13. Support Vector Machines

STA3142 Statistical Machine Learning

Kibok Lee

Assistant Professor of

Applied Statistics / Statistics and Data Science

Apr {9, 11}, 2024

연세대학교
YONSEI UNIVERSITY

Assignment 2

- Due **Friday 4/12, 11:59pm**
- Topics
  - (Math/Programming) Logistic Regression
  - (Math/Programming) Softmax Regression
  - (Math) Gaussian Discriminant Analysis
  - (Programming) Naïve Bayes for Spam Classification

- Please read the instruction carefully!
  - Submit one <u>pdf</u> and one <u>zip</u> file separately
  - Write your code only in the designated spaces
  - Do not import additional libraries
  - …
- If you feel difficult, consider to take **option 2**.

Midterm

- **Tuesday 4/23, 1:10pm — 2:50pm KST**
  - Please come here by 1:00pm!
  - In-person exam
- Closed book with **an A4-size cheat sheet**
  - You can print/write anything on **both side**.
- Coverage: Lec 6—13
  - True / False, multiple choice, math
- Short practice midterm will be out.
  - To be familiar with the type of midterm questions
  - # questions is about a half of the actual exam
  - **No solution will be provided**

Midterm Coverage

- 4,5: Linear Algebra & Probability Review
  - Not main topics, but you should be familiar with them.
  - Some contents (that we feel difficult) can be given FYI.
- 6,7. Linear Regression (and Other Topics)
- 8. Logistic/Softmax Regression
- 9. Generative Classifiers
- 10. Other Classifiers
- 11. Regularization and Validation
- 12. Kernel Methods
- 13. Support Vector Machines

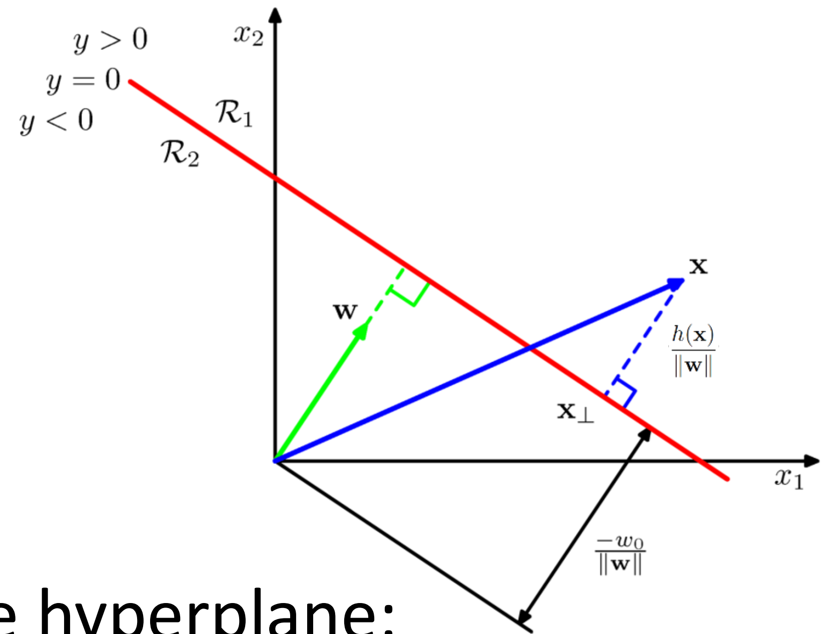
Outline

- Hard-Margin SVM
- Soft-Margin SVM
- Primal Optimization
- Multiclass SVM
- Kernel SVM (next)

Support Vector Machines

Linear Discriminant Function

- $h(\mathbf{x}) = \mathbf{w}^T \phi(\mathbf{x}) + b$



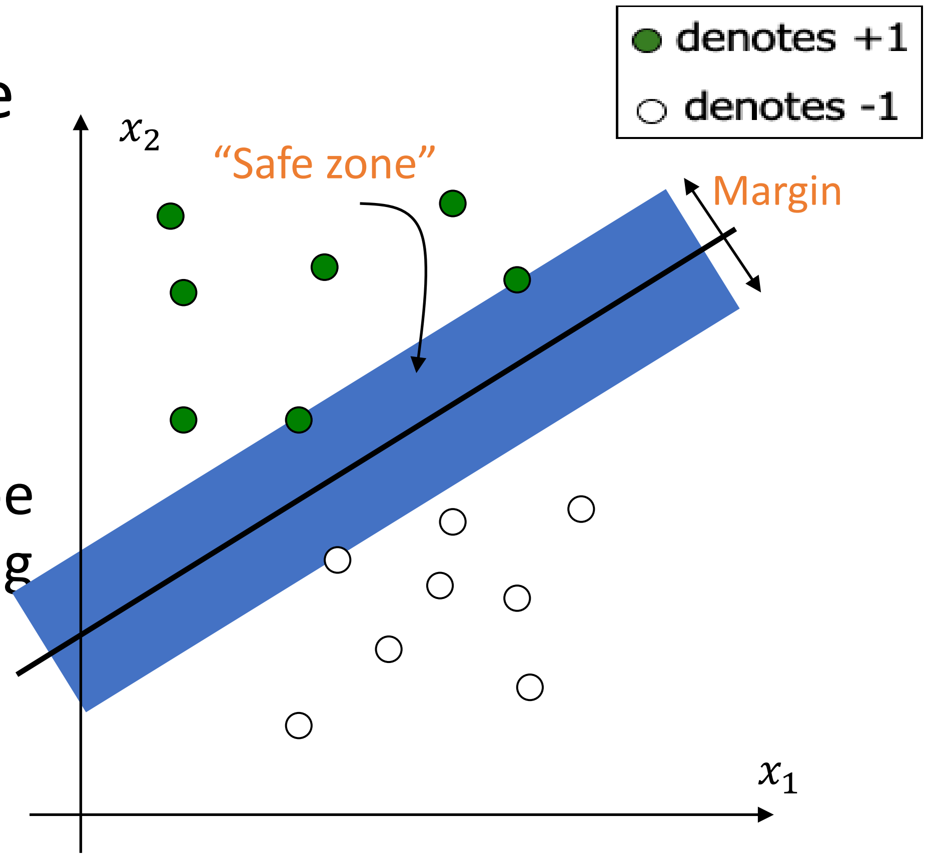
- Classification rule:
  - $y = +1$ if $h(\mathbf{x}) \geq 0$
  - $y = -1$ otherwise
- Decision boundary is the hyperplane:

$$h(\mathbf{x}) = \mathbf{w}^T \phi(\mathbf{x}) + b = 0$$

  - $\mathbf{w}$ determines the direction.
  - b determines the offset.

Maximum Margin Classifier

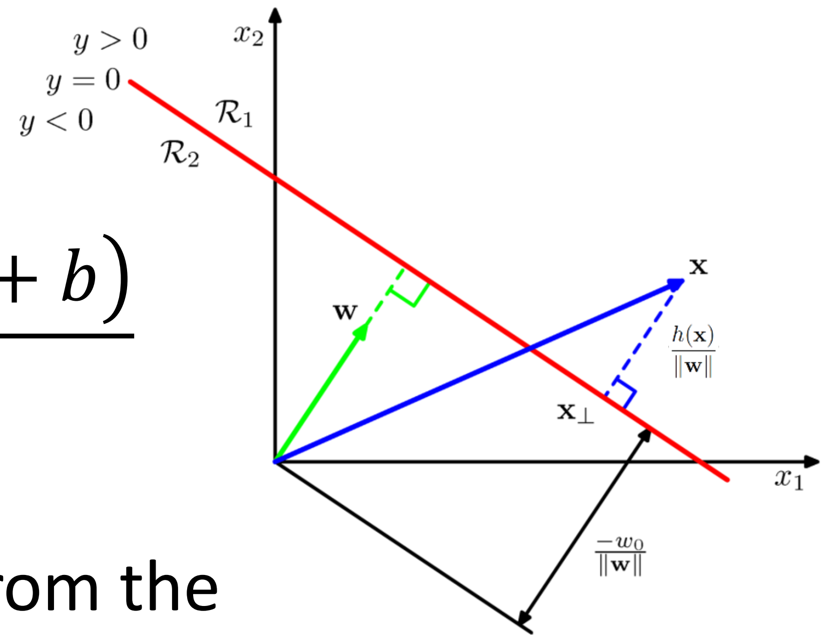
- A linear classifier with the maximum **margin** is considered to be a good classifier.
  - **Margin** is the width by which the boundary can be extended without touching any data point
- Why is the maximum margin good?
  - **Robust to outliers**, i.e., strong generalization ability.



SVM: Formulation

- Distance from $\phi(\mathbf{x}^{(n)})$ to the hyperplane $\mathbf{w}^T\phi(\mathbf{x}) + b = 0$:

$$\frac{y^{(n)}\left(\mathbf{w}^T\phi\left(\mathbf{x}^{(n)}\right) + b\right)}{\|\mathbf{w}\|}$$



- Margin is the distance from the decision boundary to the closest data
  - Assuming that all data are linearly separable

$$\min_n \frac{y^{(n)}\left(\mathbf{w}^T\phi\left(\mathbf{x}^{(n)}\right) + b\right)}{\|\mathbf{w}\|}$$

SVM: Derivation

- Maximize the distance from the decision boundary to the closest data:

$$\max_{\mathbf{w},b}\left[\frac{1}{\|\mathbf{w}\|}\min_{n} y^{(n)}\left(\mathbf{w}^T\phi\left(\mathbf{x}^{(n)}\right)+b\right)\right]$$

- For a solution, rescaling it gives us another solution:

$$\mathbf{w} \leftarrow c\mathbf{w}, b \leftarrow cb$$

- We can rescale $\mathbf{w}$ and b such that:

$$y^{(n)}\left(\mathbf{w}^T\phi\left(\mathbf{x}^{(n)}\right)+b\right) \geq 1, \forall n = 1, \dots, N$$

  - where the equality holds when n is the argmin, i.e.,

$$\min_{n} y^{(n)}\left(\mathbf{w}^T\phi\left(\mathbf{x}^{(n)}\right)+b\right) = 1$$

$$\max_{\mathbf{w},b}\frac{1}{\|\mathbf{w}\|}$$

- $\max_{\mathbf{w},b}\frac{1}{\|\mathbf{w}\|}$ is equivalent to $\min_{\mathbf{w},b}\frac{1}{2}\|\mathbf{w}\|^2$ Why?

