

SVM with nonlinear mapping

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

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

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

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

Support Vector Machines (dual)

- Primal problem:

$$\min_{\boldsymbol{w},\boldsymbol{\xi}} \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_i \xi_i$$

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

- Equivalent to:

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

- Under certain condition (e.g., slater's condition), exchanging $\min, \max$ will not change the optimal solution:

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

Support Vector Machines (dual)

- Reorganize the equation:

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

- Now, for any given $\boldsymbol{\alpha}, \boldsymbol{\beta}$, the minimizer of $\boldsymbol{w}$ will satisfy

$$\frac{\partial L}{\partial \boldsymbol{w}} = \boldsymbol{w} - \sum_i \alpha_i y_i \varphi(\boldsymbol{x}_i) = 0 \quad \Rightarrow \boldsymbol{w}^* = \sum_i y_i \alpha_i \varphi(\boldsymbol{x}_i)$$

  Also, we have $C = \alpha_i + \beta_i$, otherwise ξ_i can make the objective function $-\infty$

- Substitue these two equations back we get

$$\max_{\boldsymbol{\alpha}\geq 0, \boldsymbol{\beta}\geq 0, C = \boldsymbol{\alpha}+\boldsymbol{\beta}} -\frac{1}{2}\sum_{i,j} \alpha_i \alpha_j y_i y_j \varphi(\boldsymbol{x}_i)^T \varphi(\boldsymbol{x}_j) + \sum_i \alpha_i$$

Support Vector Machines (dual)

- Therefore, we get the following dual problem

$$\max_{C \geq \boldsymbol{\alpha} \geq 0} \{-\frac{1}{2}\boldsymbol{\alpha}^T Q \boldsymbol{\alpha} + \boldsymbol{e}^T \boldsymbol{\alpha}\} := D(\boldsymbol{\alpha}),$$

  where Q is an n by n matrix with $Q_{ij} = y_i y_j \varphi(\boldsymbol{x}_i)^T \varphi(\boldsymbol{x}_j)$
- Based on the derivations, we know
  1. Primal minimum = dual maximum (under slater's condition)
  2. Let $\boldsymbol{\alpha}^*$ be the dual solution and $\boldsymbol{w}^*$ be the primal solution, we have

$$\boldsymbol{w}^* = \sum_i y_i \alpha_i^* \varphi(\boldsymbol{x}_i)$$

- 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"

$$K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \varphi(\boldsymbol{x}_i)^T \varphi(\boldsymbol{x}_j)$$

This is all we need to know for Kernel SVM!

Kernel Trick

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

$$K(\mathbf{x}_i, \mathbf{x}_j) = \varphi(\mathbf{x}_i)^T \varphi(\mathbf{x}_j)$$

  This is all we need to know for Kernel SVM!
- Examples:
  - Gaussian kernel: $K(\mathbf{x}_i, \mathbf{x}_j) = e^{-\gamma \|\mathbf{x}_i - \mathbf{x}_j\|^2}$
  - Polynomial kernel: $K(\mathbf{x}_i, \mathbf{x}_j) = (\gamma \mathbf{x}_i^T \mathbf{x}_j + c)^d$
  - 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
      (Lodhi et al., "Text classification using string kernels". JMLR, 2002).

Support Vector Machines (dual)

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

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

where $Q_{ij} = K(\boldsymbol{x}_i, \boldsymbol{x}_j)$

Support Vector Machines (dual)

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

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

  where $Q_{ij} = K(\boldsymbol{x}_i, \boldsymbol{x}_j)$
- Prediction: for a test data $\boldsymbol{x}$,

$$\begin{aligned} \boldsymbol{w}^T \varphi(\boldsymbol{x}) &= \sum_{i=1}^{n} y_i \alpha_i \varphi(\boldsymbol{x}_i)^T \varphi(\boldsymbol{x}) \\ &= \sum_{i=1}^{n} y_i \alpha_i K(\boldsymbol{x}_i, \boldsymbol{x}) \end{aligned}$$

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

$$\min_{\boldsymbol{w}} \frac{1}{2}\|\boldsymbol{w}\|^2 + \frac{1}{2}\sum_{i=1}^{n}(\boldsymbol{w}^T\varphi(\boldsymbol{x}_i) - y_i)^2$$

- Dual problem:

$$\min_{\boldsymbol{\alpha}} \boldsymbol{\alpha}^T Q\boldsymbol{\alpha} + \|\boldsymbol{\alpha}\|^2 - 2\boldsymbol{\alpha}^T\mathbf{y}$$

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.

