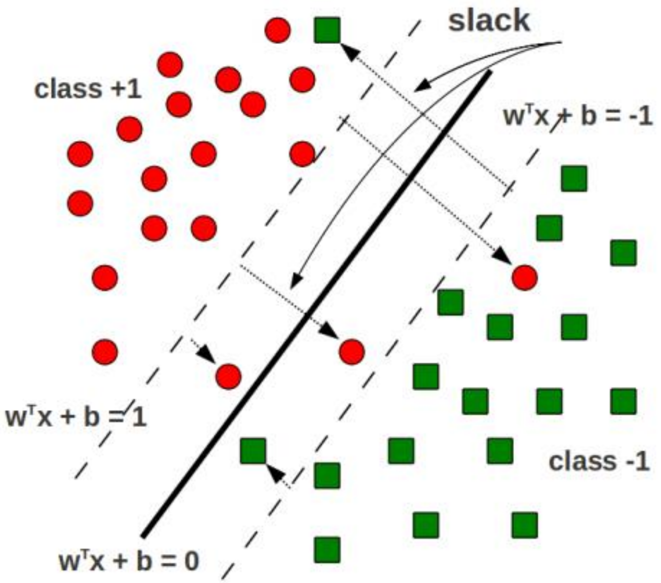
Support Vector Machines (contd)

CS771: Introduction to Machine Learning

Piyush Rai

Soft-Margin SVM

- Goal: Still want to maximize the margin such that
  - Soft-margin constraints $y_n(\boldsymbol{w}^\top \boldsymbol{x}_n + b) \geq 1 - \xi_n$ are satisfied for all training ex.
  - Do not have too many margin violations (sum of slacks $\sum_{n=1}^{N} \xi_n$ should be small)



- The objective func. for soft-margin SVM

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

$$\text{subject to} \quad y_n(\boldsymbol{w}^T \boldsymbol{x}_n + b) \geq 1 - \xi_n, \quad \xi_n \geq 0 \qquad n = 1, \ldots, N$$

- Hyperparameter C controls the trade off between large margin and small training error (need to tune)
  - Too large C: small training error but also small margin (bad)
  - Too small C: large margin but large training error (bad)

Solving Soft-Margin SVM

- Recall the soft-margin SVM optimization problem

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

$$\text{subject to} \quad 1 \leq y_n(\mathbf{w}^T\mathbf{x}_n + b) + \xi_n, \quad -\xi_n \leq 0 \qquad n = 1, \ldots, N$$

- Here $\boldsymbol{\xi} = [\xi_1, \xi_2, \ldots, \xi_N]$ is the vector of slack variables
- Introduce Lagrange multipliers α_n, β_n for each constraint and solve Lagrangian

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

- The terms in red color above were not present in the hard-margin SVM
- Two set of dual variables $\boldsymbol{\alpha} = [\alpha_1, \alpha_2, \ldots, \alpha_N]$ and $\boldsymbol{\beta} = [\beta_1, \beta_2, \ldots, \beta_N]$
- Will eliminate the primal var $\mathbf{w}$, b, $\boldsymbol{\xi}$ to get dual problem containing the dual variables

Solving Soft-Margin SVM

Note: if we ignore the bias term b then we don't need to handle the constraint $\sum_{n=1}^{N} \alpha_n y_n = 0$ (problem becomes a bit more easy to solve)

Otherwise, the α_n's are coupled and some opt. techniques such as co-ordinate aspect can't easily applied

- The Lagrangian problem to solve

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

- Take (partial) derivatives of $\mathcal{L}$ w.r.t. $\boldsymbol{w}, b$, and ξ_n and setting to zero gives

Weighted sum of training inputs

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

- Using $C - \alpha_n - \beta_n = 0$ and $\beta_n \geq 0$, we have $\alpha_n \leq C$ (for hard-margin, $\alpha_n \geq 0$)
- Substituting these in the Lagrangian $\mathcal{L}$ gives the Dual problem

The dual variables $\boldsymbol{\beta}$ don't appear in the dual problem!