SVM: Derivation

- The optimization problem is

$$\min_{\mathbf{w},b} \frac{1}{2} \|\mathbf{w}\|^2$$

  - subject to

$$y^{(n)}\left(\mathbf{w}^T \phi\left(\mathbf{x}^{(n)}\right) + b\right) \geq 1, \forall n = 1, \ldots, N$$

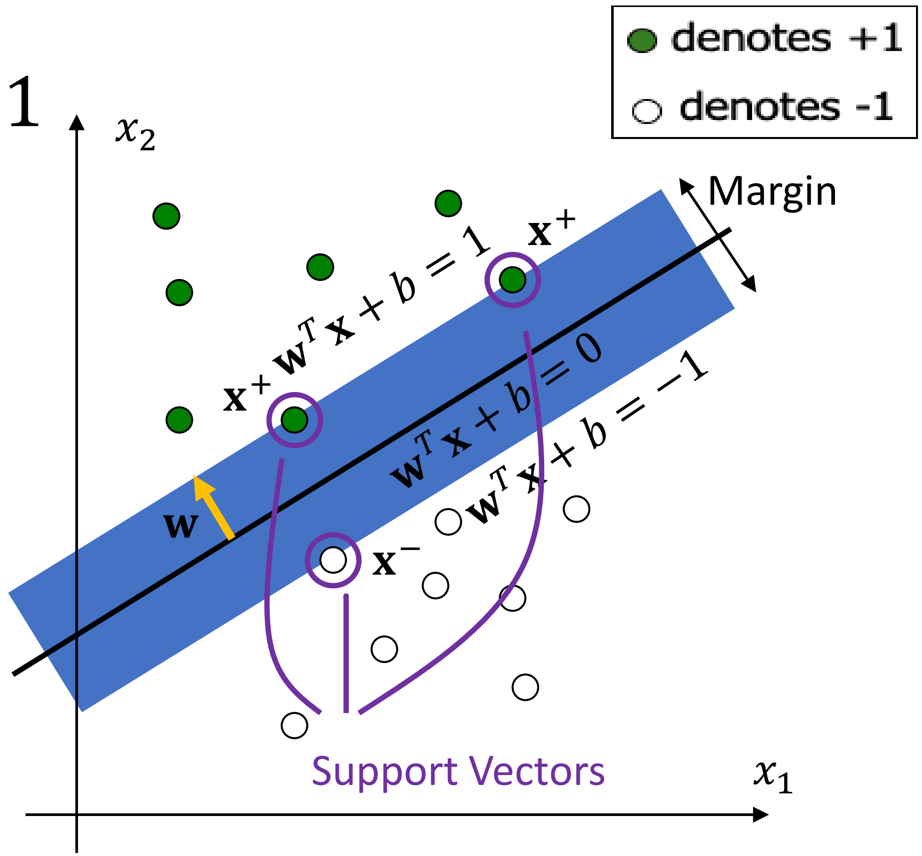
- Derivation holds because

$$\min_n y^{(n)}\left(\mathbf{w}^T \phi\left(\mathbf{x}^{(n)}\right) + b\right) = 1$$

- Why?

SVM: Derivation

- $y^{(n)}\left(\mathbf{w}^T\phi\left(\mathbf{x}^{(n)}\right)+b\right)\geq 1$ is equivalent to
  - $\left(\mathbf{w}^T\phi\left(\mathbf{x}^{(n)}\right)+b\geq 1\right)$ $\wedge\left(y^{(n)}=1\right)$
  - $\left(\mathbf{w}^T\phi\left(\mathbf{x}^{(n)}\right)+b\leq -1\right)$ $\wedge\left(y^{(n)}=-1\right)$
- Let $\phi(\mathbf{x}^+)$ and $\phi(\mathbf{x}^-)$ be the closest positive and negative samples.

