Yet Another Tutorial on Support Vector Machines

This is a work in progress.

EnVision Systems LLC

Forward

- There are many excellent SVM tutorials on the net that emphasize various facets of SVM (and kernel methods in general) to various levels of detail.
- The fact that they emphasize different things or explain the same things differently helps learning.
- This is a work in progress. Things I will haven't gotten to:
  - Multi-class classification
  - Review of some SVM packages w/examples
  - An interesting kernel or 2
  - Loss function and comparisons to other methods
  - VC & SRM vs. ERM + standard Regularization

What's wrong with Backpropagation!?

- Curse of Dimensionality
  - To maintain generalization (as per the bias-variance trade-off) the number of observations (samples) required is exponential w/r the feature vector size
    - Typically induces sub-optimal procedures (e.g. wrapper methods, PCA) to find best small set of features which leads to suboptimal performance
- Feature Vector Encoding Requirement
  - Not every variable directly maps to being encoded as a number
- Error surface can lead to suboptimal performance
  - Local minimums + local optimizers $\Rightarrow$ suboptimal result.
  - Plateaus - No way of knowing whether training is done because it gradient could go to 0 for a long time.
- *Regularization, aka Weight Decay, can mitigate these issues in the same way it does, in the end, for SVM. (In fact merely stopping BPs training early can serve that purpose but that requires a validation set which may not be available).*
- *That being said, for many problems where there's lots of training data (and in particular even with respect to a large feature vector size) where there's significant non-linearities and getting a good result requires significant subsets of variables that only when taken to together help predict the target (in some non-linear way) BP can be great.*

Linear Discriminants

- (Hyper)Plane Decision Boundary

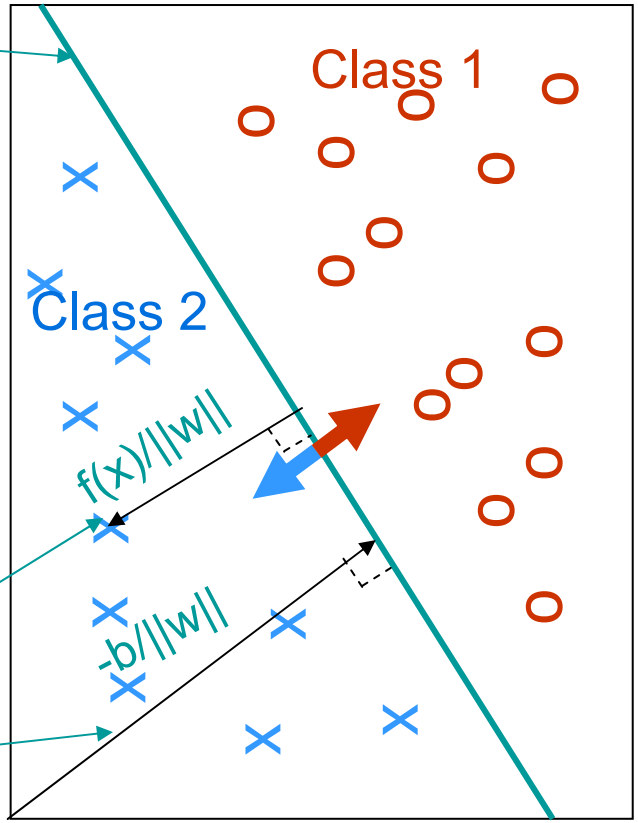
$$\mathbf{w}^T\mathbf{x}+b=0$$

  - *Note:* $k(\mathbf{W}^T\mathbf{X}+b)=0\ \forall k$
- **w** is the normal vector of the plane that defines the planes orientation
- **b** is the planes bias w/r to the origin
- If $f(x_i) = \mathbf{w}^T\mathbf{x}_i+b$ for an input point x_i, the predicted class of x_i is given by

$$\mathbf{g(x_i) = sign(f(x_i))}$$

- $||w||^*$ normalizes $f(x_i)$ to actual distance:
  - Distance of point x to plane = $f(x_i)/||w||$
  - Distance of origin to plane = $-b/||w||$
- *w^Tx is a Dot Product which can be written <w,x> (though it also encompasses a a more general definition of Dot Product)*

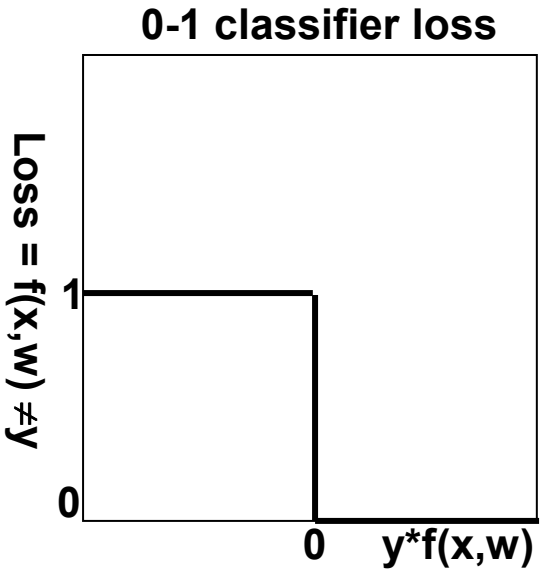
* *Note :* $||\mathbf{w}|| = \sqrt{(\sum_i w_i^2)} = length(w)$



An empirically linearly separable problem. A simple plane can separate the data with no errors