Given $\boldsymbol{\alpha}$, $\boldsymbol{w}$ and b can be found just like the hard-margin SVM case

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

Maximizing a concave function (or minimizing a convex function) s.t. $\boldsymbol{\alpha} \leq \boldsymbol{C}$ and $\sum_{n=1}^{N} \alpha_n y_n = 0$. Many methods to solve it.

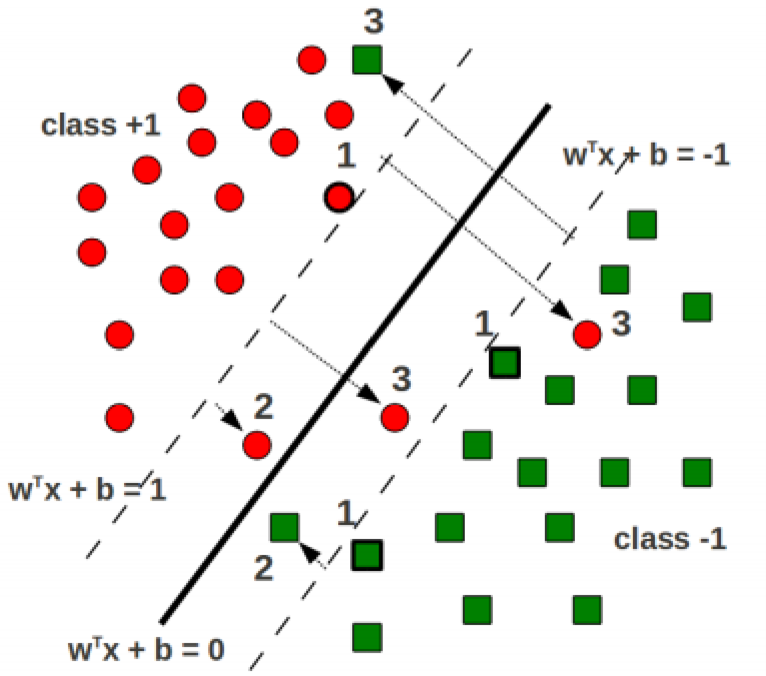
$$\max_{\boldsymbol{\alpha}\leq C} \mathcal{L}_D(\boldsymbol{\alpha}) = \boldsymbol{\alpha}^\top \mathbf{1} - \frac{1}{2}\boldsymbol{\alpha}^\top \mathbf{G}\boldsymbol{\alpha}$$

In the solution, $\boldsymbol{\alpha}$ will still be sparse just like the hard-margin SVM case. Nonzero α_n correspond to the support vectors

(Note: For various SVM solvers, can 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
  - All lied on the supporting hyperplanes $\boldsymbol{w}^\top \boldsymbol{x}_n + b = 1$ and $\boldsymbol{w}^\top \boldsymbol{x}_n + b = -1$
- The soft-margin SVM solution has <u>three</u> types of support vectors (with nonzero α_n)



1. Lying on the supporting hyperplanes
2. Lying within the margin region but still on the correct side of the hyperplane
3. Lying on the wrong side of the hyperplane (misclassified training examples)

(Proof left as an exercise)

SVMs via Dual Formulation: Some Comments

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

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

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

Note: Both these ignore the bias term b otherwise will need another constraint $\sum_{n=1}^{N} \alpha_n y_n = 0$

- The dual formulation is nice due to two primary reasons
  - Allows conveniently handling the margin based constraint (via Lagrangians)
  - Allows learning nonlinear separators by replacing inner products in $G_{nm} = y_n y_m \boldsymbol{x}_n^\top \boldsymbol{x}_m$ by general kernel-based similarities (more on this when we talk about kernels)
- However, dual formulation can be expensive if N is large (esp. compared to D)
  - Need to solve for N variables $\boldsymbol{\alpha} = [\alpha_1, \alpha_2, \dots, \alpha_N]$
  - Need to pre-compute and store $N \times N$ gram matrix G
- Lot of work on speeding up SVM in these settings (e.g., can use co-ord. descent for $\boldsymbol{\alpha}$)

A Co-ordinate Ascent Algorithm for SVM

- Recall the dual objective of soft-margin SVM (assuming no bias b)

$$\underset{0\leq\boldsymbol{\alpha}\leq C}{\operatorname{argmax}} \sum_{n=1}^{N} \alpha_n - \frac{1}{2}\sum_{m,n=1}^{N} \alpha_m \alpha_n y_m y_n \boldsymbol{x}_m^\top \boldsymbol{x}_n$$

