

# SVM (Contd), Multiclass and One-Class SVM

Piyush Rai

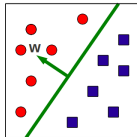
Introduction to Machine Learning (CS771A)

September 4, 2018



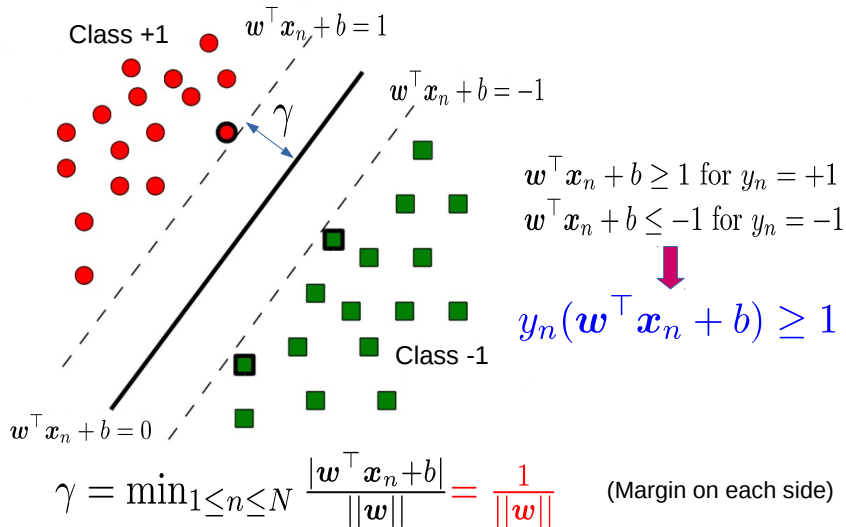
# Recap: Hyperplane-based Classification

- Basic idea: Learn to separate by a hyperplane  $\mathbf{w}^\top \mathbf{x} + b = 0$



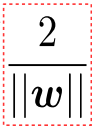
- Predict the label of a test input  $\mathbf{x}_*$  as:  $\hat{y}_* = \text{sign}(\mathbf{w}^\top \mathbf{x}_* + b)$
- The hyperplane may be “implied” by the model, or learned directly
  - Implied: Prototype-based classification, nearest neighbors, generative classification, etc.
  - Directly learned: Logistic regression, Perceptron, Support Vector Machine, etc.
- The “direct” approach defines a model with parameters  $\mathbf{w}$  (and optionally  $b$ ) and learns them by minimizing a suitable loss function (and doesn't model  $\mathbf{x}$ , i.e., purely discriminative)
- The hyperplane need not be linear (e.g., can be made nonlinear using kernel methods - next class)

# Recap: Hyperplanes and Margin




## Recap: Maximum-Margin Hyperplane

$$(\hat{\mathbf{w}}, \hat{b}) = \arg \max_{\mathbf{w}, b} \frac{2}{\|\mathbf{w}\|}, \quad \text{s.t.} \quad y_n(\mathbf{w}^\top \mathbf{x}_n + b) \geq 1$$

Total margin  $(2\gamma)$  

(equivalent to)


$$(\hat{\mathbf{w}}, \hat{b}) = \arg \min_{\mathbf{w}, b} \frac{\|\mathbf{w}\|^2}{2}, \quad \text{s.t.} \quad y_n(\mathbf{w}^\top \mathbf{x}_n + b) \geq 1$$

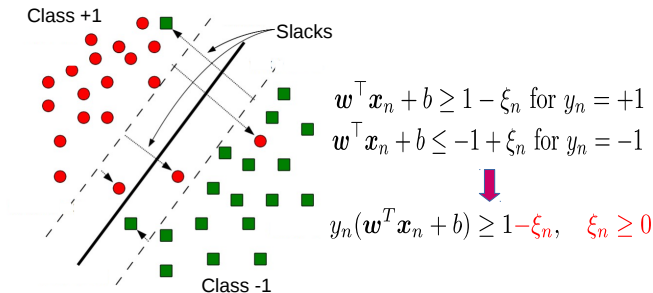
**Hard-margin SVM**

(“hard” = want all points to satisfy the margin constraint)



# Recap: Maximum-Margin Hyperplane with Slacks

- Still want a max-margin hyperplane but want to **relax the hard constraint**  $y_n(\mathbf{w}^\top \mathbf{x}_n + b) \geq 1$
- Let's allow every point  $\mathbf{x}_n$  to “slack the constraint” by a distance  $\xi_n \geq 0$