Loss and Risk

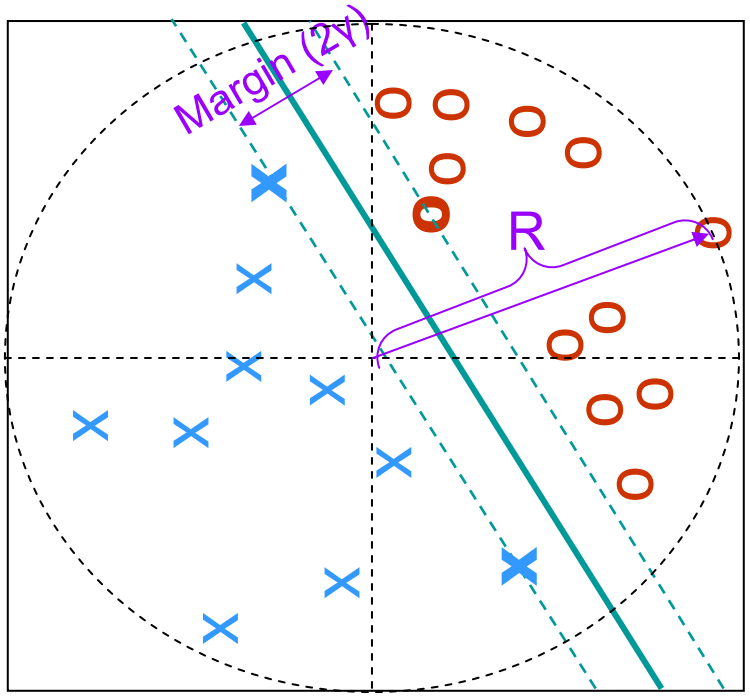
- Loss – L(f(x,w),y) - the cost associated for the discrepancy between the given response (target), y, of a given predictor/response pair, (x,y), and the machine learning algorithms response to x given its current parameters w.
  - Common: For classifiers, 0-1 Loss, i.e. $f(x,w) \neq y$, $(f(x,w) - y)^2$ for regression
  - Loss can incorporate different costs as desired say to deal with FP vs. FN
- $Risk_{emp}$- $E_i(L(f(x_i,w),y_i))$ - The average (expected) Loss over an entire training set of observations.
- Risk – Total Risk - The expected Loss over the underlying data joint probability distribution $\int_{xy} L(f(x,w),y)p(x,y)dxdy$



A standard classification Loss plot as shown for the classifier 0-1 Loss function. The x-axis is **y*f(x,w)** where the target values, $y \in \{-1,1\}$. This $\Rightarrow$ if the sign of the target and predicted values match (correct class) there's no loss, otherwise the loss=1

ENVISION SYSTEMS LLC

Perceptron Classifier



- A classifier for linearly separable data
- γ - Margin – Max distance from plane decision boundary containing no training data.
- R – distance from origin to farthest point.

Algorithm

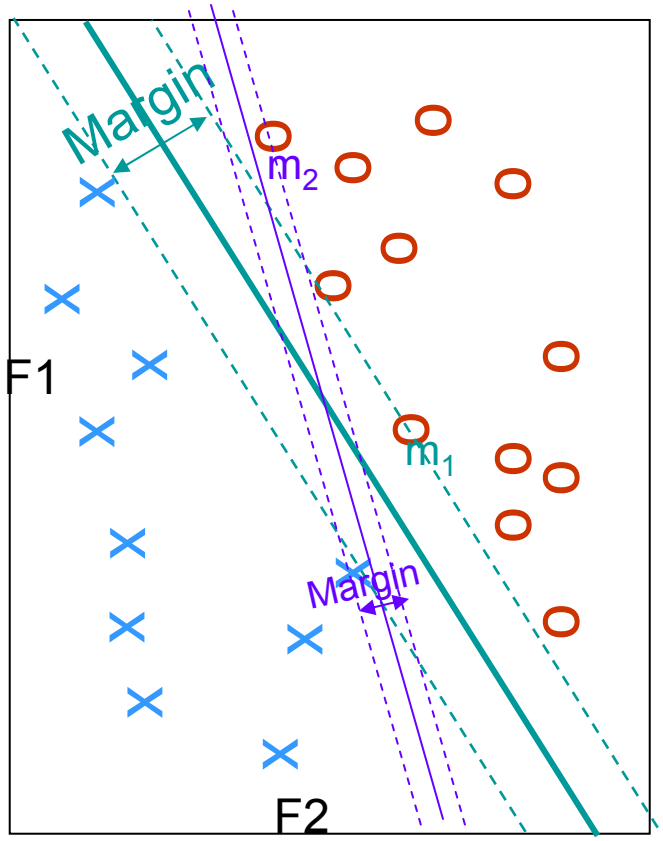
// y_i = {-1,1}. $\boldsymbol{x_i}$, $\boldsymbol{w}$ = $\Re^n$

1. **w**=0; b=0; K=0,ErrFlag=1;
2. while(ErrFlag) {
   A. ErrFlag=0;
   B. For all i {
      1. If $y_i(<\mathbf{w},\mathbf{x_i}>+b)<= 0$; *// error*
         A. $\mathbf{w}=\mathbf{w}+\eta y_i \mathbf{x_i}$; *// Gradient*
         B. $b=b + \eta y_i R^2$; *// Gradient*
         C. K++;
         D. ErrFlag=1; }}

- $y_i(<\mathbf{w},\mathbf{x_i}>+b) <= 0$ iff classification error
- Max iterations K <= $(2R/\gamma)^2$
  - Bigger Margin → Faster Convergence
  - Intuitively planes with bigger Margins are better solutions
  - **Independent of learning rate! (η)**

Comparing Margins

- Linear separability $\Rightarrow$ infinite number of perfect classifying planes
  - One with the biggest margin is intuitively the best
- Let w_1,b_1 and w_2,b_2 be parameters for two possible separating planes and m_1, m_2 be points on the respective margins of those decision planes. f(x)=<w,x>+b
  - $f(m_1)$, $f(m_2)$ can only be compared when normalized by w_1, w_2, respectively
    - OR
  - Set $f(m_1)$, $f(m_2)$ = 1. Largest margin $\Rightarrow$ smallest $||w_1||$, $||w_2||$
- Want smallest ||w|| for y(<x,w>+b) ≥ 1
  - All on margin points = 1
  - All off margin points > 1.



