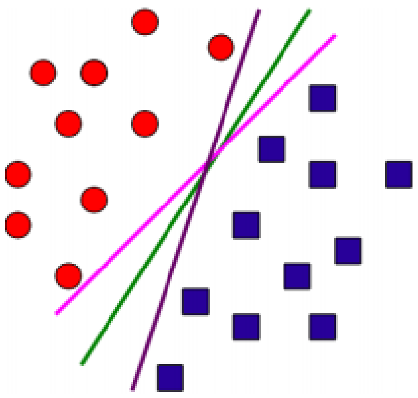
Large-Margin Classification and Support Vector Machine

CS771: Introduction to Machine Learning

Perceptron and (lack of) Margins

- Perceptron would learn a hyperplane (of many possible) that separates the classes



Basically, it will learn the hyperplane which corresponds to the $\boldsymbol{w}$ that minimizes the Perceptron loss

Kind of an "unsafe" situation to have – ideally would like it to be reasonably away from closest training examples from either class

- Doesn't guarantee any "margin" around the hyperplane
  - The hyperplane can get arbitrarily close to some training example(s) on either side
  - This may not be good for generalization performance

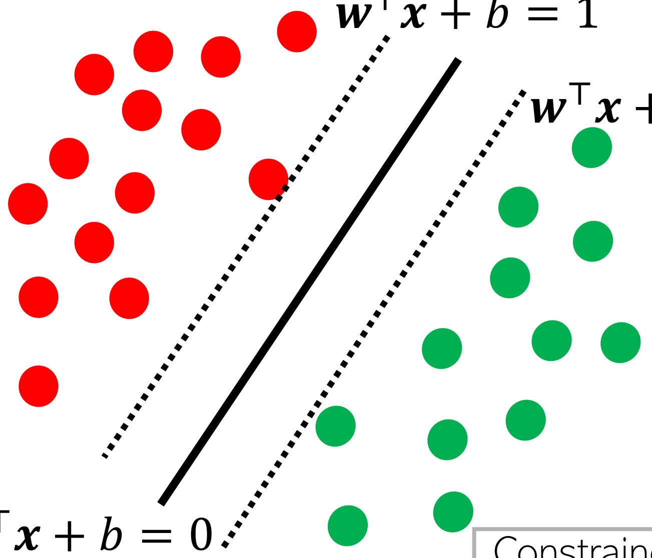
$\gamma > 0$ is some pre-specified margin

- Can artificially introduce margin by changing the mistake condition to $y_n \boldsymbol{w}^\top \boldsymbol{x}_n \leq \gamma$
- Methods like logistic regression also do not guarantee large margins
- Support Vector Machine (SVM) does it directly by learning the max. margin hyperplane

Support Vector Machine (SVM)

SVM originally proposed by Vapnik and colleagues in early 90s

- Hyperplane based classifier. Ensures a large margin around the hyperplane
- Will assume a linear hyperplane to be of the form $\boldsymbol{w}^\top \boldsymbol{x} + b = 0$ (nonlinear ext. later)



$$\boldsymbol{w}^\top \boldsymbol{x}_n + b \geq 1 \quad \text{if} \quad y_n = +1$$
$$\boldsymbol{w}^\top \boldsymbol{x}_n + b \leq -1 \quad \text{if} \quad y_n = -1$$
$$y_n(\boldsymbol{w}^\top \boldsymbol{x}_n + b) \geq 1 \quad \forall n$$

Distance from the closest point (on either side)

"Margin" of the hyperplane

Distance of an input $\boldsymbol{x}_n$ from the h.p.

$$\gamma = \min_{1 \leq n \leq N} \frac{|\boldsymbol{w}^\top \boldsymbol{x}_n + b|}{\|\boldsymbol{w}\|} = \frac{1}{\|\boldsymbol{w}\|}$$

Constrained optimization problem

Want the hyperplane $(\boldsymbol{w}, b)$ such that this margin is maximized (max-margin hyperplane) and $y_n(\boldsymbol{w}^\top \boldsymbol{x}_n + b) \geq 1 \quad \forall n$