- Points with  $\xi_n \geq 0$  will be either in the **margin region** or totally on the **wrong side**
- **New Objective:** Maximize the margin while keeping the **sum of slacks**  $\sum_{n=1}^N \xi_n$  small
- **Note:** Can also think of the **sum of slacks** as the **total training error**



# Recap: Maximum-Margin Hyperplane with Slacks

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

Annotations:

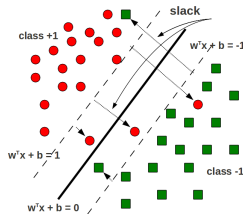
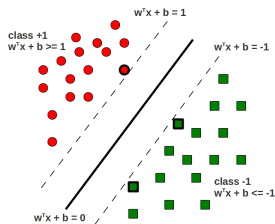
- Maximize the margin (points to  $\frac{\|\mathbf{w}\|^2}{2}$ )
- Hyperparameter to balance the two (points to  $C$ )
- Minimize the sum of slacks (don't have too many violations) (points to  $\sum_{n=1}^N \xi_n$ )

$$\text{s.t. } y_n(\mathbf{w}^\top \mathbf{x}_n + b) \geq 1 - \xi_n \quad \text{Slack-relaxed constraints}$$
$$\xi_n \geq 0$$

- This formulation is known as the “soft-margin” SVM
- Very small  $C$ : Large margin but also large training error. :-()
- Very large  $C$ : Small training error but also small margin. :-()
- $C$  controls the trade-off between large margin and small training error



# Summary: Hard-Margin SVM vs Soft-Margin SVM



- Objective for the hard-margin SVM (unknowns are  $\mathbf{w}$  and  $b$ )

$$\begin{aligned} \arg \min_{\mathbf{w}, b} \quad & \frac{\|\mathbf{w}\|^2}{2} \\ \text{subject to} \quad & y_n(\mathbf{w}^T \mathbf{x}_n + b) \geq 1, \quad n = 1, \dots, N \end{aligned}$$

- Objective for the soft-margin SVM (unknowns are  $\mathbf{w}$ ,  $b$ , and  $\{\xi_n\}_{n=1}^N$ )

$$\begin{aligned} \arg \min_{\mathbf{w}, b, \xi} \quad & \frac{\|\mathbf{w}\|^2}{2} + C \sum_{n=1}^N \xi_n \\ \text{subject to} \quad & y_n(\mathbf{w}^T \mathbf{x}_n + b) \geq 1 - \xi_n, \quad \xi_n \geq 0 \quad n = 1, \dots, N \end{aligned}$$

- In either case, we have to solve a constrained, convex optimization problem



# Solving SVM Objectives





# Solving Hard-Margin SVM

- The hard-margin SVM optimization problem is:

$$\begin{array}{ll} \arg \min_{\mathbf{w}, b} & \frac{\|\mathbf{w}\|^2}{2} \\ \text{subject to} & 1 - y_n(\mathbf{w}^T \mathbf{x}_n + b) \leq 0, \quad n = 1, \dots, N \end{array}$$

- A constrained optimization problem. Can solve using Lagrange's method
- Introduce **Lagrange Multipliers**  $\alpha_n$  ( $n = \{1, \dots, N\}$ ), one for each constraint, and solve

$$\min_{\mathbf{w}, b} \max_{\alpha \geq 0} \mathcal{L}(\mathbf{w}, b, \alpha) = \frac{\|\mathbf{w}\|^2}{2} + \sum_{n=1}^N \alpha_n \{1 - y_n(\mathbf{w}^T \mathbf{x}_n + b)\}$$

- Note:  $\alpha = [\alpha_1, \dots, \alpha_N]$  is the vector of Lagrange multipliers
- Note: It is easier (and helpful; we will soon see why) to solve the **dual problem**: min and then max



# Solving Hard-Margin SVM

- The dual problem (min then max) is

$$\max_{\alpha \geq 0} \min_{\mathbf{w}, b} \mathcal{L}(\mathbf{w}, b, \alpha) = \frac{\mathbf{w}^\top \mathbf{w}}{2} + \sum_{n=1}^N \alpha_n \{1 - y_n(\mathbf{w}^\top \mathbf{x}_n + b)\}$$

