Support Vector Machines 2; Multiclass Classification

Linear

Outline

- SVM Review
- Support vectors
- Multiclass linear classificaiton

SVM Motivation

Tabular Data

- The support vector machine (SVM) is one of the best ``off-the-shelf" classification methods.
- " … looking at Kaggle challenges that are not related to vision or sequential tasks, gradient boosting, random forests, or SVMs are winning most of the competitions" (Klambauer et al., NeurIPS 2017)
- "On 75 small datasets with less than 1000 data points, random forests and SVMs outperform [FNNs]. On 46 larger datasets with at least 1000 data points, SNNs show the highest performance followed by SVMs and random forests (Klambauer et al., NeurIPS 2017)
- "We evaluate 179 classifiers arising from 17 families ... We use 121 data sets ... The classifiers most likely to be the best are the random forest (RF) versions ... However, the difference is not statistically significant with the second best, the SVM with Gaussian kernel implemented in C using LibSVM" (Delgado et al., JMLR 2014)

OSM Hyperplane

- The SVM is obtained by kernelizing the OSM hyperplane (equivalently, regularized ERM with the hinge loss).

$$\begin{aligned} \min_{\boldsymbol{w},b,\xi} \quad & \frac{1}{2}\|\boldsymbol{w}\|^2 + \frac{C}{n}\sum_{i=1}^{n}\xi_i \qquad \text{(OSM)} \\ \text{s.t.} \quad & y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b) \geq 1 - \xi_i \quad \forall i \\ & \xi_i \geq 0 \quad \forall i \end{aligned}$$

$\downarrow$

(w^*, b^*, ξ^*)

$x \longmapsto \text{sign}\{(w^*)^T x + b^*\}$

Poll

The number of optimization variables in the dual optimization problem (associated to the OSM hyperplane) is on the order of

(A) d, the original input dimension

(B) m, the dimensions of the output of the feature map Φ

(C) n, the number of training data points

(D) None of the above

OSM Dual Formulation

- The OSM hyperplane can be kernelized by solving the dual

$$\alpha = \begin{bmatrix} \alpha_1 \\ \vdots \\ \alpha_n \end{bmatrix}$$

$$\begin{aligned} \max_{\boldsymbol{\alpha}} \quad & -\frac{1}{2}\sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j \langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle + \sum_{i=1}^{n} \alpha_i \\ \text{s.t} \quad & \sum_i \alpha_i y_i = 0 \\ & 0 \le \alpha_i \le \frac{C}{n}, \quad \forall i = 1, \ldots, n \end{aligned}$$

- The classifier is expressed

$$f(\boldsymbol{x}) = \text{sign}\left\{ \sum_{i=1}^{n} \alpha_i^* y_i \langle \boldsymbol{x}_i, \boldsymbol{x} \rangle + b^* \right\}$$

- Taking any j with $0 < \alpha_j^* < \frac{C}{n}$, the offset is given by

$$b^* = y_j - \sum_{i=1}^{n} \alpha_i^* y_i \langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle$$

The Support Vector Machine

- Let k be an inner product/SPD kernel $k(x, x') = \langle \Phi(x), \Phi(x') \rangle$
- The *support vector machine* is the classifier obtained by solving

$$\begin{aligned} \max_{\boldsymbol{\alpha}} \quad & -\frac{1}{2}\sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j k(\boldsymbol{x}_i, \boldsymbol{x}_j) + \sum_{i=1}^{n} \alpha_i \\ \text{s.t} \quad & \sum_i \alpha_i y_i = 0 \\ & 0 \le \alpha_i \le \frac{C}{n}, \quad \forall i = 1, \ldots, n \end{aligned}$$

- The classifier is expressed

$$f(\boldsymbol{x}) = \text{sign}\left\{ \sum_{i=1}^{n} \alpha_i^* y_i k(\boldsymbol{x}_i, \boldsymbol{x}) + b^* \right\}$$

where $b^* = y_j - \sum_{i=1}^{n} \alpha_i^* y_i k(\boldsymbol{x}_i, \boldsymbol{x}_j)$ for any j with $0 < \alpha_j^* < \frac{C}{n}$.