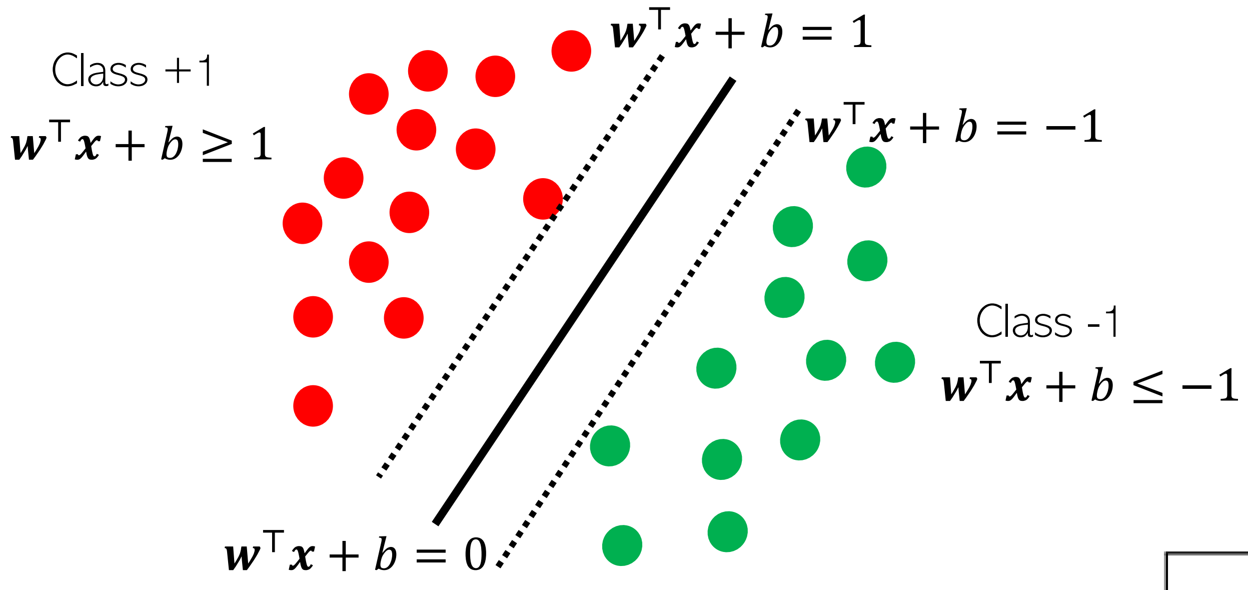
$$\text{Total margin} = \frac{2}{\|\boldsymbol{w}\|}$$

- Two other "supporting" hyperplanes defining a "no man's land"
  - Ensure that zero training examples fall in this region (will relax later)
  - The SVM idea: Position the hyperplane s.t. this region is as "wide" as possible

The 1/-1 in supp. h.p. equations is arbitrary; can replace by any scalar m/-m and solution won't change, except a simple scaling of $\boldsymbol{w}$

Hard-Margin SVM

- Hard-Margin: Every training example must fulfil margin condition $y_n(\boldsymbol{w}^\top \boldsymbol{x}_n + b) \geq 1$
- Meaning: Must not have any example in the no-man's land



- Also want to maximize margin $2\gamma = \frac{2}{||\boldsymbol{w}||}$
- Equivalent to <u>minimizing</u> $||\boldsymbol{w}||^2$ or $\frac{||\boldsymbol{w}||^2}{2}$
- The objective func. for hard-margin SVM