Note that $\boldsymbol{w} = \sum_{n=1}^{N} \alpha_n y_n \boldsymbol{x}_n$

- Focusing on just one of the components of $\boldsymbol{\alpha}$ (say α_n), the objective becomes

Can compute these in the beginning itself

Can efficiently compute it if we also store $\boldsymbol{w}$. It is equal to $\boldsymbol{w}^\top \boldsymbol{x}_n - \alpha_n y_n \|\boldsymbol{x}_n\|^2$

$$\underset{0\leq\alpha_n\leq C}{\operatorname{argmax}} \; \alpha_n - \frac{1}{2}\alpha_n^2 \|\boldsymbol{x}_n\|^2 - \frac{1}{2}\alpha_n y_n \sum_{m\neq n} \alpha_m y_m \boldsymbol{x}_m^\top \boldsymbol{x}_n$$

- The above is a simple quadratic maximization of a concave function: Global maxima
- If constraint violated, project α_n in $[0, C]$: If $\alpha_n < 0$, set it to 0, if $\alpha_n > C$, set it to C
- Can cycle through each coordinate α_n in a random or cyclic fashion

Solving for SVM in the Primal

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

$$\arg\min_{w,b,\xi} \frac{||w||^2}{2} + C\sum_{n=1}^{N}\xi_n$$
$$\text{subject to} \quad y_n(w^T x_n + b) \geq 1 - \xi_n, \quad \xi_n \geq 0 \qquad n = 1, \ldots, N$$

- Note that slack ξ_n is the same as $\max\{0, 1 - y_n(\boldsymbol{w}^\top \boldsymbol{x}_n + b)\}$, i.e., hinge loss for $(\boldsymbol{x}_n, y_n)$
- Thus the above is equivalent to minimizing the ℓ_2 regularized hinge loss

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

- Sum of slacks is like sum of hinge losses, C and λ play similar roles
- Can learn $(\boldsymbol{w}, b)$ directly by minimizing $\mathcal{L}(\boldsymbol{w}, b)$ using (stochastic) (sub)grad. descent
  - Hinge-loss version preferred for linear SVMs, or with other regularizers on $\boldsymbol{w}$ (e.g., ℓ_1)

SVM: At Test Time

- Prediction for a test point

$$y_* = \text{sign}(\boldsymbol{w}^\top \boldsymbol{x}_* + b) \qquad \text{(Approach 1)}$$

$$= \text{sign}\left(\sum_{n=1}^{N} \alpha_n y_n \boldsymbol{x}_n^\top \boldsymbol{x}_* + b\right) \qquad \text{(Approach 2)}$$

Dot product similarity of the test input $\boldsymbol{x}_*$ with the training input $\boldsymbol{x}_n$

- For linear SVMs, we usually prefer approach 1 since it is faster (just one dot product)
- The second approach's cost scales in the number of support vectors found by SVM (i.e., training examples with nonzero α_n). Also need to store them at test time
- The second approach is useful (and has to be used) for nonlinear SVMs where $\boldsymbol{w}$ cannot usually be expressed as a finite dimensional vector (more when we talk about kernel methods)

Multi-class SVM

- Multiclass SVMs (assuming $K > 2$ classes) use K wt vectors $\boldsymbol{W} = [\boldsymbol{w}_1, \boldsymbol{w}_2, \ldots, \boldsymbol{w}_K]$

  Prediction at test time: $\hat{y}_* = \mathrm{argmax}_{k \in \{1,2,\ldots,K\}} \boldsymbol{w}_k^\top \boldsymbol{x}_*$

- Like binary SVM, can formulate a maximum-margin problem (without or with slacks)

$$\hat{\mathbf{W}} = \arg\min_{\mathbf{W}} \sum_{k=1}^{K} \frac{||\boldsymbol{w}_k||^2}{2} \quad \text{s.t.} \quad \boldsymbol{w}_{y_n}^\top \boldsymbol{x}_n \geq \boldsymbol{w}_k^\top \boldsymbol{x}_n + 1 \quad \forall k \neq y_n$$