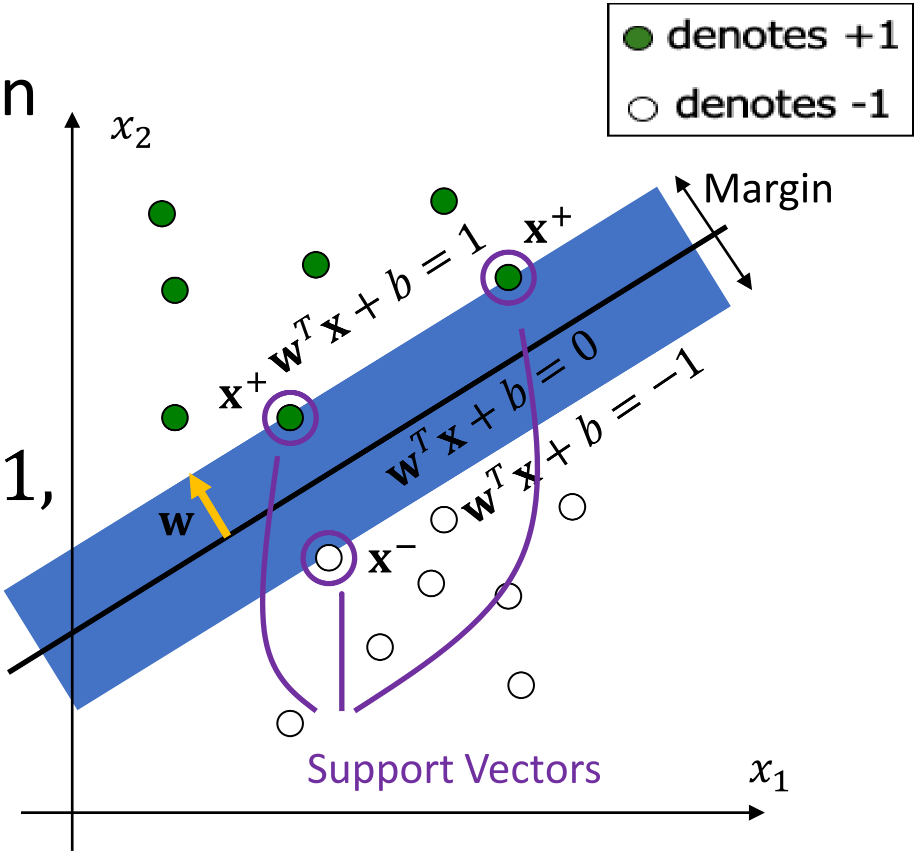


- Then, $\mathbf{w}^T\phi(\mathbf{x}^+)+b=1$, $\mathbf{w}^T\phi(\mathbf{x}^-)+b=-1$.
- We can optimize for those samples only:

$$\min_n y^{(n)}\left(\mathbf{w}^T\phi\left(\mathbf{x}^{(n)}\right)+b\right)=1$$

SVM: Another Derivation

- For a discriminant function $h(\mathbf{x}) = \mathbf{w}^T \phi(\mathbf{x}) + b$,
- Set a proper scale of $\mathbf{w}$ and b such that
- $y^{(n)}\left(\mathbf{w}^T \phi\left(\mathbf{x}^{(n)}\right) + b\right) \geq 1$, $\forall n = 1, \ldots, N$
- Let $\phi(\mathbf{x}^+)$ and $\phi(\mathbf{x}^-)$ be the closest positive and negative samples.



- Then, $\mathbf{w}^T \phi(\mathbf{x}^+) + b = 1$, $\mathbf{w}^T \phi(\mathbf{x}^-) + b = -1$.
- (Continue)

SVM: Another Derivation

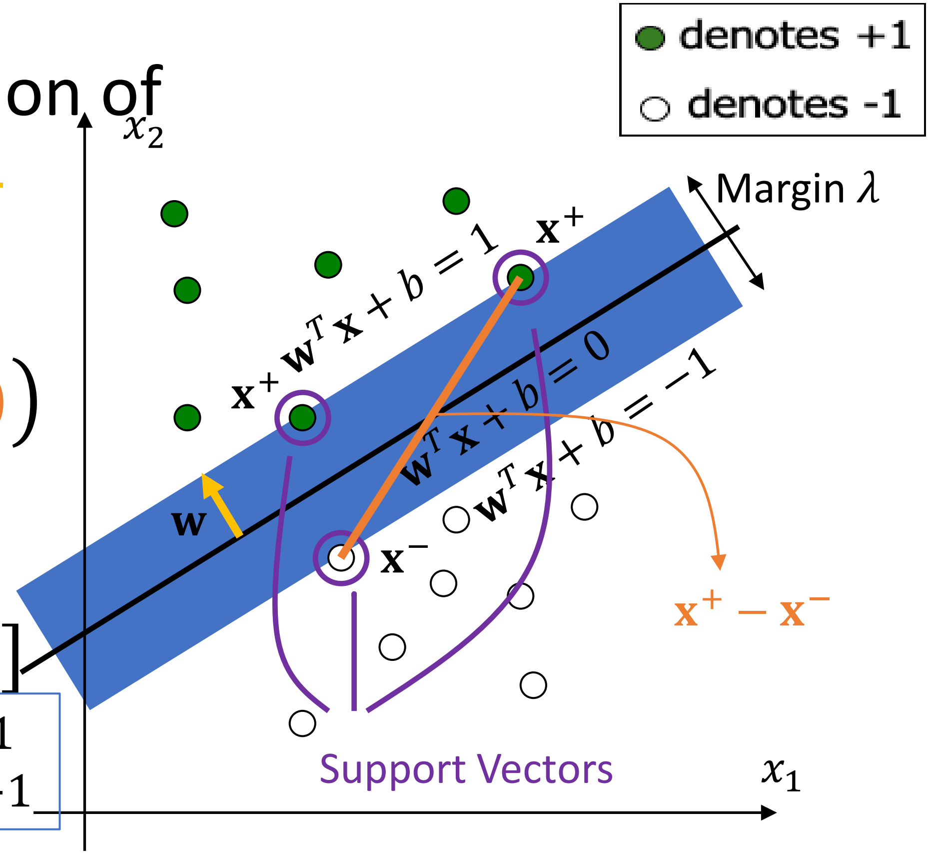
- The length of the projection of $\phi(\mathbf{x}^+) - \phi(\mathbf{x}^-)$ onto $\frac{\mathbf{w}}{\|\mathbf{w}\|}$ is the margin λ:

$$\lambda = \frac{\mathbf{w}^T}{\|\mathbf{w}\|}\left(\phi(\mathbf{x}^+) - \phi(\mathbf{x}^-)\right)$$

$$= \frac{1}{\|\mathbf{w}\|}\left[\left(\mathbf{w}^T\phi(\mathbf{x}^+) + b\right) - \left(\mathbf{w}^T\phi(\mathbf{x}^-) + b\right)\right]$$

$$= \frac{2}{\|\mathbf{w}\|}$$

$$\mathbf{w}^T\phi(\mathbf{x}^+) + b = 1$$
$$\mathbf{w}^T\phi(\mathbf{x}^-) + b = -1$$



- We maximize the margin $\lambda = \frac{2}{\|\mathbf{w}\|}$
- $\max_{\mathbf{w},b} \lambda = \frac{2}{\|\mathbf{w}\|}$ is equivalent to $\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|^2$

Support Vector Machines (SVM)

- Objective function:

$$\min_{\mathbf{w},b} \frac{1}{2} \|\mathbf{w}\|^2$$

  - subject to

$$y^{(n)}\left(\mathbf{w}^T \phi\left(\mathbf{x}^{(n)}\right) + b\right) \geq 1, \forall n = 1, \dots, N$$

- This is a constrained optimization problem.
  - We can solve this using **Lagrange multipliers**. (convex optimization)
  - We will discuss this later.

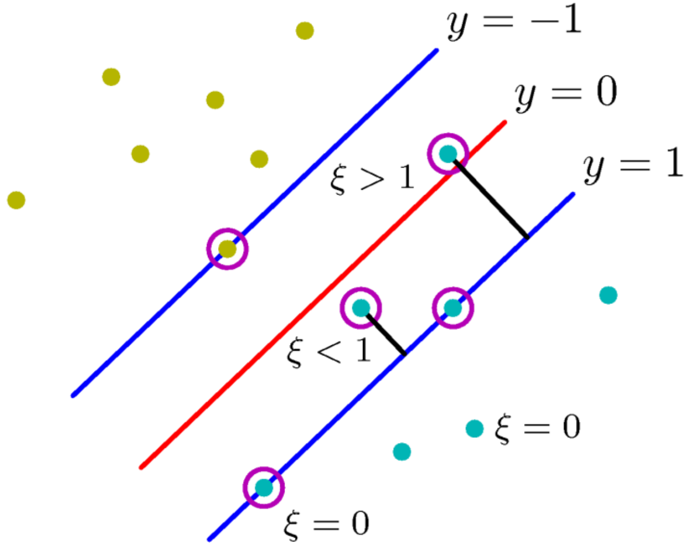
Soft-Margin SVM

- (Hard-margin) SVM requires an assumption that all data are linearly separable.

$$y^{(n)}\left(\mathbf{w}^T\phi\left(\mathbf{x}^{(n)}\right)+b\right)\geq 1$$

- Soft-margin SVM introduces slack variables $\xi^{(n)}$ for each data point:

$$y^{(n)}\left(\mathbf{w}^T\phi\left(\mathbf{x}^{(n)}\right)+b\right)\geq 1-\xi^{(n)}$$



Soft-Margin SVM

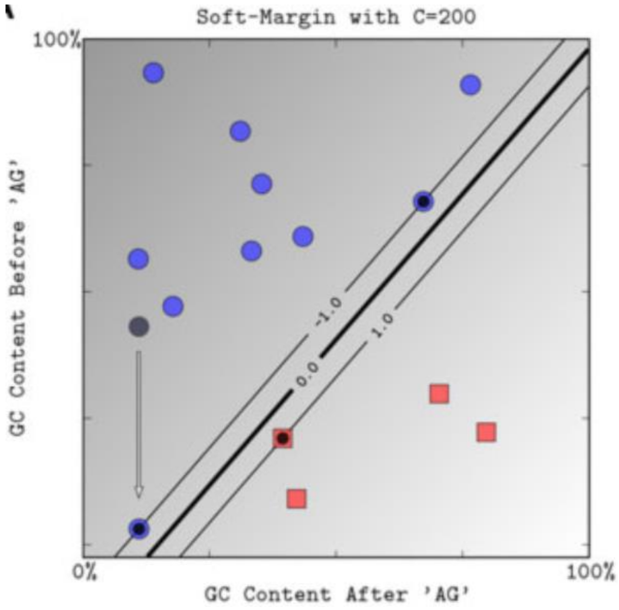
- We would like to maximize the margin and minimize slack variables simultaneously.
- Primal optimization for soft-margin SVM:

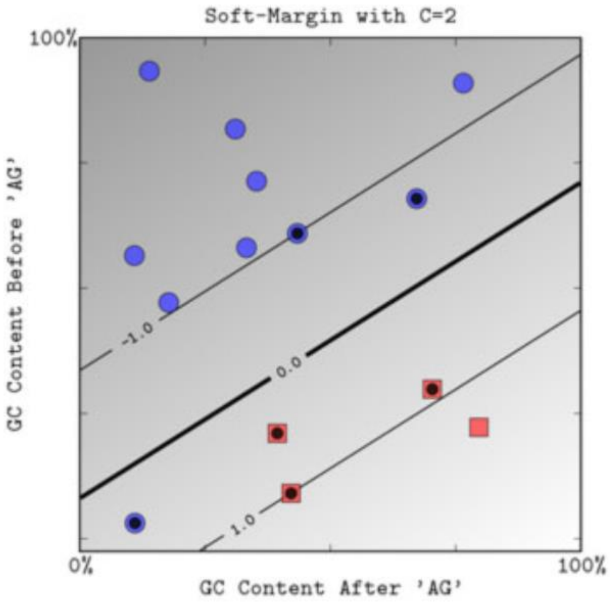
$$\min_{\mathbf{w},b,\boldsymbol{\xi}} C \sum_{n=1}^{N} \xi^{(n)} + \frac{1}{2}\|\mathbf{w}\|^2$$

  - subject to $y^{(n)} h\left(\mathbf{x}^{(n)}\right) \geq 1 - \xi^{(n)}, \forall n$
    $\xi^{(n)} \geq 0, \forall n$
- Larger slack cost $C \geq 0$ penalizes slack variables $\boldsymbol{\xi} = \left[\xi^{(1)}, \dots, \xi^{(N)}\right]$ more.
  - An empirically good range is $C \in [10^{-1}, 10^{3}]$

Soft-Margin SVM

- Introducing a small slack can give a better margin.





Primal Optimization

Primal Optimization

- Soft-margin SVM:

$$\min_{\mathbf{w},b,\xi} C \sum_{n=1}^{N} \xi^{(n)} + \frac{1}{2}\|\mathbf{w}\|^2$$

  - subject to $y^{(n)} h\left(\mathbf{x}^{(n)}\right) \geq 1 - \xi^{(n)}, \forall n$

    $\xi^{(n)} \geq 0, \forall n$

- This is a constrained optimization problem.
  - We can solve this using **Lagrange multipliers**. (convex optimization)
  - Lagrange multipliers convert the constraint into a penalty function.

Primal Optimization

- Soft-margin SVM:

$$\min_{\mathbf{w},b,\boldsymbol{\xi}} C \sum_{n=1}^{N} \xi^{(n)} + \frac{1}{2} \|\mathbf{w}\|^2$$

  - subject to $y^{(n)} h\left(\mathbf{x}^{(n)}\right) \geq 1 - \xi^{(n)}, \forall n$
    $\xi^{(n)} \geq 0, \forall n$

- Merging two constraints in one inequality:

$$\begin{aligned} y^{(n)} h\left(\mathbf{x}^{(n)}\right) &\geq 1 - \xi^{(n)}, \forall n \\ \xi^{(n)} &\geq 0, \forall n \end{aligned} \quad \Rightarrow \quad \xi^{(n)} \geq \max\left(0, 1 - y^{(n)} h\left(\mathbf{x}^{(n)}\right)\right)$$

Primal Optimization

- Soft-margin SVM:

$$\min_{\mathbf{w},b,\boldsymbol{\xi}} C \sum_{n=1}^{N} \xi^{(n)} + \frac{1}{2}\|\mathbf{w}\|^2$$

$$\geq \min_{\mathbf{w},b} C \sum_{n=1}^{N} \max\left(0, 1 - y^{(n)} h\left(\mathbf{x}^{(n)}\right)\right) + \frac{1}{2}\|\mathbf{w}\|^2$$

- When equality holds for all n, all constraints are satisfied, and the objective is minimized.
  - i.e., we apply **partial optimization** with respect to $\boldsymbol{\xi}$ by taking $\xi^{(n)} = \max\left(0, 1 - y^{(n)} h\left(\mathbf{x}^{(n)}\right)\right), \forall n$
  - This is valid because we optimize a **convex problem**.

Primal Optimization

- Soft-margin SVM:

$$\min_{\mathbf{w},b,\xi} C \sum_{n=1}^{N} \xi^{(n)} + \frac{1}{2}\|\mathbf{w}\|^2$$

  - subject to $y^{(n)} h\left(\mathbf{x}^{(n)}\right) \geq 1 - \xi^{(n)}, \forall n$

$$\xi^{(n)} \geq 0, \forall n$$

- **Lagrangian formulation**:

$$\min_{\mathbf{w},b} C \sum_{n=1}^{N} \max\left(0, 1 - y^{(n)} h\left(\mathbf{x}^{(n)}\right)\right) + \frac{1}{2}\|\mathbf{w}\|^2$$

  - This can be optimized using gradient-based methods!
    - Batch gradient descent (BGD)
    - Stochastic gradient descent (SGD)

Primal Optimization using BGD

- Computing the (sub)gradient with respect $\mathbf{w}$ and b:

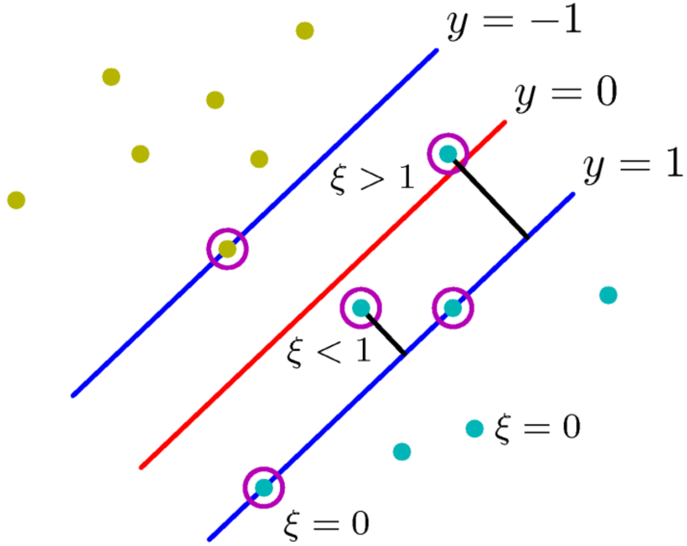
$$\nabla_{\mathbf{w}}\mathcal{L} = -C\sum_{n=1}^{N} y^{(n)}\phi\left(\mathbf{x}^{(n)}\right) I\left(1 - y^{(n)}h\left(\mathbf{x}^{(n)}\right) \geq 0\right) + \mathbf{w}$$

$$\nabla_{b}\mathcal{L} = -C\sum_{n=1}^{N} y^{(n)} I\left(1 - y^{(n)}h\left(\mathbf{x}^{(n)}\right) \geq 0\right)$$

- We can derive the SGD update rule similarly.

Support Vectors

- In SVM, only **a few training data** that have a margin of 1 or less actually affect the final solution $(\mathbf{w}, b)$.
  - These are called **support vectors (SVs)**.
- This is a nice property when kernelizing the method, as kernelization requires to memorize training data used to compute the solution.



Multiclass SVM

Multiclass SVM

- There are multiple ways to extend SVM for multiclass classification.
  - One-versus-rest (OVR) computes discriminant function of shape (N, K)
  - One-versus-one (OVO) computes discriminant functions (for all pairs) of shape $(N, K(K-1)/2)$
- sklearn.svm.SVC

class `sklearn.svm.SVC`(**, C=1.0, kernel='rbf', degree=3, gamma='scale', coef0=0.0, shrinking=True, probability=False, tol=0.001, cache_size=200, class_weight=None, verbose=False, max_iter=-1, decision_function_shape='ovr', break_ties=False, random_state=None*) [source]

decision_function_shape : *{'ovo', 'ovr'}, default='ovr'*

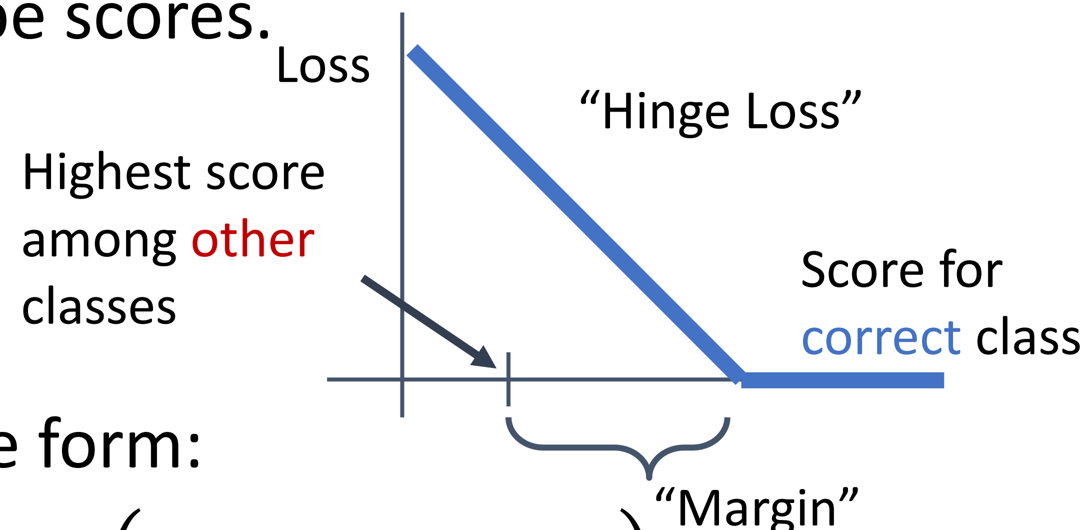
Whether to return a one-vs-rest ('ovr') decision function of shape (n_samples, n_classes) as all other classifiers, or the original one-vs-one ('ovo') decision function of libsvm which has shape (n_samples, n_classes * (n_classes - 1) / 2). However, note that internally, one-vs-one ('ovo') is always used as a multi-class strategy to train models; an ovr matrix is only constructed from the ovo matrix. The parameter is ignored for binary classification.

Multiclass SVM

- There are multiple ways to extend SVM for multiclass classification.
  - One-versus-rest (OVR) computes discriminant function of shape (N, K)
  - One-versus-one (OVO) computes discriminant functions (for all pairs) of shape $(N, K(K-1)/2)$
- $K = 2$: $y^{(n)} h\left(\mathbf{x}^{(n)}\right) \geq 1 - \xi^{(n)}$
- $K \geq 2$: $h_{y^{(n)}}\left(\mathbf{x}^{(n)}\right) - h_j\left(\mathbf{x}^{(n)}\right) \geq 1 - \xi_j^{(n)}, \forall j \neq y^{(n)}$
- Similar to softmax regression, we can derive multiclass SVM loss from the OVR formulation.