Support Vectors

- From complementary slackness,

$$\forall i \quad \alpha_i^* \underbrace{\left(1 - \xi_i^* - y_i\left(\langle w^*, x_i\rangle + b^*\right)\right)}_{g_i} = 0$$

- If $\boldsymbol{x}_i$ satisfies

$$y_i\left(\langle w^*, x_i\rangle + b^*\right) = 1 - \xi_i^*$$

we call $\boldsymbol{x}_i$ a *support vector*.

- Therefore, if $\boldsymbol{x}_i$ is *not* a support vector, then $\alpha_i^* = 0$
- Conclusion: The SVM classifier depends only on the SVs:

$$f(x) = \text{sign}\left\{\sum_{i=1}^{n} \alpha_i^* y_i\, k(x, x_i) + b^*\right\} = \text{sign}\left\{\sum_{\substack{i:\, x_i \\ \text{is a SV}}} \alpha_i^* y_i\, k(x, x_i) + b^*\right\}$$

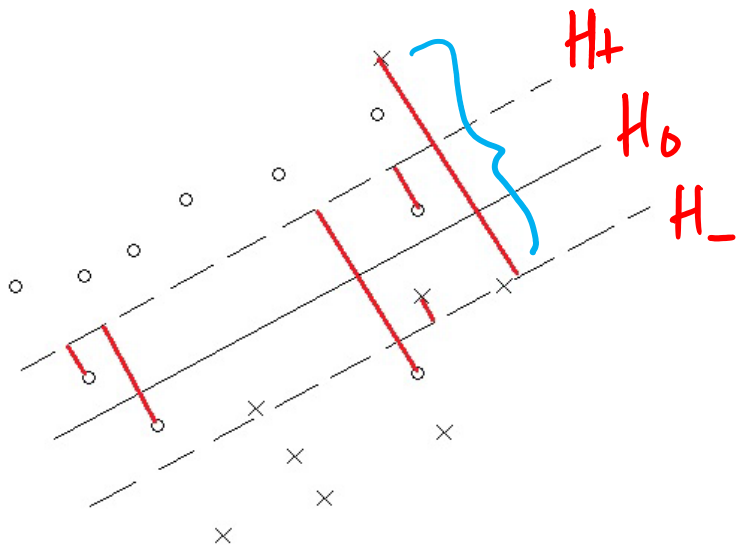
Margin Errors

- Recall the *decision boundary* is the hyperplane

$$H_0 := \{\boldsymbol{x} \mid \langle \boldsymbol{w}^*, \boldsymbol{x} \rangle + b^* = 0\}$$

- The *margin boundaries* are hyperplanes H_+ and H_- defined by

$$H_\pm := \{\boldsymbol{x} \mid \langle \boldsymbol{w}^*, \boldsymbol{x} \rangle + b^* = \pm 1\}$$