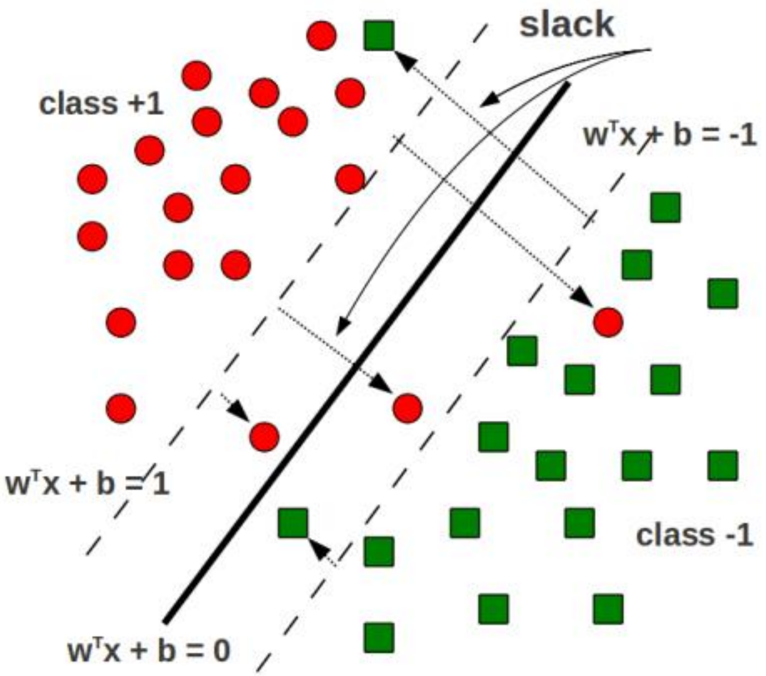
$$\min_{\boldsymbol{w},b} \quad f(\boldsymbol{w},b) = \frac{||\boldsymbol{w}||^2}{2}$$

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

Constrained optimization problem with N inequality constraints. Objective and constraints both are convex

Lagrange based optimization can be used to solve it

Soft-Margin SVM (More Commonly Used)



Note/verify that the slack for each training example is just the hinge loss

$$\xi_n = \max\{0, 1 - y_n(\boldsymbol{w}^\top \boldsymbol{x}_n + b)\}$$

- Allow some training examples to fall within the no-man's land (margin region)

  Helps in getting a wider margin (and better generalization)

- Even okay for some training examples to fall totally on the wrong side of h.p.
- Extent of "violation" by a training input $(\boldsymbol{x}_n, y_n)$ is known as slack $\xi_n \geq 0$
- $\xi_n > 1$ means totally on the wrong side

$$\boldsymbol{w}^\top \boldsymbol{x}_n + b \geq 1 - \xi_n \quad \text{if} \quad y_n = +1$$

$$\boldsymbol{w}^\top \boldsymbol{x}_n + b \leq -1 + \xi_n \quad \text{if} \quad y_n = -1$$

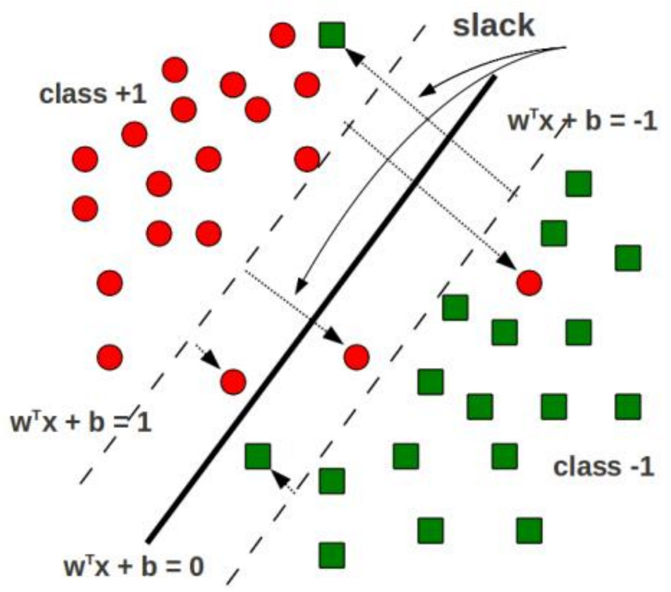
$$y_n(\boldsymbol{w}^\top \boldsymbol{x}_n + b) \geq 1 - \xi_n \quad \forall n$$

Soft-margin constraint

Soft-Margin SVM (Contd)

- 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)