- Take (partial) derivatives of  $\mathcal{L}$  w.r.t.  $\mathbf{w}$ ,  $b$  and set them to zero

$$\frac{\partial \mathcal{L}}{\partial \mathbf{w}} = 0 \Rightarrow \mathbf{w} = \sum_{n=1}^N \alpha_n y_n \mathbf{x}_n \quad \frac{\partial \mathcal{L}}{\partial b} = 0 \Rightarrow \sum_{n=1}^N \alpha_n y_n = 0$$

- Important: Note the form of the solution  $\mathbf{w}$  - it is simply a **weighted sum of all the training inputs**  $\mathbf{x}_1, \dots, \mathbf{x}_N$  (and  $\alpha_n$  is like the “importance” of  $\mathbf{x}_n$ )
- Substituting  $\mathbf{w} = \sum_{n=1}^N \alpha_n y_n \mathbf{x}_n$  in Lagrangian, we get the dual problem as (verify)

$$\max_{\alpha \geq 0} \mathcal{L}_D(\alpha) = \sum_{n=1}^N \alpha_n - \frac{1}{2} \sum_{m,n=1}^N \alpha_m \alpha_n y_m y_n (\mathbf{x}_m^\top \mathbf{x}_n)$$



# Solving Hard-Margin SVM

- Can write the objective more compactly in vector/matrix form as

$$\max_{\alpha \geq 0} \mathcal{L}_D(\alpha) = \alpha^\top \mathbf{1} - \frac{1}{2} \alpha^\top \mathbf{G} \alpha$$

where  $\mathbf{G}$  is an  $N \times N$  matrix with  $G_{mn} = y_m y_n \mathbf{x}_m^\top \mathbf{x}_n$ , and  $\mathbf{1}$  is a vector of 1s

- **Good news:** This is **maximizing a concave function** (or minimizing a convex function - verify that the Hessian is  $\mathbf{G}$ , which is p.s.d.). Note that our original SVM objective was also convex
- **Important:** Inputs  $\mathbf{x}$ 's only appear as **inner products** (helps to “kernelize”; more on this later)
- Can solve<sup>†</sup> the above objective function for  $\alpha$  using various methods, e.g.,
  - Treating the objective as a **Quadratic Program** (QP) and running some off-the-shelf QP solver such as quadprog (MATLAB), CVXOPT, CPLEX, etc.
  - Using **(projected) gradient methods** (projection needed because the  $\alpha$ 's are constrained). Gradient methods will usually be much faster than QP methods.
  - Using **co-ordinate ascent** methods (optimize for one  $\alpha_n$  at a time); often very fast

<sup>†</sup> If interested in more details of the solver, see: “Support Vector Machine Solvers” by Bottou and Lin



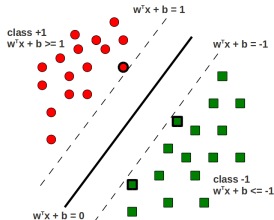
# Hard-Margin SVM: The Solution

- Once we have the  $\alpha_n$ 's,  $\mathbf{w}$  and  $b$  can be computed as:

$$\mathbf{w} = \sum_{n=1}^N \alpha_n y_n \mathbf{x}_n \quad (\text{we already saw this})$$

$$b = -\frac{1}{2} \left( \min_{n:y_n=+1} \mathbf{w}^T \mathbf{x}_n + \max_{n:y_n=-1} \mathbf{w}^T \mathbf{x}_n \right) \quad (\text{exercise})$$

- A nice property:** Most  $\alpha_n$ 's in the solution will be zero (**sparse solution**)



- Reason: **Karush-Kuhn-Tucker (KKT) conditions**
- For the optimal  $\alpha_n$ 's

$$\alpha_n \{1 - y_n(\mathbf{w}^T \mathbf{x}_n + b)\} = 0$$

- $\alpha_n$  is **non-zero** only if  $\mathbf{x}_n$  lies on one of the two **margin boundaries**, i.e., for which  $y_n(\mathbf{w}^T \mathbf{x}_n + b) = 1$
- These examples are called **support vectors**
- Recall the support vectors “support” the margin boundaries