- Best Decision Plane with biggest margin
- Bad Decision Plane with much smaller margin

Basic Primal SVM Formulation

- "Basic" means only for linearly separable data.
- Want smallest ||w|| for $y(\langle x,w\rangle+b) \geq 1$. Standard form:

$$\tfrac{1}{2}||w||^2 \text{ s.t. } y_i(\langle x_i,w\rangle+b) \geq 1 \ \forall i$$

- Looks like an unpleasant constrained optimization problem.

Lagrangian Optimization

Optimization with equality constraints reformulated as unconstrained optimization.

Min f(**x**) with constraints {h_i(**x**)=0}

λ_i - Lagrangian multiplier for i^{th} constraint, $\boldsymbol{\lambda}$ is the vector of λ_i

$$L(x,\lambda) = f(x) + \sum_{i=1}^{n} \lambda_i h_i(x)$$

Optimum (x*, λ*) is at 0 gradient but not a saddle point

$$1) \nabla_x L(x^*, \lambda^*) = 0$$

$$2) \nabla_\lambda L(x^*, \lambda^*) = 0$$

$$3) \nabla^2_{xx} L(x^*, \lambda^*) \geq 0$$ (*PSD = No saddle*)

$$0 = \nabla_x L(x^*, \lambda^*) = \nabla_x f(x^*) + \sum_{i=1}^{n} \lambda_i \nabla_x h_i(x^*)$$

$$\Rightarrow \nabla_x f(x^*) \,||\, \nabla_x h(x^*)$$

$$s.t. \nabla_x h(x^*) = \sum_{i=1}^{n} \lambda_i \nabla_x h_i(x^*)$$ (Linear combo $\Rightarrow$ ∇h $\in$span{∇h_i})

$$but\ given\ h(x) = 0, \nabla_x h(x^*) \perp h(x)$$

$$\Rightarrow \nabla_x f(x^*) \perp h(x^*)$$

So at x* only way to decrease f is to move perpendicular to the constraint surface $\Rightarrow$ leaving the constraint surface $\Rightarrow$ violating the constraints.

$\Rightarrow$ We are at the (local) min x=x*

Lagrangian Optimization

Function to minimize:
$f(x) = x_1+x_2$ (a 45° ramp)
$\nabla f(x) = [1,1]$ *(gradients go up)*

Constraint:

$$h(x) = 0 = x_1^2 + x_2^2 - 1$$

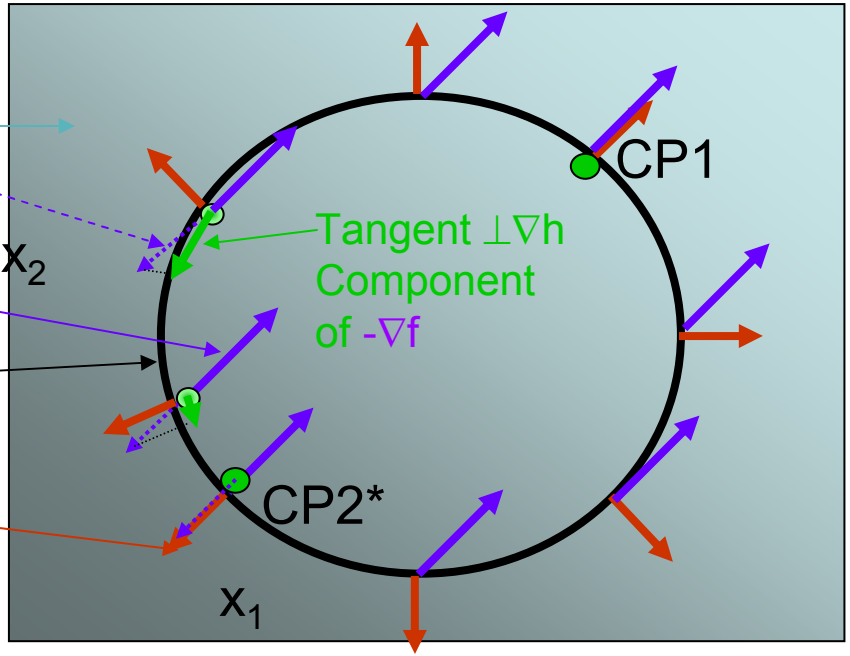
$\nabla h(x) = [2x_1, 2x_2]$

So, the problem is to find the min of this ramp, f(x), constrained to the unit circle.

At the critical points, CP1 & CP2, $\nabla h(x)$ and $\nabla f(x)$ align.

At critical points, there is no component of $\nabla f(x)$, that can move along the circle. Going downhill (on f(x)) would entail violating the constraint.

Second derivative test verifies a min for CP2*



Thought: Imagine a marble in circular tube affixed to a piece of flat piece of wood tilted s.t. CP2 is it's low point. The marble will only move if some portion of gravitational force is along the tube. This is true except at the critical points. At CP2 the gravity is pulling the marble solely ⊥ to the tube circle and it's at the minimum. By def. ⊥ to the direction of the tube means ∥ to the tubes gradient but so is the gravity gradient.