Sum of slacks is like the training error



- The objective func. for soft-margin SVM

Lagrange based optimization can be used to solve it

Inversely prop. to margin

Trade-off hyperparam

training error

Constrained optimization problem with $2N$ inequality constraints. Objective and constraints both are convex

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

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

- 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 the SVM Problem

Solving Hard-Margin SVM

- The hard-margin SVM optimization problem is

$$\min_{\boldsymbol{w},b} \quad f(\boldsymbol{w},b) = \frac{||\boldsymbol{w}||^2}{2}$$
$$\text{subject to} \quad 1 - y_n(\boldsymbol{w}^T \boldsymbol{x}_n + b) \leq 0, \qquad n = 1, \ldots, N$$

- A constrained optimization problem. One option is to solve using Lagrange's method
- Introduce Lagrange multipliers α_n $(n = 1, \ldots, N)$, one for each constraint, and solve

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

- $\boldsymbol{\alpha} = [\alpha_1, \alpha_2, \ldots, \alpha_N]$ denotes the vector of Lagrange multipliers
- It is easier (and helpful; we will soon see why) to solve the dual: min and then max

Solving Hard-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 ascent can't easily be applied

- The dual problem (min then max) is

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

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

$$\frac{\partial \mathcal{L}}{\partial w} = 0 \Rightarrow \boxed{w = \sum_{n=1}^{N} \alpha_n y_n x_n} \qquad \frac{\partial \mathcal{L}}{\partial b} = 0 \Rightarrow \sum_{n=1}^{N} \alpha_n y_n = 0$$

α_n tells us how important training example (x_n, y_n) is

- The solution w is simply a weighted sum of all the training inputs
- Substituting $w = \sum_{n=1}^{N} \alpha_n y_n x_n$ in the Lagrangian, we get the dual problem as (verify)

This is also a "quadratic program" (QP) – a quadratic function of the variables α

$$\boxed{\max_{\alpha \geq 0} \mathcal{L}_D(\alpha) = \sum_{n=1}^{N} \alpha_n - \frac{1}{2} \sum_{m,n=1}^{N} \alpha_m \alpha_n y_m y_n (x_m^T x_n)}$$

IMPORTANT: inputs appear only as pairwise dot products. This will be useful later on when we make SVM nonlinear using kernel methods

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

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

G is an $N \times N$ p.s.d. matrix, also called the Gram Matrix, $G_{nm} = y_n y_m x_n{}^\top x_m$, and $\mathbf{1}$ is a vector of all 1s

(Note: For various SVM solvers, can see "Support Vector Machine Solvers" by Bottou and Lin)

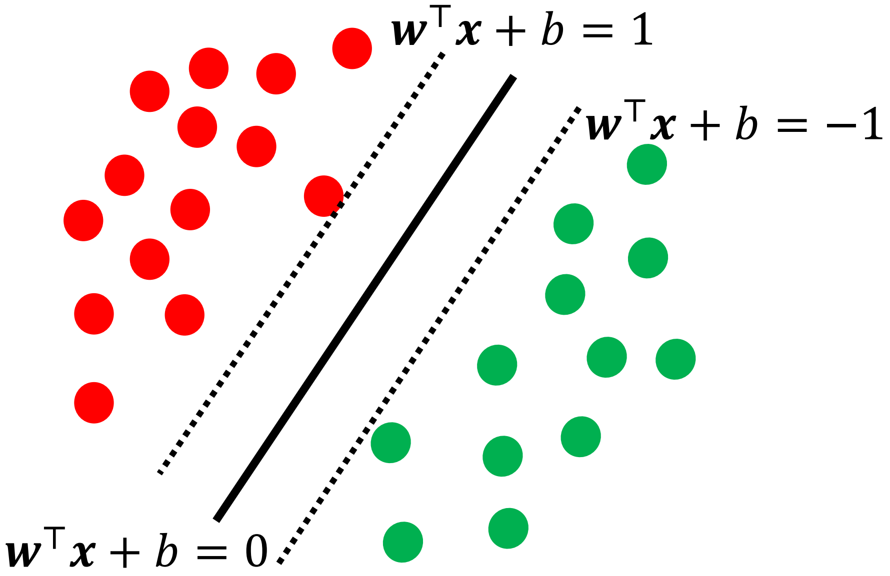
Solving Hard-Margin SVM

- One we have the α_n's by solving the dual, we can get $\boldsymbol{w}$ and b as