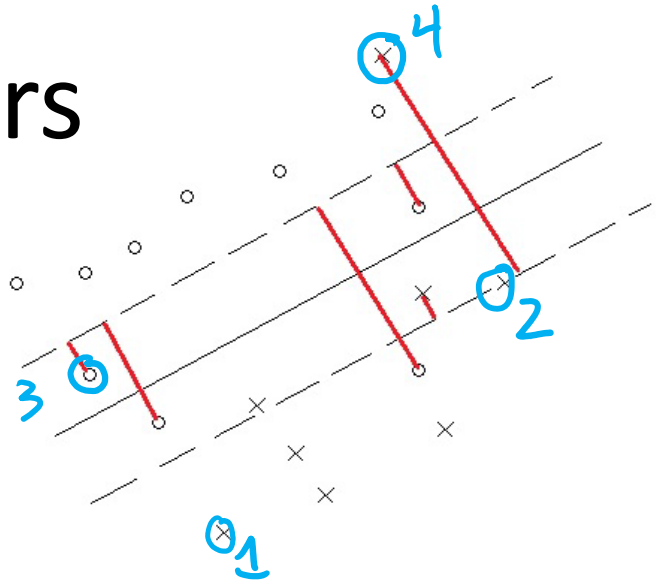


x_i is SV

$\Updownarrow$

$1 - \xi_i^* - y_i(\langle w^*, x_i \rangle + b^*) = 0$

Margin Errors



- Optimal slack variables satisfy

$$\xi_i^* = \max(0, 1 - y_i((\boldsymbol{w}^*)^T \boldsymbol{x} + b^*)).$$

- Consider four cases:

1. If $y_i(\langle \boldsymbol{w}^*, \boldsymbol{x}_i \rangle + b^*) > 1$, then $\xi_i^* = 0$ and $\boldsymbol{x}_i$ is not a support vector.
2. If $y_i(\langle \boldsymbol{w}^*, \boldsymbol{x}_i \rangle + b^*) = 1$, then $\xi_i^* = 0$ and $\boldsymbol{x}_i$ is a support vector.
3. If $0 \leqslant y_i(\langle \boldsymbol{w}^*, \boldsymbol{x}_i \rangle + b^*) < 1$, then $\xi_i^* > 0$ and $\boldsymbol{x}_i$ is a support vector. "within the margin"
4. If $y_i(\langle \boldsymbol{w}^*, \boldsymbol{x}_i \rangle + b^*) < 0$, then $\xi_i^* > 1$ and $\boldsymbol{x}_i$ is a support vector.

- Case 2-4 are referred to as *margin errors* – these are the SV

SVM Margin Errors

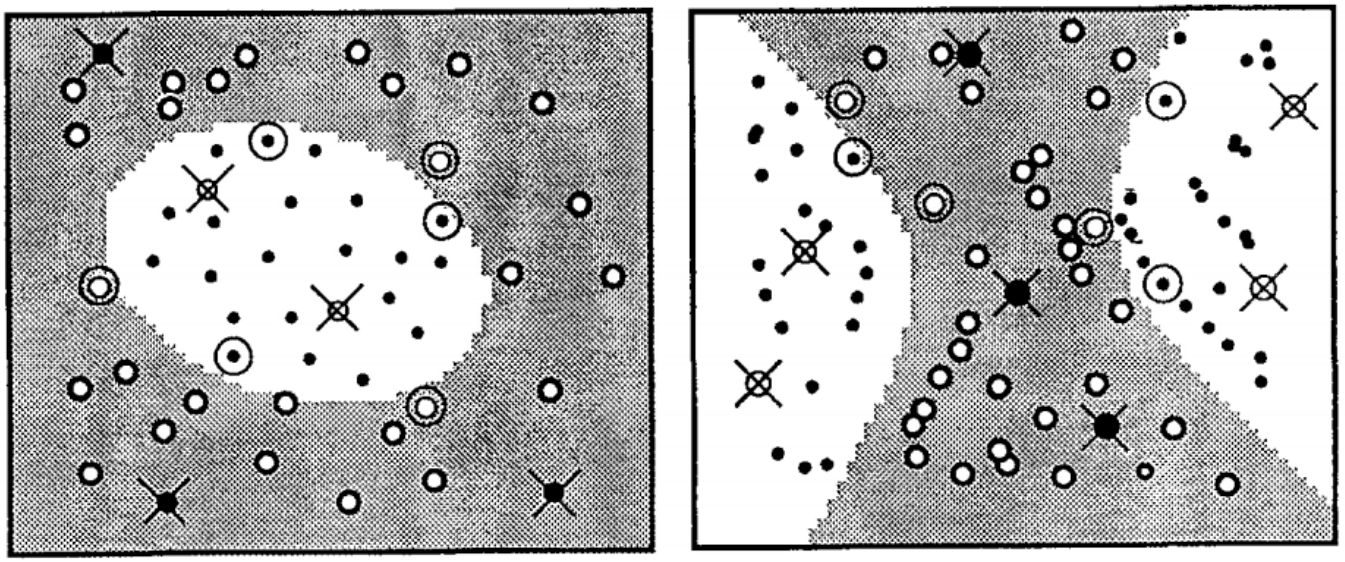


Figure from the original paper by Corinna Cortes and Vladimir Vapnik, "Support vector networks", *Machine Learning Journal*, 1995.