ENVISION SYSTEMS LLC

KKT (Karush Kuhn Tucker)

- Extends Lagrangian Optimization to inequality constraints.

$$L(x,\lambda) = f(x) + \sum_{i=1}^{n} \lambda_i h_i(x) + \sum_{i=1}^{n} \mu_i g_i(x) \quad s.t. g_i(x) \le 0$$

- Can always be written as g(x) ≤ 0
  - Example: -y(<w,x>+b)+1 ≤ 0 *(same as y(<w,x>+b) ≥ 1)*
- Not so different because inequality constraints only effect the solution when the exact equality holds at the minimum X.
  - Such a constraint is called an "Active Constraint".
- In the example above, if -y(<w*,x>+b*)+1 < 0, the constraint doesn't effect the solution. ⇒ The constraint -y(<w,x>+b)+1 ≤ 0 is "**inactive**".
- But if -y(<w,x*>+b*)+1 = 0 ⇒ the constraint -y(<w,x>+b)+1 ≤ 0 is "**active**" (but the equality holds, i.e. same as -y(<w,x>+b)+1 = 0)

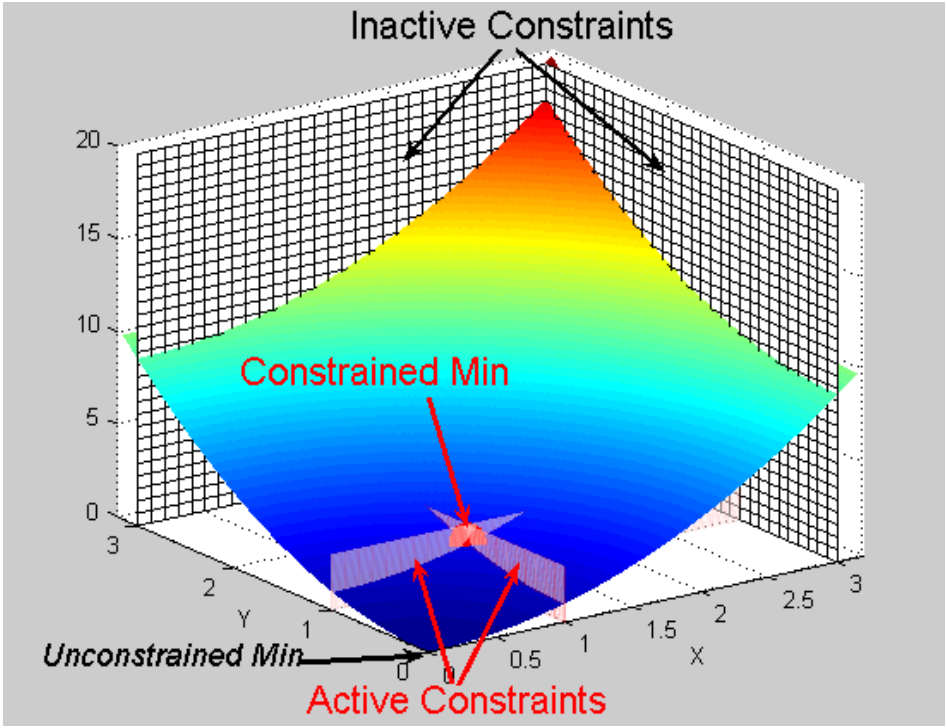
Active vs. Inactive Constraints

Example:

$f(\mathbf{X}) = ||\mathbf{X}||^2$; $\mathbf{X}=[X_1,X_2]$

Inequality Constraints $g_i(\mathbf{X})$:

1) $X_1 \geq 1 \rightarrow g_1(\mathbf{X}) = -X_1 + 1 \leq 0$
2) $X_1 \leq 3 \rightarrow g_2(\mathbf{X}) = X_1 - 3 \leq 0$
3) $X_2 \geq 1 \rightarrow g_3(\mathbf{X}) = -X_2 + 1 \leq 0$
4) $X_2 \leq 3 \rightarrow g_4(\mathbf{X}) = X_2 - 3 \leq 0$



- Unconstrained minimum is (0,0)
- Active constraints, g_1 and g_3, bound it at [1,1] at which point the equality holds, i.e.
  - $-X_1 + 1 = 0$ and $-X_2 + 1 = 0$
- Inactive constraints have no effect
  - If g2 & g4 were the only constraints, the minimum would still be (0,0)

ENVISION SYSTEMS LLC

KKT Theorem

If $x^* = \min f(x)$ s.t. $h(x) = 0$, $g(x) \leq 0$

$\rightarrow L(x, \lambda^*, \mu^*) = f(x^*) + \sum \lambda_i h_i(x^*) + \sum \mu_j g_j(x^*)$

x^* is a local min iff $\exists\ \lambda^*, \mu^*$ s.t.

$$0 = \nabla f(x^*) + \sum \lambda_i \nabla h_i(x^*) + \sum \mu_j \nabla g_j(x^*)$$

1. $\mu^*_j \geq 0\ \forall_j$
2. **$\mu^*_j = 0$ iff μ^* is an inactive constraint**
3. Hessian at solution is PSD (a real min)

Notes

- $\Rightarrow \mu_j g_j(x^*) = 0$ because either the constraint, $g_j(x^*)$, is active and so by definition $g_j(x^*) = 0$ or it's inactive and by **2.** $\mu_j = 0$.
- Doesn't say how to solve this. Seems to require simultaneously optimizing **x,** μ, and λ, . **It's hard!**
- Equality constraints can be rewritten using two inequalities
  - $h(x) = 0 \rightarrow g1(x) = h(x) \leq 0$ and $g2(x) = -h(x) \leq 0$
  - So, let's ignore equality constraints..