Score on correct class

Score on an incorrect class $k \neq y_n$

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

- The version with slack corresponds to minimizing a multi-class hinge loss

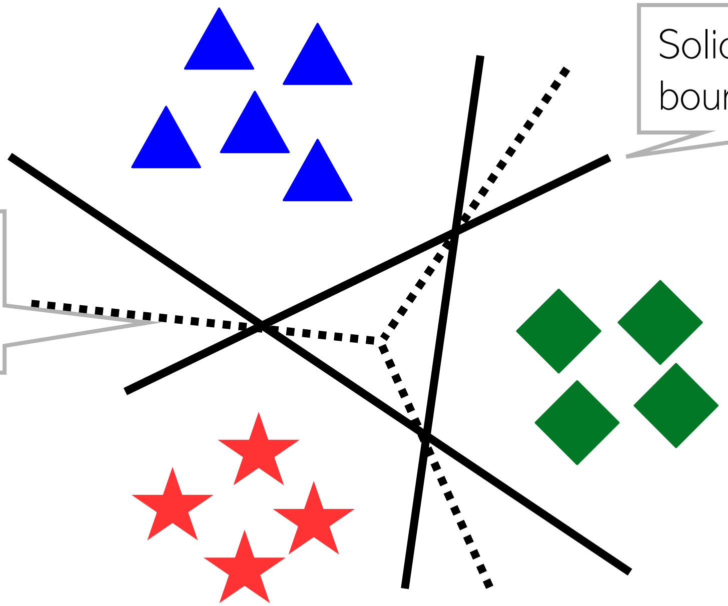
$$\mathcal{L}(\boldsymbol{W}) = \sum_{n=1}^{N} \max\left\{0, 1 + \max_{k \neq y_n} \boldsymbol{w}_k^\top \boldsymbol{x}_n - \boldsymbol{w}_{y_n}^\top \boldsymbol{x}_n\right\} + \frac{\lambda}{2} \sum_{k=1}^{K} ||\boldsymbol{w}_k||^2$$

Crammer-Singer Multi-class SVM

Loss=0 if score on correct class is at least 1 more than score on next best scoring class

Multi-class Classification using Binary Classification

- Can use binary classifiers to solve multiclass problems
- One-vs-All (also called One-vs-Rest): Construct K binary classification problems



Solid lines: one-vs-all boundaries

Dotted lines: Effective multi-class boundaries

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

$$y_* = \arg\max_k \sum_{j \neq k} \mathbf{w}_{j,k}^\top \mathbf{x}_*$$

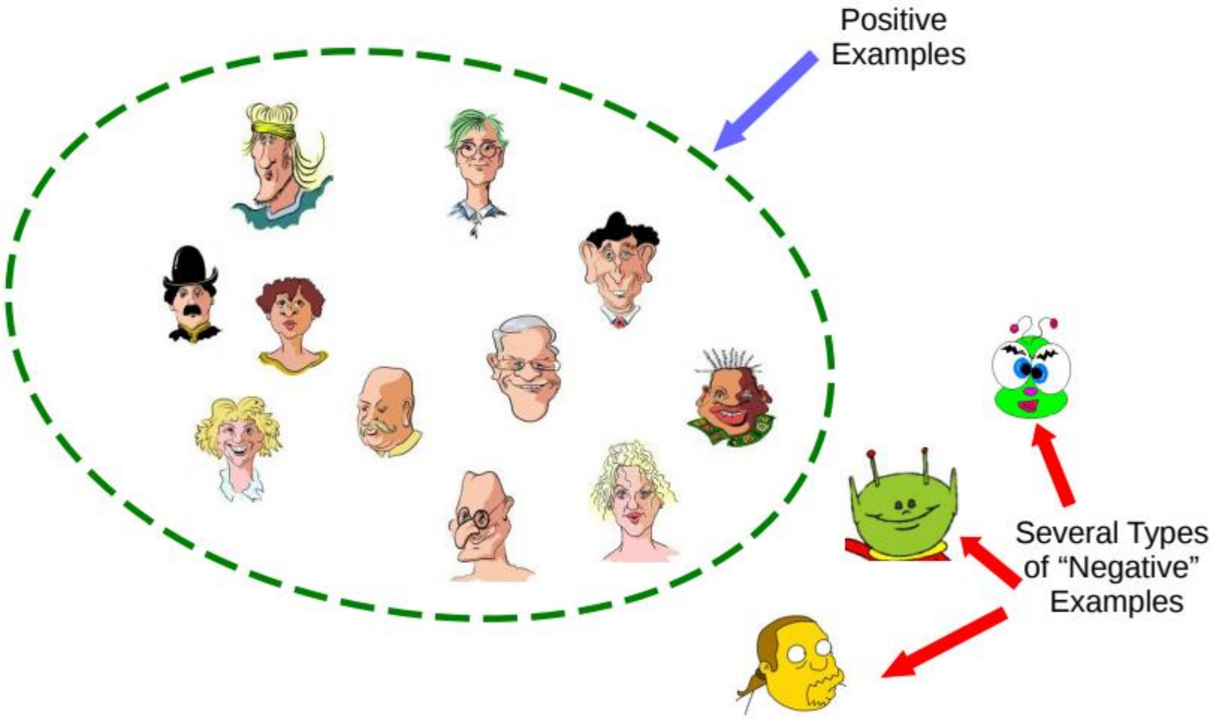
Whichever class k wins the most over other classes (or has the largest total scores against all other classes) is the prediction