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

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

- A nice property: Most α_n's in the solution will be zero (sparse solution)



- Reason: KKT conditions
- For the optimal α_n's, we must have

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

- Thus α_n nonzero only if $y_n(\boldsymbol{w}^\top \boldsymbol{x}_n + b) = 1$, i.e., the training example lies on the boundary
- These examples are called support vectors

Solving Soft-Margin SVM

- Recall the soft-margin SVM optimization problem

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

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

- 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

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

- 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 $\boldsymbol{w}$, 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

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

- 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

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

- 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

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

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.

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

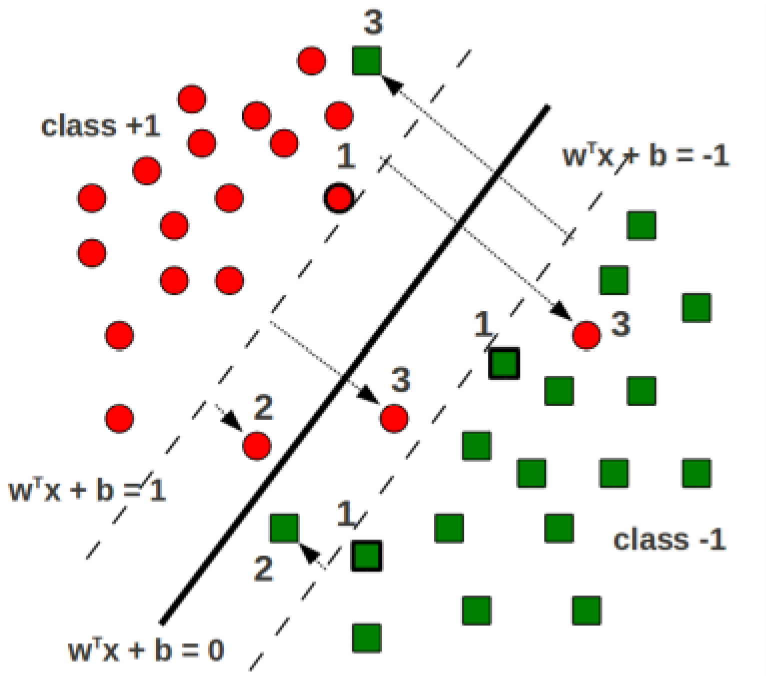
In the solution, $\boldsymbol{\alpha}$ will still be sparse just like the hard-margin SVM case. Nonzero α_n correspond to the support vectors

(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)

SVMs via Dual Formulation: Some Comments

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

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

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

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 $\boldsymbol{\alpha} = [\alpha_1, \alpha_2, \ldots, \alpha_N]$
  - 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 $\boldsymbol{\alpha}$)

SVM: At Test Time

- Prediction for a test point

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

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

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

- 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)

Solving for SVM in the Primal

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

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

- 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

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

- 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)

A Co-ordinate Ascent Algorithm for SVM

- Recall the dual objective of soft-margin SVM (assuming no bias $\boldsymbol{b}$)

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

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 $\boldsymbol{\alpha_n}$), the objective becomes

Can compute these in the beginning itself

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

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

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

SVM: Summary

- A hugely (perhaps the most, before deep learning became fashionable ☺) 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, sklearn also provides SVM
- Lots of work on scaling up SVMs* (both large $\boldsymbol{N}$ and large $\boldsymbol{D}$)
- Extensions beyond binary classification (e.g., multiclass, structured outputs)
- Can even be used for regression problems (Support Vector Regression)
- Nonlinear extensions possible via kernels

* See: "Support Vector Machine Solvers" by Bottou and Lin