Optimization: Primal to Dual Form

- Under certain circumstances, the original lagrangian (in “primal” form) can be written in “dual” form who’s maximum is equal to the primal’s minimum but is easier to solve.
  - General: Convex optimization problems with affine constraints → no local minima
  - Specific: Linear and Quadratic programming problems. SVM is the latter.
- Under such circumstances

$$\max_u q(\mu) = \min_x L(x, \mu) = \min_x [f(x) + \sum \mu_j \nabla g_j(x)]$$

$$\text{s.t. initial } \mu \geq 0$$

- So, now instead having a simultaneous optimization of μ_j, and x, we
  1. Start with arbitrary $\mu \geq 0$
  2. Minimize x
  3. Maximize u
- Note: The dual will always be $\leq$ the primal. Having the max of the dual = min of the primal is called “strong duality”. Difference is called the “duality gap”. Strong duality means there is no duality gap.

Simple SVM Dual Formulation

- Previously, we gave the basic linear separable SVM optimization

$$\tfrac{1}{2}||w||^2 \text{ s.t. } y_i(<x_i,w>+b) \geq 1 \ \forall i$$

- Note $y_i(<x_i,w>+b) \geq 1 \rightarrow g_i(x) = 1-y_i(<x_i,w>+b) \leq 0$
- As a primal lagrangian

$$L(w, b, \alpha) = \tfrac{1}{2}||w||^2 + \sum_i \alpha_i (1-y_i(<x_i,w>+b))$$

  - **Note** that standard notation for SVM uses α vs. μ, and our variables are the weights (plane parameters) **w** and bias b vs. **x,** and our x's above are our training inputs, not variables to be optimized.
- To make the dual must first compute the minimum for w and b. This can be done analytically
  - $\nabla_w L(w, b, \alpha) = 0 = w^* + \sum_i \alpha_i y_i x_i \rightarrow w^* = \sum_i \alpha_i y_i x_i$
  - $\nabla_b L(w, b, \alpha) = 0 = \sum_i -\alpha_i y_i = \sum_i \alpha_i y_i$
    - So b drops out but yields a condition.
    - Since $y_i \in \{-1,1\} \rightarrow \sum_i \alpha_i^+ = \sum_j \alpha_j^-$

Simple SVM Dual Formulation Cont'd

- As a primal lagrangian

$$L(w, b, \alpha) = \tfrac{1}{2}||w||^2 + \sum_i \alpha_i (1-y_i(\langle x_i, w \rangle +b))$$

$$L(w, b, \alpha) = \tfrac{1}{2}||w||^2 + \sum_i \alpha_i (1-y_i(\mathbf{x}_i^T \mathbf{w}+b)) \quad \textit{(just a notation change)}$$

- For dual step 1, we minimized our variables, w and b, and obtained

$$\nabla L(w, b, \alpha): w^* = \sum_i \alpha_i y_i x_i \text{ and } 0 = \sum_i \alpha_i y_i$$

- Completing step 1 of dual process, substitute $\sum_i \alpha_i y_i x_i$ for w^*:

$$q(\alpha) = \sum_{ij} \alpha_i \alpha_j y_i y_j x_i^T x_j + \sum_i \alpha_i (1-y_i(x_i^T \sum_j \alpha_j y_j x_j +b))$$

$$q(\alpha) = \tfrac{1}{2}\sum_{ij} \alpha_i \alpha_j y_i y_j x_i^T x_j - \sum_{ij} \alpha_i \alpha_j y_i y_j x_i^T x_j + \sum_i \alpha_l - \sum_i \alpha_i y_i b$$

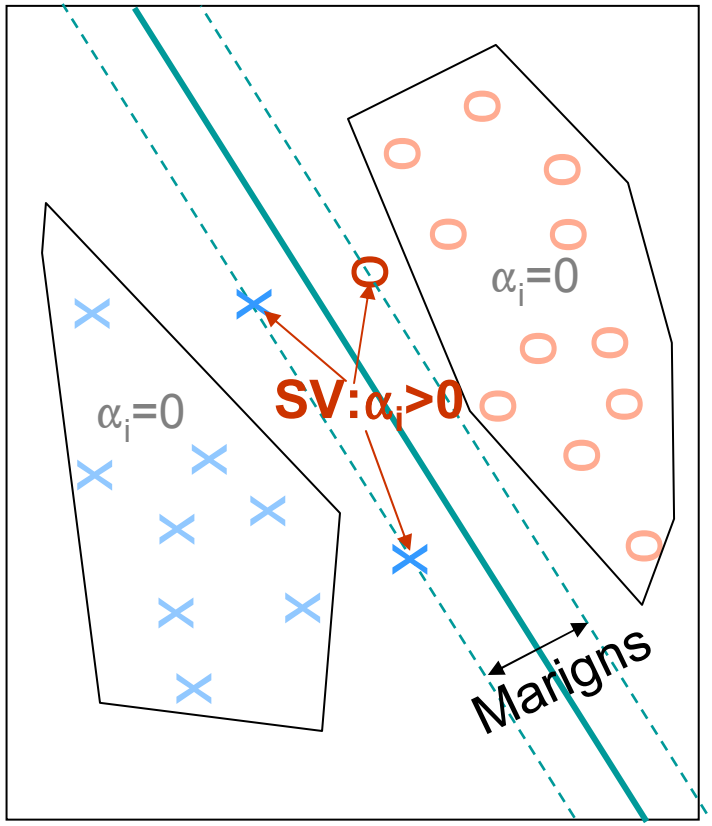
- Combining the first two terms, and given $\sum_i \alpha_i y_i = 0$ ➔ $\sum_i \alpha_i y_i b = 0$

$$\mathbf{q(\alpha) = -\tfrac{1}{2}\sum_{ij} \alpha_i \alpha_j y_i y_j x_i^T x_j + \sum_i \alpha_i}$$

- Maximizing q**(α)** w/r to α gives the optimal solution (second dual step)

What's a "Support Vector"?