Poll

True or false: If $\boldsymbol{x}_i$ is not a support vector and it is removed from the training data set, the SVM classifier does not change.

(A) True

(B) False

Remarks

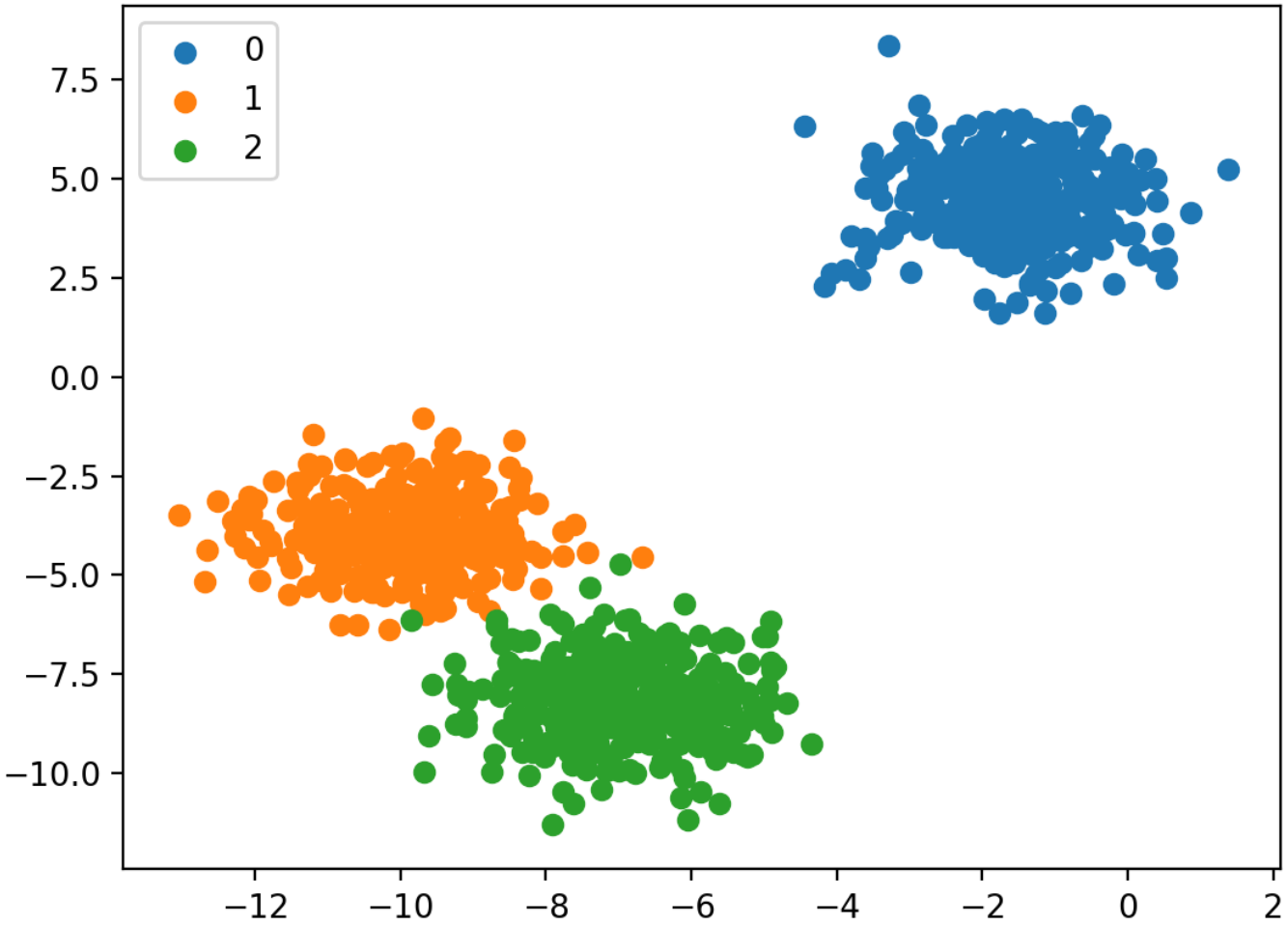
- The final classifier depends only on those training data points that are support vectors. In practice this is often only a small fraction of the original training data, so the classifier can be stored and evaluated efficiently.
- The size of the dual (i.e., the number of variables) is n, and in particular it is independent of the output dimension of the feature map $\mathbf{\Phi}$ associated with k, which could be infinite.
- Computational complexity of solving dual is $\Omega(dn^2)$ and $O(dn^3)$.
- Modern solvers use (block) coordinate descent and can handle data sets with tens of millions of points