Weight vector of the pairwise classifier for class j and k

Positive score if class k wins over class j in pairwise comparison

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

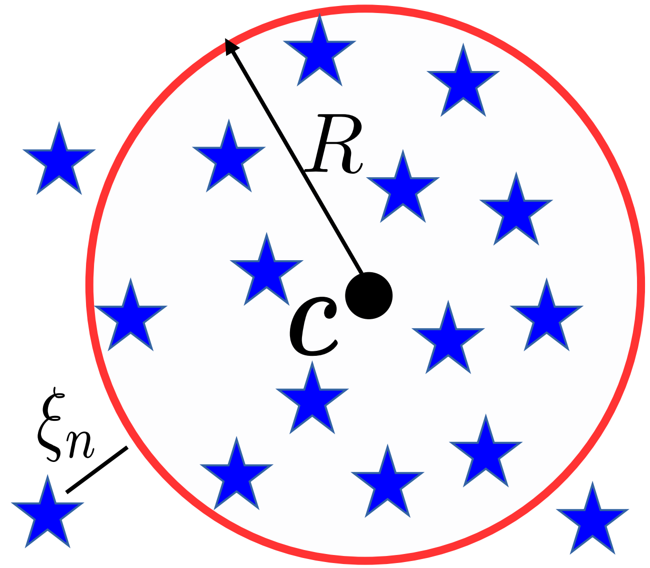
- One-class classification is an approach to learn using only one class of examples