- We have $w^* = \sum_i \alpha_i y_i x_i$ where $\alpha \geq 0$ are obtained via optimization (of the SVM dual)
  - plane defined by w is via a linear combination of the data points x s.t. sign of the contribution is given by y's class.
- So $g(x) = \text{sign}(w^T x + b) = \text{sign}(\sum_i \alpha_i y_i x_i^T x + b)$
- Note that for active constraints, $y_i(w^T x_i + b) = 1$. Those are, by definition, the points on the margin.
- By KKT, only active constraints have $\alpha_i > 0$ → only the active constraints effect $\sum_i \alpha_i y_i x_i^T x$
- Which is to say that only the points corresponding to the active constraints effect g(x) or any other term involving an α_i factor.
- These points are called **support vectors (SV), i.e. the SV in SVM,** because they define the decision plane. Others points are ignored.



The support vectors (SV) are the ones on the margin. Only they have non zero lagrange coefficients, α_i . Other points are "zero'd" out.

ENVISION SYSTEMS LLC

Classifying with SVM

- $g(x) = \text{sign}(w^{*T}x+b^*)$
- What's b*? It dropped out of the SVM dual lagrangian

$$q(\alpha) = -\tfrac{1}{2}\sum_{ij}\alpha_i\alpha_j y_i y_j x_i^T x_j + \sum_i \alpha_i$$

- Only for SVs: The constraint $y(w^{*T}x+b^*)\geq 1$ is active and so, for any SV, indexed "si" here, ➔ $y_{si}(w^{*T}x_{si}+b^*)=1$
- Solving for b* for any one SV (x_{si},y_{si})

$$b^* = y_{si} - w^{*T}x_{si}$$

- Now we can classify (i.e. evaluate for arbitrary (e.g. test) x.

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

- Note that since $w^* = \sum_s \alpha_s y_s x_s$,

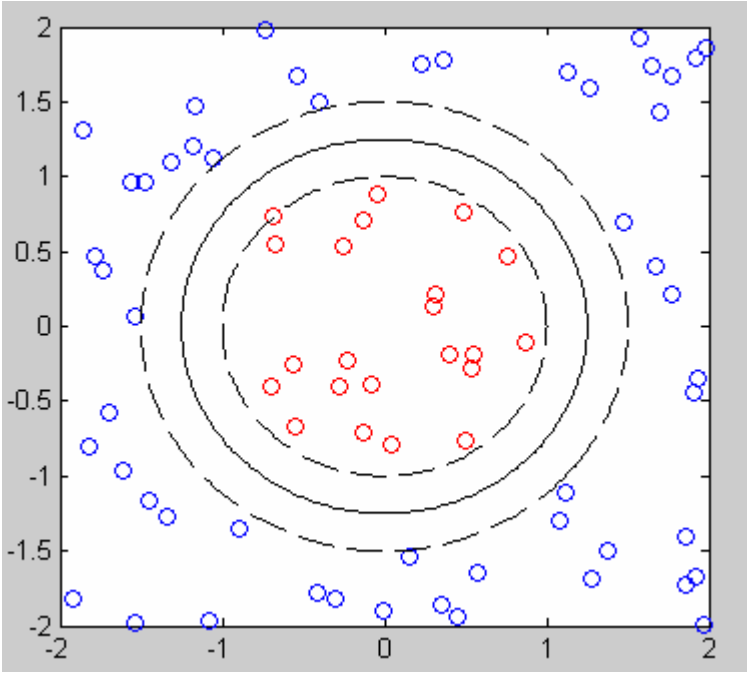
$$g(x) = \text{sign}(\sum_s \alpha^*_s y_s x_s^T x+b^*) \text{ where}$$

$$b^* = y_{si} - \sum_s \alpha^*_s y_s x_s^T x_{si}$$

  - Yes, this looks like a useless note but it becomes essential later.

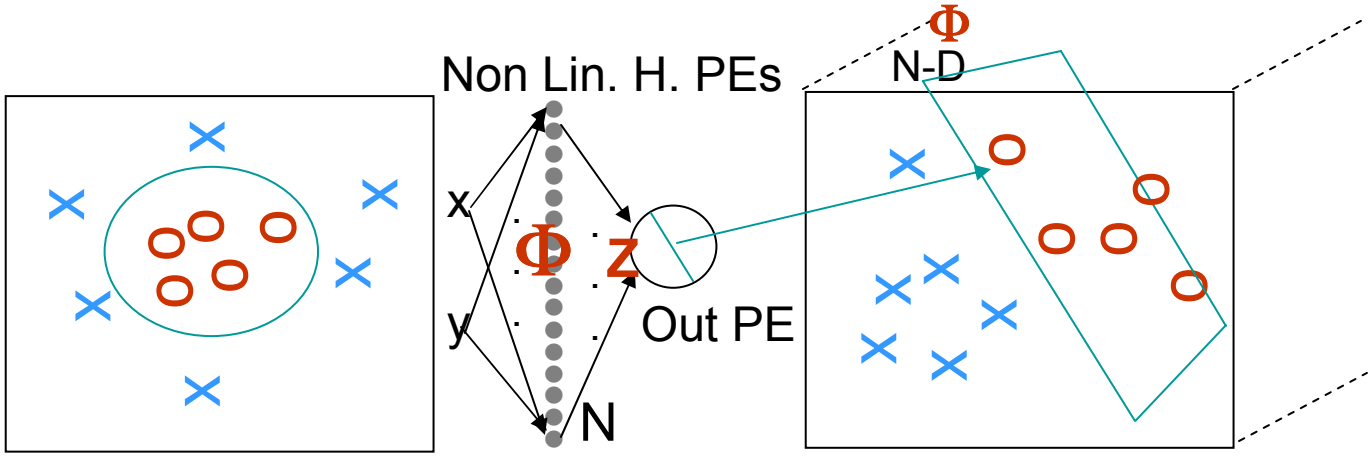
Non Linearly but Separable Data

- Obviously, we need to handle data that can't be separated by a simple plane (at least in the input space).
- Still, the concept of maximizing the margin is desirable
- As is concerning ourselves with the points near the decision boundary vs. elsewhere.
- *We eventually need to deal with data that isn't separable…but not yet*