Remark

- Suppose $\boldsymbol{\alpha}^*$ is a dual solution, and that $\boldsymbol{x}_i$ is not a support vector for a certain i. Then $\alpha_i^* = 0$.
- Now consider the modified data set obtained by removing $(\boldsymbol{x}_i, y_i)$. Let $\boldsymbol{\alpha}^*_{-i}$ be the length $n-1$ vector obtained by removing the i-th entry from $\boldsymbol{\alpha}^*$.
- Then $\boldsymbol{\alpha}^*_{-i}$ solves the dual for the modified dataset.
- Therefore the SVM remains unchanged.

Multiclass Classification

Linear



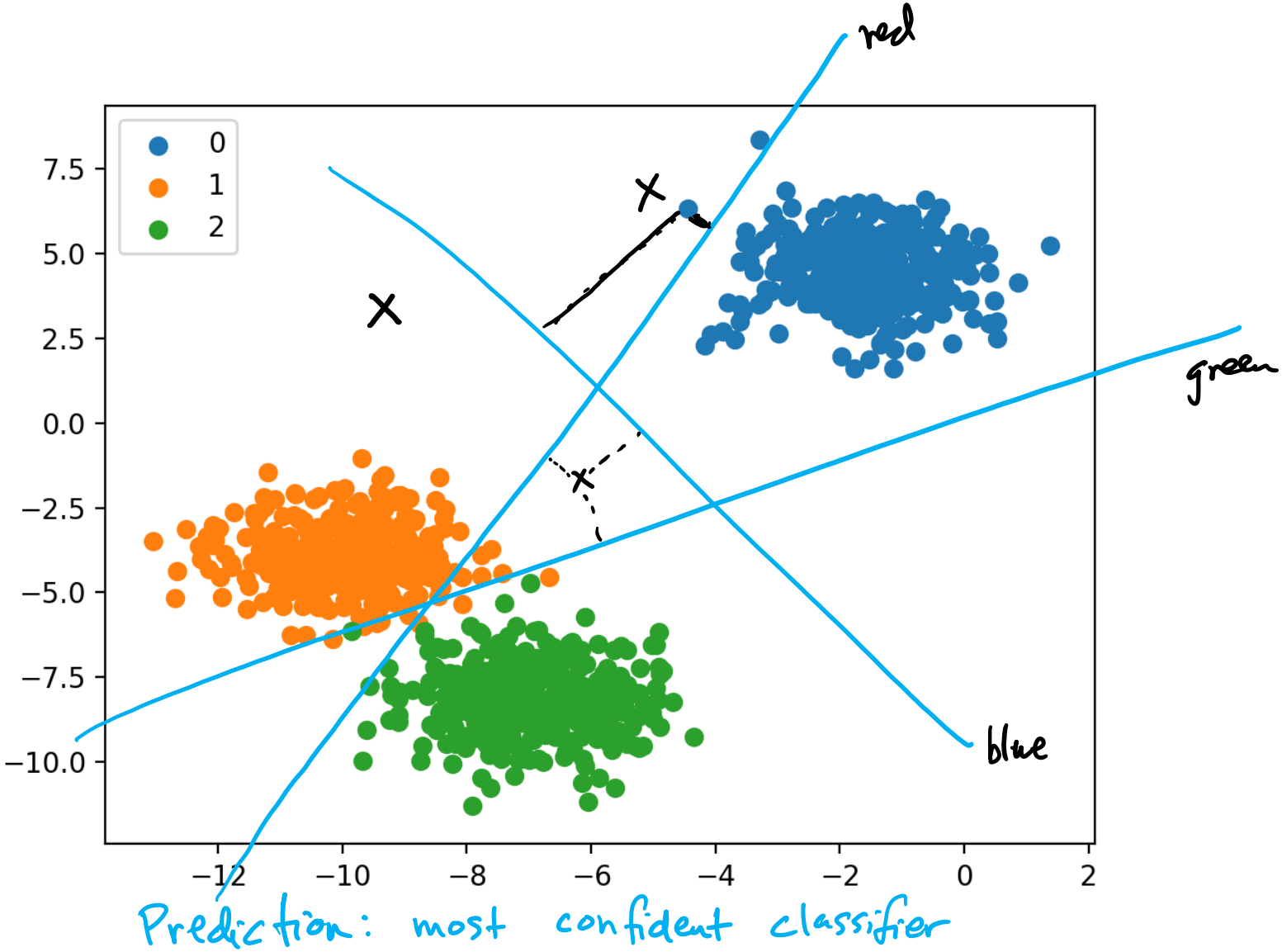
Multiclass Classification

- One vs. the rest
- All pairs

} reduce to a set of binary problems

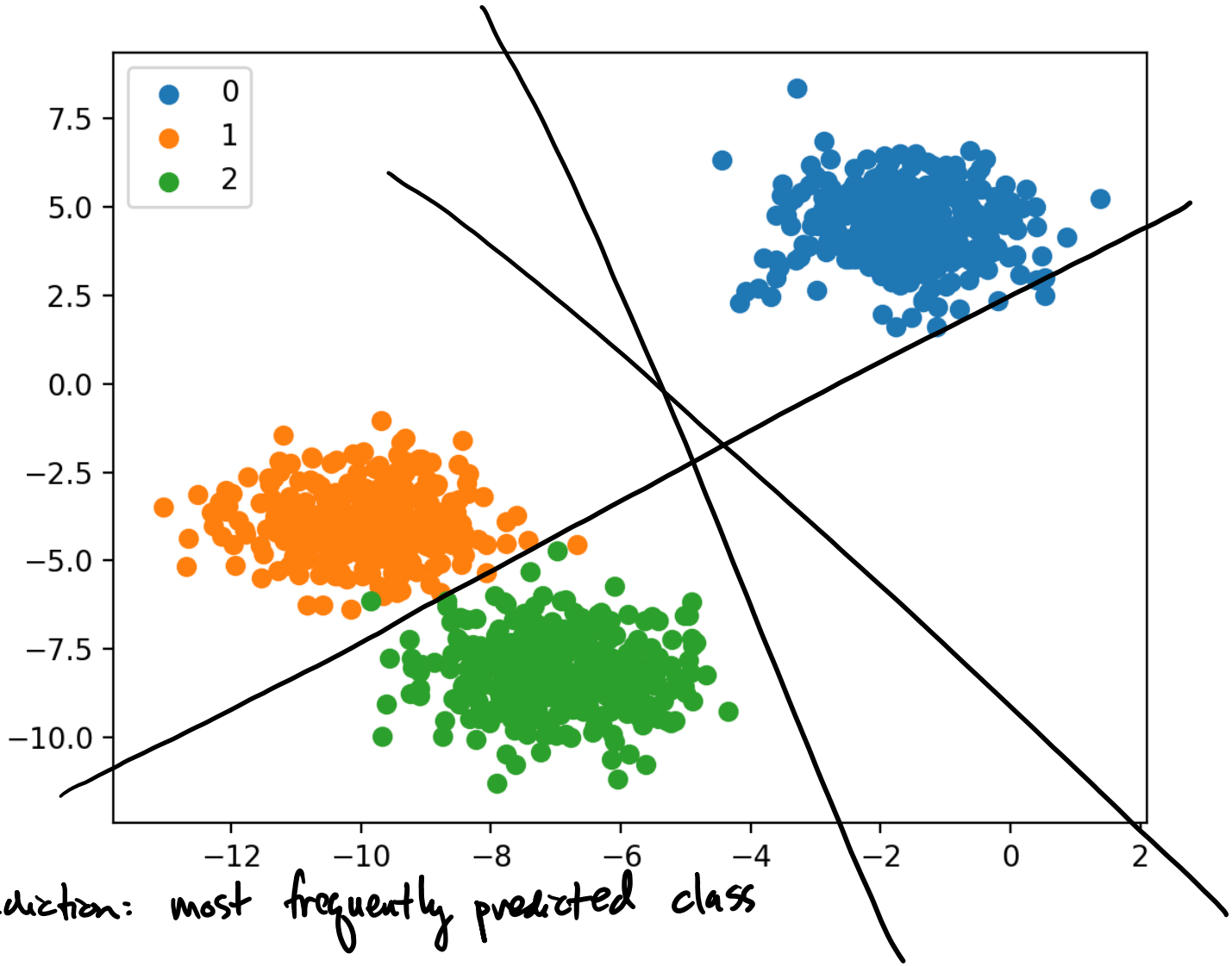
- Multiclass max margin and soft margin linear classifiers
- Multiclass loss functions
  - multiclass hinge losses
  - multiclass logistic loss