Scalability

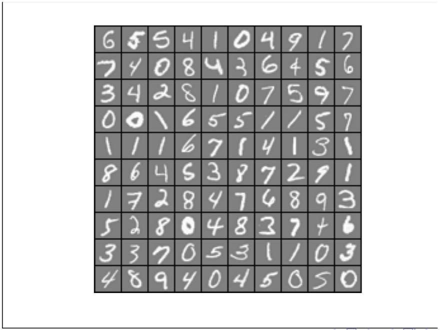
- Challenge for solving kernel SVMs (for dataset with n samples):
  - Space: $O(n^2)$ for storing the n-by-n kernel matrix (can be reduced in some cases);
  - Time: $O(n^3)$ for computing the exact solution.
- Good packages available:
  - LIBSVM (can be called in scikit-learn)

Multiclass classification

Multiclass Learning

- n data points, L labels, d features
- Input: training data $\{\boldsymbol{x}_i, y_i\}_{i=1}^n$:
  - Each $\boldsymbol{x}_i$ 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

$$f(\boldsymbol{x}) \approx y$$

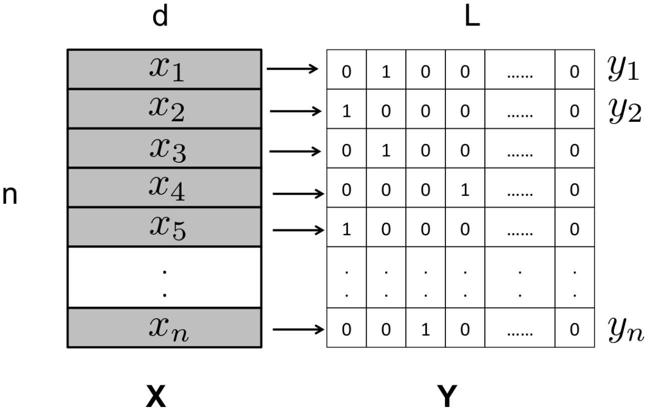


Multi-label Problems

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

Document 1	{Sports, Politics}
Document 2	{Science, Politics}
.	.
.	.
.	.
Document n	{Environment}

Illustration



- 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 ($\{\boldsymbol{x}_i : t \in \mathbf{y}_i\}$)
  - Negative samples: all the points not in class t ($\{\boldsymbol{x}_i : t \notin \mathbf{y}_i\}$)
  - $f_t(\boldsymbol{x})$: the decision value for the t-th classifier
    (larger $f_t \Rightarrow$ higher probability that $\boldsymbol{x}$ in class t)
- Prediction:

$$f(\boldsymbol{x}) = \arg\max_t f_t(\boldsymbol{x})$$

- 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 ($\{\boldsymbol{x}_i : s \in \mathbf{y}_i\}$)
  - Negative samples: all the points in class t ($\{\boldsymbol{x}_i : t \in \mathbf{y}_i\}$)
  - $f_{s,t}(\boldsymbol{x})$: the decision value for this classifier
    (larger $f_{s,t}(\boldsymbol{x}) \Rightarrow$ label s has higher probability than label t)
  - $f_{t,s}(\boldsymbol{x}) = -f_{s,t}(\boldsymbol{x})$
- Prediction:

$$f(\boldsymbol{x}) = \arg\max_{s} \left(\sum_{t} f_{s,t}(\boldsymbol{x}) \right)$$

- 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 in-sample error:

$$\min_{\boldsymbol{w}_1, \cdots, \boldsymbol{w}_L} \sum_{i=1}^{n} \text{loss}(\boldsymbol{x}_i, \mathbf{y}_i) + \lambda \sum_{j=1,\cdots,L} \boldsymbol{w}_j^T \boldsymbol{w}_j$$