Pic credit: Refael Chickvashvili

One-class Classification via SVM-type Methods

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

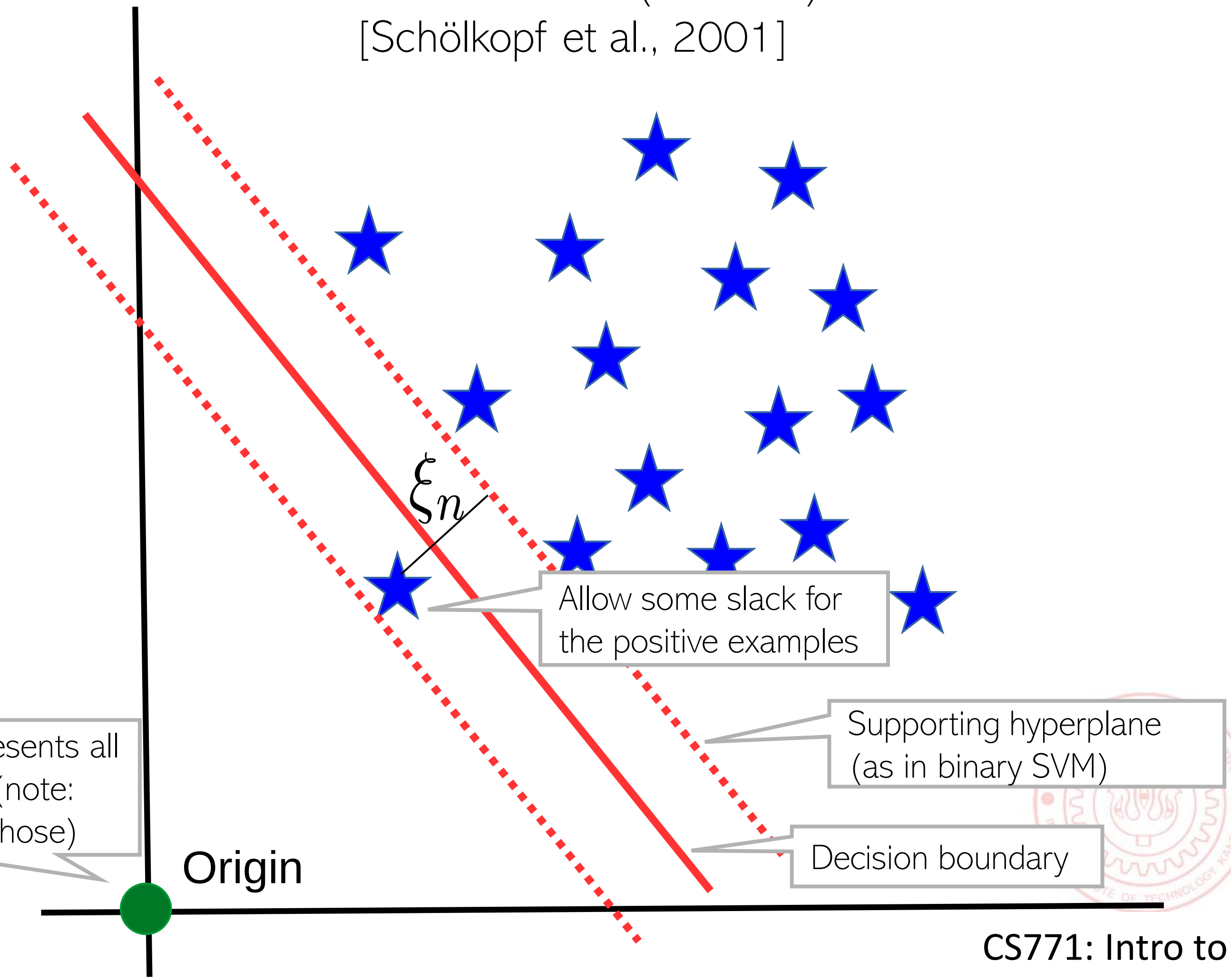
"Support Vector Data Description" (SVDD)
[Tax and Duin, 2004]



Learn a ball of smallest possible radius R centered at location c that encloses all positive examples (may allow some positives to "slack off" and fall outside)

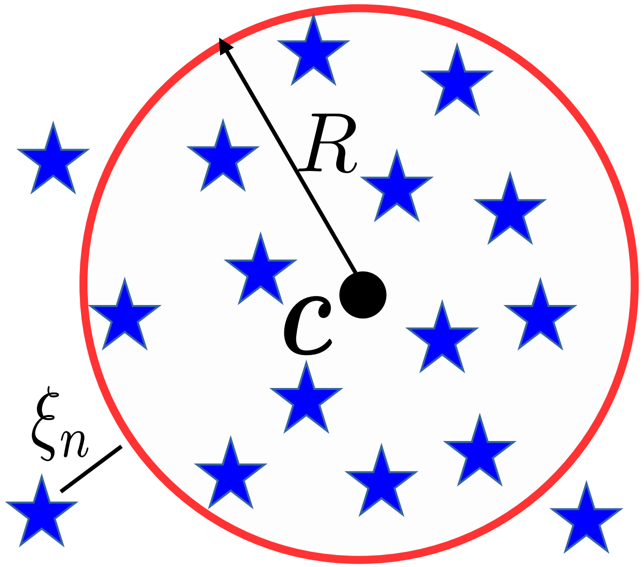
"One-Class SVM" (OC-SVM)
[Schölkopf et al., 2001]