One vs. The Rest



$\binom{K}{2}$

All Pairs



Prediction: most frequently predicted class

Multiclass Linear Classification

- K classes
- For each $k \in \{1, \ldots, K\}$, there is an affine function

$$f_k(\boldsymbol{x}) = \boldsymbol{w}_k^T \boldsymbol{x} + b_k$$

score function

- The final classifier is

$$\boldsymbol{x} \mapsto \text{argmax}_k \;\; f_k(\boldsymbol{x})$$

- We have already seen an example: LDA
- How can we extend other linear classification methods to learn these affine functions?

Max-Margin Formulation

- Training data $(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_n, y_n)$ $\quad y_i \in \{1, \ldots, K\}$
- The training data is *linearly separable* iff there exist $\boldsymbol{w}_1, b_1, \ldots, \boldsymbol{w}_K, b_K$ such that

$$\underbrace{w_{y_i}^T x_i + b_{y_i}}_{\text{score function for class } y_i} > \underbrace{w_k^T x_i + b_k}_{\text{score function for class } k} \quad \forall i \quad \forall k \neq y_i$$

- Max-margin separating hyperplane

$$\min_{\{w_k, b_k\}} \frac{1}{2} \sum_{k=1}^{K} \|w_k\|^2$$

$$\text{s.t.} \quad \left(w_{y_i}^T x_i + b_{y_i}\right) - \left(w_k^T x_i + b_k\right) \geq 1 \quad \forall i \quad \forall k \neq y_i$$

- Reduces to what we have studied when $K = 2$

Soft-Margin Formulation

- Crammer-Singer version:

$$\min_{\{w_k, b_k\}, \{\xi_i\}} \quad \frac{1}{2}\sum_k \|w_k\|^2 + \frac{C}{n}\sum_{i=1}^{n} \xi_i$$

$$\text{s.t.} \quad \left(w_{y_i}^T x_i + b_{y_i}\right) - \left(w_k^T x_i + b_k\right) \geq 1 - \xi_i \quad \forall i, k \neq y_i$$

$$\xi_i \geq 0 \quad \forall i$$

- Weston-Watkins version:

$$\min_{\{w_k, b_k\}, \{\xi_{ik}\}} \quad \frac{1}{2}\sum \|w_k\|^2 + \frac{C}{n}\sum_{i=1}^{n}\sum_{k \neq y_i} \xi_{ik}$$