# Solving Soft-Margin SVM



# Solving Soft-Margin SVM

- Recall the soft-margin SVM optimization problem:

$$\begin{aligned} \min_{\mathbf{w}, b, \boldsymbol{\xi}} \quad & f(\mathbf{w}, b, \boldsymbol{\xi}) = \frac{\|\mathbf{w}\|^2}{2} + C \sum_{n=1}^N \xi_n \\ \text{subject to} \quad & 1 \leq y_n(\mathbf{w}^T \mathbf{x}_n + b) + \xi_n, \quad -\xi_n \leq 0 \quad n = 1, \dots, N \end{aligned}$$

- Note:  $\boldsymbol{\xi} = [\xi_1, \dots, \xi_N]$  is the vector of slack variables
- Introduce **Lagrange Multipliers**  $\alpha_n, \beta_n$  ( $n = \{1, \dots, N\}$ ), for constraints, and solve the Lagrangian:

$$\min_{\mathbf{w}, b, \boldsymbol{\xi}} \max_{\boldsymbol{\alpha} \geq 0, \boldsymbol{\beta} \geq 0} \mathcal{L}(\mathbf{w}, b, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\beta}) = \frac{\|\mathbf{w}\|^2}{2} + C \sum_{n=1}^N \xi_n + \sum_{n=1}^N \alpha_n \{1 - y_n(\mathbf{w}^T \mathbf{x}_n + b) - \xi_n\} - \sum_{n=1}^N \beta_n \xi_n$$

- Note: The terms in red above were not present in the hard-margin SVM
- Two sets of dual variables  $\boldsymbol{\alpha} = [\alpha_1, \dots, \alpha_N]$  and  $\boldsymbol{\beta} = [\beta_1, \dots, \beta_N]$ . We'll eliminate the primal variables  $\mathbf{w}, b, \boldsymbol{\xi}$  to get dual problem containing the dual variables (just like in the hard margin case)

# Solving Soft-Margin SVM

- The Lagrangian problem to solve

$$\min_{\mathbf{w}, b, \xi} \max_{\alpha \geq 0, \beta \geq 0} \mathcal{L}(\mathbf{w}, b, \xi, \alpha, \beta) = \frac{\mathbf{w}^T \mathbf{w}}{2} + C \sum_{n=1}^N \xi_n + \sum_{n=1}^N \alpha_n \{1 - y_n(\mathbf{w}^T \mathbf{x}_n + b) - \xi_n\} - \sum_{n=1}^N \beta_n \xi_n$$

- Take (partial) derivatives of  $\mathcal{L}$  w.r.t.  $\mathbf{w}$ ,  $b$ ,  $\xi_n$  and set them to zero

$$\frac{\partial \mathcal{L}}{\partial \mathbf{w}} = 0 \Rightarrow \mathbf{w} = \sum_{n=1}^N \alpha_n y_n \mathbf{x}_n, \quad \frac{\partial \mathcal{L}}{\partial b} = 0 \Rightarrow \sum_{n=1}^N \alpha_n y_n = 0, \quad \frac{\partial \mathcal{L}}{\partial \xi_n} = 0 \Rightarrow C - \alpha_n - \beta_n = 0$$

- Note: Solution of  $\mathbf{w}$  again has the same form as in the hard-margin case (weighted sum of all inputs with  $\alpha_n$  being the importance of input  $\mathbf{x}_n$ )
- Note: Using  $C - \alpha_n - \beta_n = 0$  and  $\beta_n \geq 0 \Rightarrow \alpha_n \leq C$  (recall that, for the hard-margin case,  $\alpha \geq 0$ )
- Substituting these in the Lagrangian  $\mathcal{L}$  gives the **Dual** problem

$$\max_{\alpha \leq C, \beta \geq 0} \mathcal{L}_D(\alpha, \beta) = \sum_{n=1}^N \alpha_n - \frac{1}{2} \sum_{m,n=1}^N \alpha_m \alpha_n y_m y_n (\mathbf{x}_m^T \mathbf{x}_n)$$



# Solving Soft-Margin SVM

- Interestingly, the dual variables  $\beta$  don't appear in the objective!
- Just like the hard-margin case, we can write the dual more compactly as

$$\max_{\alpha \leq C} \mathcal{L}_D(\alpha) = \alpha^\top \mathbf{1} - \frac{1}{2} \alpha^\top \mathbf{G} \alpha$$

where  $\mathbf{G}$  is an  $N \times N$  matrix with  $G_{mn} = y_m y_n \mathbf{x}_m^\top \mathbf{x}_n$ , and  $\mathbf{1}$  is a vector of 1s