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

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

Main idea

- For simplicity, we assume a linear model
- Model parameters: $\boldsymbol{w}_1, \ldots, \boldsymbol{w}_L$
- For each data point $\boldsymbol{x}$, compute the decision value for each label:

$$\boldsymbol{w}_1^T \boldsymbol{x}, \quad \boldsymbol{w}_2^T \boldsymbol{x}, \quad \ldots, \quad \boldsymbol{w}_L^T \boldsymbol{x}$$

- Prediction:

$$y = \arg\max_t \boldsymbol{w}_t^T \boldsymbol{x}$$

- For training data $\boldsymbol{x}_i$, y_i is the true label, so we want

$$y_i \approx \arg\max_t \boldsymbol{w}_t^T \boldsymbol{x}_i \quad \forall i$$

Softmax

- The predicted score for each class:

$$\boldsymbol{w}_1^T \boldsymbol{x}_i, \; \boldsymbol{w}_2^T \boldsymbol{x}_i, \; \cdots$$

- Loss for the i-th data is defined by

$$-\log \left(\frac{e^{\boldsymbol{w}_{y_i}^T \boldsymbol{x}_i}}{\sum_j e^{\boldsymbol{w}_j^T \boldsymbol{x}_i}} \right)$$

  (Probability of choosing the correct label)
- Solve a single optimization problem

$$\min_{\boldsymbol{w}_1,\cdots,\boldsymbol{w}_L} \sum_{i=1}^{n} -\log \left(\frac{e^{\boldsymbol{w}_{y_i}^T \boldsymbol{x}_i}}{\sum_j e^{\boldsymbol{w}_j^T \boldsymbol{x}_i}} \right) + \lambda \sum_j \boldsymbol{w}_j^T \boldsymbol{w}_j$$

Weston-Watkins Formulation

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

$$\min_{\{\boldsymbol{w}_t\},\{\xi_i^t\}} \quad \frac{1}{2}\sum_{t=1}^{L}\|\boldsymbol{w}_t\|^2 + C\sum_{i=1}^{n}\sum_{t=1}^{L}\xi_i^t$$

$$\text{s.t.} \quad \boldsymbol{w}_{y_i}^T\boldsymbol{x}_i - \boldsymbol{w}_t^T\boldsymbol{x}_i \geq 1 - \xi_i^t, \quad \xi_i^t \geq 0 \ \ \forall t \neq y_i, \ \forall i = 1, \dots, n$$

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

$$\boldsymbol{w}_{y_i}^T\boldsymbol{x}_i - \boldsymbol{w}_t^T\boldsymbol{x}_i \geq 1$$

  or we pay a penalty ξ_i^t
- Prediction:

$$f(\boldsymbol{x}) = \arg\max_t \boldsymbol{w}_t^T\boldsymbol{x}_i$$

Crammer-Singer Formulation

- Proposed in Carmmer and Singer, "On the algorithmic implementation of multiclass kernel-based vector machines". JMLR, 2001.

$$\min_{\{\boldsymbol{w}_t\},\{\xi_i^t\}} \quad \frac{1}{2}\sum_{t=1}^{L}\|\boldsymbol{w}_t\|^2 + C\sum_{i=1}^{n}\xi_i$$
$$\text{s.t.} \quad \boldsymbol{w}_{y_i}^T\boldsymbol{x}_i - \boldsymbol{w}_t^T\boldsymbol{x}_i \geq 1 - \xi_i, \quad \forall t \neq y_i, \ \forall i = 1, \ldots, n$$
$$\xi_i \geq 0 \ \ \forall i = 1, \ldots, n$$

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

$$\boldsymbol{w}_{y_i}^T\boldsymbol{x}_i - \boldsymbol{w}_t^T\boldsymbol{x}_i \geq 1$$

- For each point i, we only pay the largest penalty
- Prediction:

$$f(\boldsymbol{x}) = \arg\max_t \boldsymbol{w}_t^T\boldsymbol{x}_i$$

Conclusions

- SVM, Kernel SVM, Kernel methods

Questions?