Multiclass SVM Loss

- One-versus-rest (OVR): The score of the correct class should be higher than all the other scores.
- Let $\mathbf{s} = h(\mathbf{x}) \in \mathbb{R}^K$ be scores.



- The SVM loss has the form:

$$L_i = \sum_{j \neq y^{(i)}} \max\left(0, s_j - s_{y^{(i)}} + 1\right)$$

Slide adapted from Justin Johnson

Multiclass SVM Loss



cat	**3.2**	1.3	2.2
car	5.1	**4.9**	2.5
frog	-1.7	2.0	**-3.1**

Classifier scores:

$$\mathbf{s} = h(\mathbf{x}) \in \mathbb{R}^K$$

The SVM loss has the form:

$$L_i = \sum_{j \neq y^{(i)}} \max\left(0, s_j - s_{y^{(i)}} + 1\right)$$

Multiclass SVM Loss

Classifier scores:

$$\mathbf{s} = h(\mathbf{x}) \in \mathbb{R}^K$$

			
cat	**3.2**	1.3	2.2
car	5.1	**4.9**	2.5
frog	-1.7	2.0	**-3.1**
Loss	2.9		

The SVM loss has the form:

$$L_i = \sum_{j \neq y^{(i)}} \max\left(0, s_j - s_{y^{(i)}} + 1\right)$$

= max(0, 5.1 - 3.2 + 1)
  + max(0, -1.7 - 3.2 + 1)
= max(0, 2.9) + max(0, -3.9)
= 2.9 + 0
= 2.9

Multiclass SVM Loss

cat	**3.2**	1.3	2.2
car	5.1	**4.9**	2.5
frog	-1.7	2.0	**-3.1**
Loss	2.9	0	

Classifier scores:

$$\mathbf{s} = h(\mathbf{x}) \in \mathbb{R}^K$$

The SVM loss has the form:

$$L_i = \sum_{j \neq y^{(i)}} \max\left(0, s_j - s_{y^{(i)}} + 1\right)$$

= max(0, 1.3 - 4.9 + 1)
+max(0, 2.0 - 4.9 + 1)
= max(0, -2.6) + max(0, -1.9)
= 0 + 0
= 0

Multiclass SVM Loss

cat	**3.2**	1.3	2.2
car	5.1	**4.9**	2.5
frog	-1.7	2.0	**-3.1**
Loss	2.9	0	12.9

Classifier scores:

$$\mathbf{s} = h(\mathbf{x}) \in \mathbb{R}^K$$

The SVM loss has the form:

$$L_i = \sum_{j \neq y^{(i)}} \max\left(0, s_j - s_{y^{(i)}} + 1\right)$$

= max(0, 2.2 - (-3.1) + 1)
+max(0, 2.5 - (-3.1) + 1)
= max(0, 6.3) + max(0, 6.6)
= 6.3 + 6.6
= 12.9

Multiclass SVM Loss

cat	**3.2**	1.3	2.2
car	5.1	**4.9**	2.5
frog	-1.7	2.0	**-3.1**
Loss	2.9	0	12.9

Classifier scores:

$$\mathbf{s} = h(\mathbf{x}) \in \mathbb{R}^K$$

The SVM loss has the form:

$$L_i = \sum_{j \neq y^{(i)}} \max\left(0, s_j - s_{y^{(i)}} + 1\right)$$

Avg loss over the dataset:

L = (2.9 + 0.0 + 12.9) / 3
= 5.27

Dual Optimization

Dual Optimization

- **Primal** optimization requires a direct access to feature mappings $\phi(\mathbf{x})$.
- We can kernelize SVM to remove explicit $\phi(\mathbf{x})$.
  - This formulation is called **dual** formulation.
  - In this case, you can use any kernel function, such as polynomial, radial basis function (RBF), and so on.
- With dual variables $a^{(n)}$, we have the followings:

$$\mathbf{w} = \sum_{n=1}^{N} a^{(n)} y^{(n)} \phi\left(\mathbf{x}^{(n)}\right)$$

$$h(\mathbf{x}) = \mathbf{w}^T \phi(\mathbf{x}) + b = \sum_{n=1}^{N} a^{(n)} y^{(n)} k\left(\mathbf{x}, \mathbf{x}^{(n)}\right) + b$$

Kernelizing Hard-Margin SVM

- Objective function:

$$\min_{\mathbf{w},b} \frac{1}{2} \|\mathbf{w}\|^2$$

  - subject to
    $$y^{(n)}\left(\mathbf{w}^T \phi\left(\mathbf{x}^{(n)}\right) + b\right) \geq 1, \forall n = 1, \dots, N$$

- This is a constrained optimization problem.
  - We can solve this using **Lagrange multipliers**. (convex optimization)
  - Kernelization can naturally be done by deriving dual optimization problem.

Next: Constrained Optimization, Kernel SVM