Pretend that origin represents all the negative examples (note: we aren't given any of those)



One-class Classification via SVM-type Methods

"Support Vector Data Description" (SVDD)
[Tax and Duin, 2004]



Want to keep the ball's radius as small as possible

Hyperparameter ν to trade-off b/w the two terms

Want to keep training error (sum of slacks) to be small

$$\arg\min_{R,\boldsymbol{c},\xi} R^2 + \frac{1}{\nu N}\sum_{n=1}^{N}\xi_n$$

Want all training examples to fall within the ball (up to some slack ξ_n)

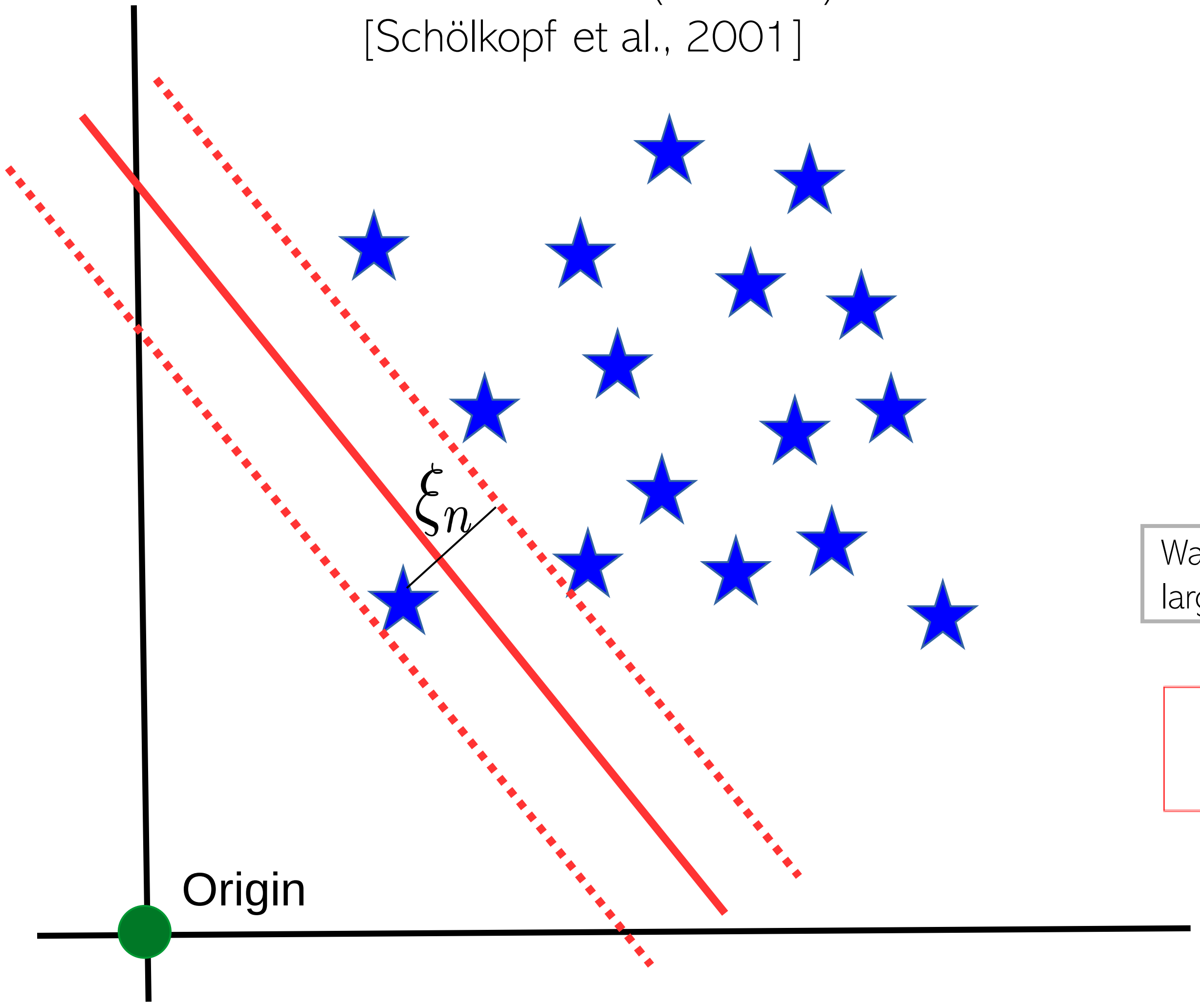
$$\text{s.t. } ||\boldsymbol{x}_n - \mathbf{c}||^2 \leq R^2 + \xi_n \quad \forall n$$