$$\text{s.t.} \quad \left(w_{y_i}^T x_i + b_{y_i}\right) - \left(w_k^T x_i + b_k\right) \geq 1 - \xi_{ik} \quad \forall i, k \neq y_i$$

$$\xi_{ik} \geq 0$$

Multiclass Hinge Losses

- ERM formulation: See my lecture notes for Crammer-Singer and Weston-Watkins
- These multiclass methods can be kernelized

$$\min_{w_1, b_1, \ldots, w_k, b_k} \frac{1}{n} \sum L(y_i, f(x_i)) + \Omega(f)$$

$$f(x) = (w_1^T x + b_1, \ldots, w_k^T x + b_k)$$

Multiclass Logistic Regression

- Recall

$$\eta_k(\boldsymbol{x}) = \Pr(Y = k \mid \boldsymbol{X} = \boldsymbol{x})$$

$\forall x \quad \sum_{k=1}^{K} \eta_k(x) = 1$

- Define the *softmax* function, which maps

$$\mathbb{R}^K \to \{(p_1, \ldots, p_K) \mid p_k \geq 0, \sum_k p_k = 1\}$$

probability simplex

by $\psi(\boldsymbol{v}) = (\psi_1(\boldsymbol{v}), \ldots, \psi_K(\boldsymbol{v}))$ where

$$\psi_k(\boldsymbol{v}) := \frac{\exp(v_k)}{\sum_{\ell=1}^{K} \exp(v_\ell)}$$

- The *multinomial logistic regression* model assumes the existence of $\boldsymbol{w}_1, b_1, \ldots, \boldsymbol{w}_K, b_K$ such that

$$\eta_k(\boldsymbol{x}) = \psi_k(w_1^T x + b_1, \ldots, w_K^T x + b_K)$$

~~where $f_k(\boldsymbol{x}) = \boldsymbol{w}_k^T \boldsymbol{x} + b_k$.~~

MLE/ERM

- Let $\boldsymbol{\theta}$ collect $\boldsymbol{w}_1, b_1, \ldots, \boldsymbol{w}_K, b_K$.
- Denote

$$f_{\boldsymbol{\theta}}(\boldsymbol{x}) = (f_1(x), \ldots, f_K(x))$$

$$= (w_1^T x + b_1, \ldots, w_K^T x + b_K)$$

- The maximum likelihood estimation problem can be written

$$\min_{\boldsymbol{\theta}} \frac{1}{n} \sum_{i=1}^{n} L(y_i, f_{\boldsymbol{\theta}}(\boldsymbol{x}_i))$$

where

$$L(y, \boldsymbol{v}) = -\ln \psi_y(v)$$

is the *multinomial logistic loss*.

- This loss is convex as a function of $\boldsymbol{v}$
- Agrees with binary version when $K = 2$

Connection to Cross-Entropy

- If $\boldsymbol{p} = (p_1, \ldots, p_K)$ and $\boldsymbol{q} = (q_1, \ldots, q_K)$ are two probability vectors, the *cross-entropy* between them is

$$CE(\boldsymbol{q}, \boldsymbol{p}) := -\sum_k q_k \ln(p_k)$$

$e_k = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$ ← kth position

- Let $\boldsymbol{e}_k$ denote the kth standard basis vector in $\mathbb{R}^K$
- Then

$$L(y, v) = CE(e_y, \psi(v))$$

"cross entrop loss"

Hence the cross-entropy loss and multinomial logistic ~~regression~~ loss are the same.

- Preview: If $\boldsymbol{\theta}$ contains the weights of a neural network $f_{\boldsymbol{\theta}}(\boldsymbol{x})$ with K nodes in the output layer, the network is trained by solving

$$\min_\theta \sum_{i=1}^n CE(e_{y_i}, \psi(f_\theta(x))) = \sum_{i=1}^n L(y_i, f_\theta(x_i))$$