Everything separable is linearly separable!
After transforming the data into higher dimensions

- 3 layer BP (MLP)** can approximate any L2 (i.e. “practical”) function to a given precision w/ enough hidden units *
  - A finite data set is, of course, always separable given a enough dimensions.
- The BP output unit is linear unit(s) and so this implies that the hidden layer has transformed the data **nonlinearly** into a **higher** dimensional space in which its decision boundary is a simple hyperplane.
  - Note that this says NOTHING about the loss of generality



Φ(x)→ Z

Where Φ is the entire hidden layer function

** RHN et. al showed this in ’87 whereas Kolmogorov’s Theorem just said, essentially, that such a non-linear Hid PE and output functions always existed for any function but not how to make it*

Where are we?

- So, there always exists a non linear transform $\Phi(x)$ to a vastly higher dimensional space that will give us linear separability.

$$g(x) = \text{sign}(w^T\Phi(x) + b)$$

- We don't know how to get an optimal $\Phi(x)$
- $\Phi(x)$ might be of infinite dimension($\Re^\infty$)
  - A perfect recipe for overfitting
  - Still will have the problem with local minima
  - Training an infinite BP would take some time (even with Lev. Marq.!).
  - Even just computing g(x) is impossible. Those infinite dot products are expensive!
- Can't do anything about BP's problems in particular with large $\Phi(x)$
- But, amazingly enough, we CAN do something about large or infinite dimensional $\Phi(x)$ using SVM via another mechanism

Kernels (Dot Product)

- Informally: Given two samples, x,y, a kernel, **k**, is a function that can be equivalently expressed as the dot product of some function, **Φ**, on each of those points.

$$\mathbf{k(x,y) = \langle \Phi(x), \Phi(y) \rangle}$$

- This kernel formulation is known as a "Dot product kernel"
- $\Phi(x) \in \mathcal{H}$ is a Hilbert Space: A vector space with a dot product. $\Re^d \in \mathcal{H}$
  - Often referred to as "Feature Space"
- $\langle \mathcal{H}, \mathcal{H} \rangle \rightarrow R$ - is an abstract dot product of which the standard dot product, e.g. $\langle \Phi(x), \Phi(y) \rangle = \Phi(x)^T \Phi(y)$ is just one (standard) definition of dot product.
- Dot products are, by definition, symmetric so k(x,y) = k(y,x)
- $k(x,y) \in \Re$ - just a number that corresponds to a similarity metric
- The simplest kernel is the standard dot product $x^t y$ where the Hilbert space is the same as the input space.

The Kernel Trick
Primal form no benefit

- Our standard decision function is:

$$g(x) = \text{sign}(w^{*T}\mathbf{x} + b^*)$$

- Yields a Linear Decision Boundary (LDB) in the input space.
- But we know that a $\Phi(x)$ can, through the use of a non-linear mapping of x, admit a Non-LDB via a LDB in the Feature/Hilbert space.
- Yielding

$$g(x) = \text{sign}(w^{*T}\Phi(\mathbf{x}) + b^*)$$

- Not any help. Intractable for $\Phi(x)$ of high or infinite dimensions.

The Kernel Trick
only in dual

- "Primal" form from last slide

$$g(x) = \text{sign}(w^{*T}\Phi(\mathbf{x}) + b^*))$$

  - Intractable because $\Phi(x)$ may be of infinite dimension

- But, in Dual form *(from the formulation of the dual optimization problem that* $\nabla L_w(w, b, \alpha)$*:* $w^* = \sum_i \alpha_i y_i x_i = \sum_s \alpha^*_s y_s x_s$, $s \in \{SV\} \equiv \alpha^*_s > 0$*)*

$$W^* = \sum_s \alpha^* y\, \mathbf{\Phi(x_s)} \Rightarrow g(x) = \text{sign}(\sum_s \alpha^*_s y_s \mathbf{<\Phi(x_s), \Phi(x)>} + b^*)$$

- But, by kernel definition (aka via "The Kernel Trick")

$$k(x,y) = <\Phi(x), \Phi(y)> \Rightarrow \mathbf{g(x) = \text{sign}(\sum_s \alpha^*_s y\, k(x_s,x) + b^*)}$$

- The intractable Φ is gone! Operations just in the input space.
  - Not totally free. Need to sum over all SVs (or in general over all α^*)
- Likewise $b^* = y_{si} - \sum_s \alpha^*_s y_s x_s^T x_{si}$ becomes $\mathbf{y_{si} - \sum_s \alpha^*_s y_s k(x_s, x_{si})}$
  - Any SVM package will compute b*.

Kernel Trick
Dual SVM Optimization

- The kernel trick is applied at all points with $\langle x_i, x_j \rangle$ (aka $x_i^T x_j$) because that implies $\langle \Phi(x_i), \Phi(x_j) \rangle$ which is intractable.
- From "way back" we gave the SVM dual optimization problem

$$\alpha^* = \text{Min}_\alpha \left[q(\alpha) = -\tfrac{1}{2} \sum_{ij} \alpha_i \alpha_j y_i y_j x_i^T x_j + \sum_i \alpha_i \right]$$

- Obviously, the same kernel trick game of
  - $X \rightarrow \Phi(x) \rightarrow \langle x_i, x_j \rangle \rightarrow \langle \Phi(x_i), \Phi(x_j) \rangle = k(x_i, x_j) \rightarrow$

$$\alpha^* = \text{Min}_\alpha \left[q(\alpha) = -\tfrac{1}{2} \sum_{ij} \alpha_i \alpha_j y_i y_j k(x_i, x_j) + \sum_i \alpha_i \right]$$

- This is the full "**Hard Margin**" SVM – can solve non-LDB problems but demands separability.
- So, the optimization problem itself only uses the kernel information.