- Like hard-margin case, solving the dual requires concave maximization (or convex minimization)
- Can be solved<sup>†</sup> the same way as hard-margin SVM (except that  $\alpha \leq C$ )
  - Can solve for  $\alpha$  using QP solvers or (projected) gradient methods
- Given  $\alpha$ , the solution for  $\mathbf{w}, b$  has the same form as hard-margin case
- **Note:**  $\alpha$  is again **sparse**. Nonzero  $\alpha_n$ 's correspond to the **support vectors**

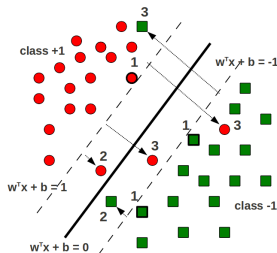
<sup>†</sup> If interested in more details of the solver, see: "Support Vector Machine Solvers" by Bottou and Lin





# Support Vectors in Soft-Margin SVM

- The hard-margin SVM solution had only one type of support vectors
  - .. ones that lie on the margin boundaries  $\mathbf{w}^T \mathbf{x} + b = -1$  and  $\mathbf{w}^T \mathbf{x} + b = +1$
- The soft-margin SVM solution has **three types of support vectors**



- 1 Lying on the margin boundaries  $\mathbf{w}^T \mathbf{x} + b = -1$  and  $\mathbf{w}^T \mathbf{x} + b = +1$  ( $\xi_n = 0$ )
- 2 Lying within the margin region ( $0 < \xi_n < 1$ ) but still on the correct side
- 3 Lying on the wrong side of the hyperplane ( $\xi_n \geq 1$ )



# SVMs via Dual Formulation: Some Comments

- Recall the final dual objectives for hard-margin and soft-margin SVM

$$\text{Hard-Margin SVM: } \max_{\alpha \geq 0} \mathcal{L}_D(\alpha) = \alpha^\top \mathbf{1} - \frac{1}{2} \alpha^\top \mathbf{G} \alpha$$

$$\text{Soft-Margin SVM: } \max_{\alpha \leq C} \mathcal{L}_D(\alpha) = \alpha^\top \mathbf{1} - \frac{1}{2} \alpha^\top \mathbf{G} \alpha$$

- The dual formulation is nice due to two primary reasons:
  - Allows conveniently handling the margin based constraint (via Lagrangians)
  - Important:** Allows learning nonlinear separators by replacing inner products (e.g.,  $G_{mn} = y_m y_n \mathbf{x}_m^\top \mathbf{x}_n$ ) by kernelized similarities (kernelized SVMs)
- However, the dual formulation can be expensive if  $N$  is large. Have to solve for  $N$  variables  $\alpha = [\alpha_1, \dots, \alpha_N]$ , and also need to store an  $N \times N$  matrix  $\mathbf{G}$
- A lot of work<sup>†</sup> on speeding up SVM in these settings (e.g., can use co-ord. descent for  $\alpha$ )

<sup>†</sup> See: "Support Vector Machine Solvers" by Bottou and Lin



# SVM: The Regularized Loss Function View

- Maximize the margin subject to constraints led to the soft-margin formulation of SVM

$$\begin{aligned} \arg \min_{\mathbf{w}, b, \xi} \quad & \frac{\|\mathbf{w}\|^2}{2} + C \sum_{n=1}^N \xi_n \\ \text{subject to} \quad & y_n(\mathbf{w}^\top \mathbf{x}_n + b) \geq 1 - \xi_n, \quad \xi_n \geq 0 \quad n = 1, \dots, N \end{aligned}$$

- Note that the slack  $\xi_n$  is the same as  $\max\{0, 1 - y_n(\mathbf{w}^\top \mathbf{x}_n + b)\}$ , i.e., **hinge loss** for  $(\mathbf{x}_n, y_n)$
- Another View:** Thus the above is equivalent to minimizing the  **$\ell_2$  regularized hinge loss**

$$\mathcal{L}(\mathbf{w}, b) = \sum_{n=1}^N \max\{0, 1 - y_n(\mathbf{w}^\top \mathbf{x}_n + b)\} + \frac{\lambda}{2} \mathbf{w}^\top \mathbf{w}$$