$$\xi_n \geq 0$$

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

One-class Classification via SVM-type Methods

"One-Class SVM" (OC-SVM)
[Schölkopf et al., 2001]



$$\arg\min_{\boldsymbol{w},\rho,\xi} ||\boldsymbol{w}||^2 + \frac{1}{\nu N}\sum_{n=1}^{N}\xi_n - \rho$$

$$\text{s.t. } \boldsymbol{w}^\top \boldsymbol{x}_n \geq \rho - \xi_n \quad \forall n$$

$$\xi_n \geq 0$$

Maximize the margin (similar to binary SVM)

Want to keep training error (sum of slacks) to be small

An offset term (want it large)

Want a sufficiently large score (say ρ)

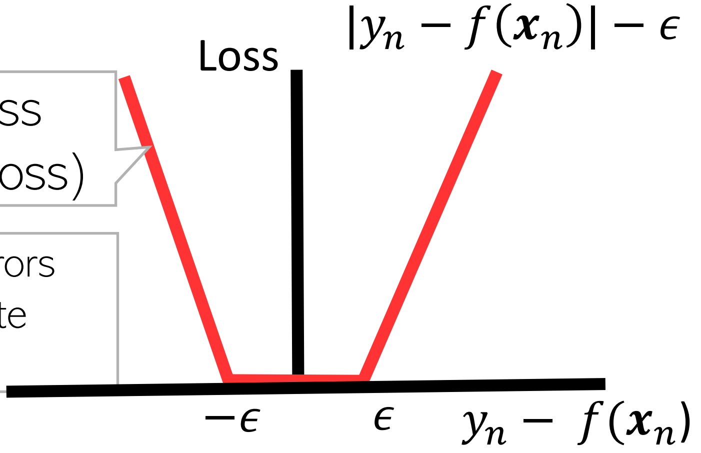
Prediction Rule: $y_* = +1 \quad \text{if } \boldsymbol{w}^\top \boldsymbol{x}_* > \rho$

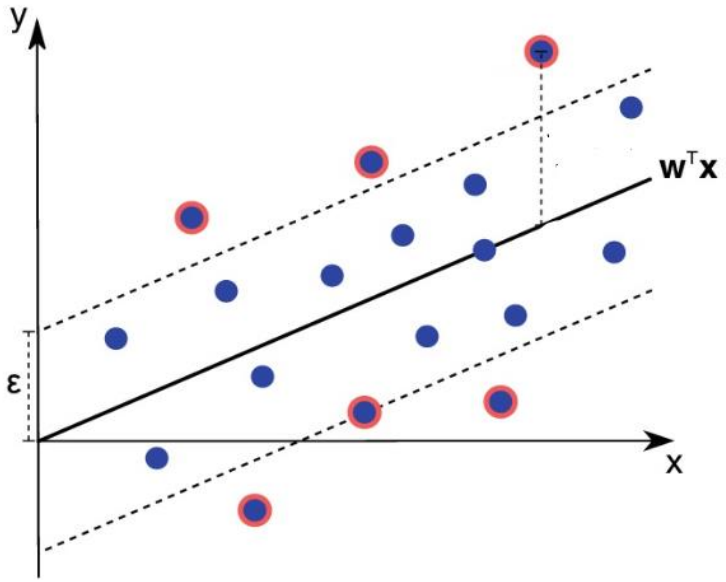
Support Vector Regression (SVR)

- SVR is an SVM variants for regression problems
- SVR uses ϵ-insensitive loss for regression

ϵ-insensitive loss (a.k.a. Vapnik loss)

Zero loss for small errors (say up to ϵ); absolute loss for larger errors





- Like the classification case, SVR also leads to a constrained optimization problem

Next class

- Nonlinear learning via Kernel Methods