- Comparing the two:** **Sum of slacks** is like **sum of hinge losses**,  $C$  and  $\lambda$  play similar roles
- Can learn  $(\mathbf{w}, b)$  directly by minimizing  $\mathcal{L}(\mathbf{w}, b)$  using (stochastic)(sub)gradient descent
  - Hinge-loss version preferred for **linear SVMs**, or with other regularizers on  $\mathbf{w}$  (e.g.,  $\ell_1$ )



# Multiclass SVM

- Multiclass SVMs use  $K$  weight vectors  $\mathbf{W} = [\mathbf{w}_1, \mathbf{w}_2, \dots, \mathbf{w}_K]$  (similar to softmax regression)

$$\hat{y}_* = \arg \max_k \mathbf{w}_k^\top \mathbf{x}_n \quad (\text{prediction rule})$$

- Just like binary case, we can formulate a maximum-margin problem (without or with slacks)

$$\begin{aligned} \hat{\mathbf{W}} &= \arg \min_{\mathbf{W}} \sum_{k=1}^K \frac{\|\mathbf{w}_k\|^2}{2} & \hat{\mathbf{W}} &= \arg \min_{\mathbf{W}} \sum_{k=1}^K \frac{\|\mathbf{w}_k\|^2}{2} + C \sum_{n=1}^N \xi_n \\ \text{s.t. } \mathbf{w}_{y_n}^\top \mathbf{x}_n &\geq \mathbf{w}_k^\top \mathbf{x}_n + 1 \quad \forall k \neq y_n & \text{s.t. } \mathbf{w}_{y_n}^\top \mathbf{x}_n &\geq \mathbf{w}_k^\top \mathbf{x}_n + 1 - \xi_n \quad \forall k \neq y_n \end{aligned}$$

- Want **score w.r.t. correct class** to be at least 1 more than **score w.r.t. all other classes**
- The version with slack corresponds to minimizing a **multi-class hinge loss**

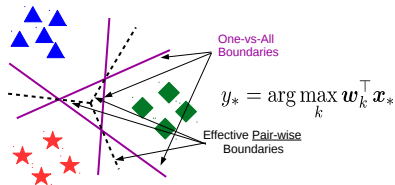
$$\mathcal{L}(\mathbf{W}) = \max\{0, 1 + \max_{k \neq y_n} \mathbf{w}_k^\top \mathbf{x}_n - \mathbf{w}_{y_n}^\top \mathbf{x}_n\} \quad (\text{Crammer-Singer multiclass SVM})$$

- Loss = 0 if **score on correct class** is at least 1 more than **score on next best scoring class**
- Can optimize these similar to how we did it for binary SVM



# Multiclass SVM using Binary SVM?

- Can use binary classifiers to solve multiclass problems
- Note: These approaches can be used with other binary classifiers too (e.g., logistic regression)
- One-vs-All (also called One-vs-Rest): Construct  $K$  binary classification problems



- All-Pairs: Learn  $K$ -choose-2 binary classifiers, one for each pair of classes  $(j, k)$

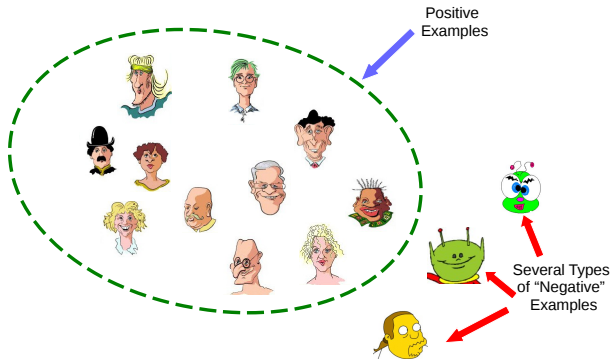
$$y_* = \arg \max_k \sum_{j \neq k} w_{j,k}^\top x_* \quad (\text{predict } k \text{ that wins over all others the most})$$

- All-Pairs approach can be expensive at training and test time (but ways to speed up)



# One-Class Classification

- Can we learn from examples of just one class, say positive examples?
- May be desirable if there are many types of negative examples

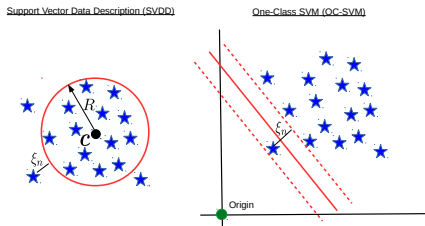


- “Outlier/Novelty Detection” problems can also be formulated like this

Figure credit: Refael Chickvashvili

# One-Class Classification via SVM-like methods

- There are two popular SVM-type approaches to solve one-class problems



- Approach 1: Assume positives lie within a ball with **smallest possible radius** (and allow slacks)
  - Known as **"Support Vector Data Description"** (SVDD). Proposed by [Tax and Duin, 2004]
- Approach 2: Find a **max-marg hyperplane** separating positives from **origin** (representing negatives)
  - Known as **"One-Class SVM"** (OC-SVM). Proposed by [Schölkopf et al., 2001]
- Optimization problems for both cases can be solved similarly as in binary SVM (e.g., via Lagrangian)

# One-Class Classification via SVM-like methods

- There are two popular SVM-type approaches to solve one-class problems

Support Vector Data Description (SVDD)

$$\begin{aligned} \arg \min_{R, c, \xi} R^2 + \frac{1}{\nu N} \sum_{n=1}^N \xi_n \\ \text{s.t. } \|\mathbf{x}_n - \mathbf{c}\|^2 \leq R^2 + \xi_n \quad \forall n \\ \xi_n \geq 0 \end{aligned}$$

Prediction Rule:  $y_* = +1$  if  $\|\mathbf{x}_* - \mathbf{c}\|^2 - R^2 > 0$

One-Class SVM (OC-SVM)

$$\begin{aligned} \arg \min_{\mathbf{w}, \rho, \xi} \|\mathbf{w}\|^2 + \frac{1}{\nu N} \sum_{n=1}^N \xi_n - \rho \\ \text{s.t. } \mathbf{w}^\top \mathbf{x}_n \geq \rho - \xi_n \quad \forall n \\ \xi_n \geq 0 \end{aligned}$$

Prediction Rule:  $y_* = +1$  if  $\mathbf{w}^\top \mathbf{x}_* > \rho$

- Approach 1: Assume positives lie within a ball with **smallest possible radius** (and allow slacks)
  - Known as **“Support Vector Data Description” (SVDD)**. Proposed by [Tax and Duin, 2004]
- Approach 2: Find a **max-marg hyperplane** separating positives from **origin** (representing negatives)
  - Known as **“One-Class SVM” (OC-SVM)**. Proposed by [Schölkopf et al., 2001]
- Optimization problems for both cases can be solved similarly as in binary SVM (e.g., via Lagrangian)



# Nonlinear SVM ?

- A nice property of SVM (and many other models) is that inputs only appear as **inner products**
- For example, recall the dual problem for soft-margin SVM had the form

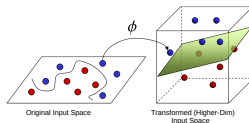
$$\arg \max_{\alpha \leq C} \mathcal{L}_D(\alpha) = \alpha^\top \mathbf{1} - \frac{1}{2} \alpha^\top \mathbf{G} \alpha$$

where  $\mathbf{G}$  is an  $N \times N$  matrix with  $G_{mn} = y_m y_n \mathbf{x}_m^\top \mathbf{x}_n$ , and  $\mathbf{1}$  is a vector of 1s

- We can replace each inner-product by any general form of inner product, e.g.

$$k(\mathbf{x}_n, \mathbf{x}_m) = \phi(\mathbf{x}_n)^\top \phi(\mathbf{x}_m)$$

.. where  $\phi$  is some transformation (e.g., a higher-dimensional mapping) of the data



- Note: Often the mapping  $\phi$  doesn't need to be explicitly computed (“kernel” magic - next class)!
- Can still learn a linear model in the new space but be nonlinear in the original space (wonderful!)

# SVM: Some Notes

- A hugely (perhaps the most!) popular classification algorithm
- Reasonably mature, highly optimized SVM softwares freely available (perhaps the reason why it is more popular than various other competing algorithms)
  - Some popular ones: libSVM, LIBLINEAR, scikit-learn also provides SVM
- Lots of work on scaling up SVMs<sup>†</sup> (both large  $N$  and large  $D$ )
- Extensions beyond binary classification (e.g., multiclass, one-class, [structured outputs](#))
- Can even be used for regression problems (Support Vector Regression)
  - The  [\$\epsilon\$ -insensitive](#) loss for regression does precisely that!
- Nonlinear extensions possible via kernels (next class)

---

<sup>†</sup> See: “Support Vector Machine Solvers” by Bottou and Lin