Gram Matrix

- **Gram Matrix** – A Positive Definite Matrix K s.t. $K_{ij} = k(x_i,x_j)$
  i. This similarity matrix is the only data information the SVM has.
  ii. This is an NxN matrix where N is the number of training samples!
- Many different equivalent conditions imply a matrix is (semi)Positive Definite.
  - Eigenvalues ≥ 0
  - $K=X^TX$ – if each column of $X= \Phi(x)$ then $K_{ij} = k(x_i,x_j) = \Phi(x_i)^T\Phi(x_j)$ which is the definition of a dot product kernel
  - $y^TKy > 0 \quad \forall y$
  - *Think of a bowl with ellipsoid contours.*

Gram Matrix K =

$$\begin{bmatrix} k(x_1,x_1) & . & . & . & k(x_1,x_n) \\ . & . & & & . \\ . & & k(x_i,x_j) & & . \\ . & & & . & . \\ k(x_n,x_1) & . & . & . & k(x_n,x_n) \end{bmatrix}$$

$$= \begin{bmatrix} \phi(x_1) \\ \bullet \\ \bullet \\ \bullet \\ \phi(x_n) \end{bmatrix} \times \begin{bmatrix} \phi(x_1) & \bullet & \bullet & \bullet & \phi(x_n) \end{bmatrix}$$

Positive Definite (PD) Kernels

- A function when applied to all possible pairs of points from any set of points taken from the input space yields a positive definite Gram matrix.

$\Rightarrow$ k(x,x) $\geq 0 \; \forall$ x $\in \; \chi$

$$\begin{bmatrix} k(x,x) & k(x,y) \\ k(y,x) & k(y,y) \end{bmatrix} PD \; \forall x, y \in \chi$$

$\Rightarrow$ Another way of showing kernels are symmetric

$\Rightarrow$ Cauchy-Schwarz inequality (from the determinent of 2x2 Gram matrix)

$$K(x,y)^2 \leq k(x,x)\, k(y,y)$$

$\Rightarrow$

$$-1 \leq \frac{k(x,y)}{\sqrt{k(x,x)k(x,y)}} \leq 1$$

- Looks like a Pearson's coefficient. Sensible when kernels viewed as similiarty metrics

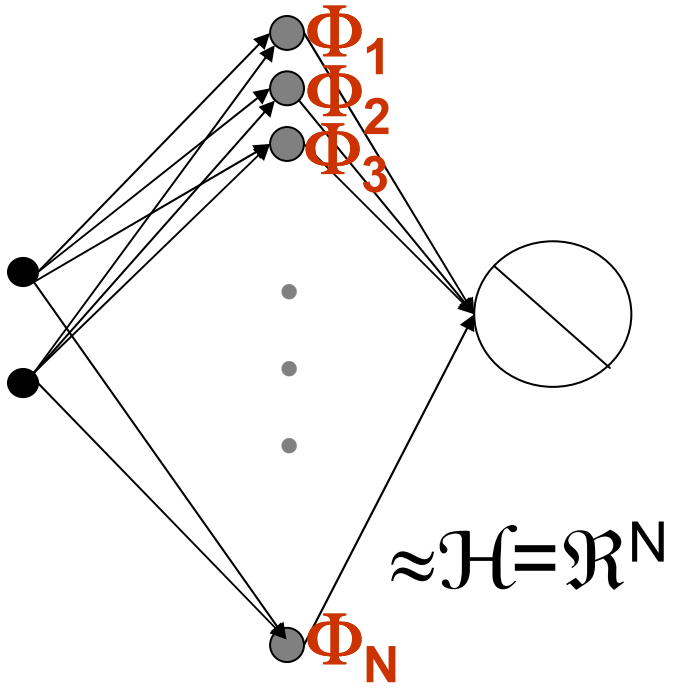
Gram Vector/Matrix Formulation

- $\langle w,\Phi(x)\rangle = \sum_{s\in SV}\alpha_s y_s\, k(x_s,x) = v_s^t k$
  - $\mathbf{v_s} = (\alpha .* \mathbf{y})^t$
    - *weighted class labels*
    - .* is vector element-wise multiplication
  - $\alpha = [\alpha_1,.., \alpha_s,\ldots \alpha_{|SV|}]^t$
  - $\mathbf{y} = [y_1,.., y_s,\ldots y_{|SV|}]^t$
  - $\mathbf{k} = [k(x_1,x),\ldots,k(x_s,x),\ldots,k(x_{|SV|},x)]$
- $q(\alpha) = \sum_{ij}\alpha_i\alpha_j y_i y_j k(x_i,x_j) + \sum_i \alpha_l =$
  - $\sum_{ij}\alpha_i y_i k(x_i,x_j)\alpha_j y_j = v^t\mathbf{K}v$
  - K is the Gram Matrix
  - $q(\alpha) = v^t\mathbf{K}v + \sum_i\alpha_l$ *the standard quadratic in matrix form.*

Kernels (Mercer)

- Mercer Kernels – Another equivalent kernel definition
  - $\int\int k(x,y)f(x)f(y)dxdy \geq 0 \ \forall f(x)$ s.t. $\int f(x)^2 dx < \infty$
  - Corresponds to a set of orthonormal basis functions $\Phi_i \in \Re$
    - $\approx$ BP Hidden units but those aren't necessarily orthogonal
    - *FYI An Orthogonal functions* → $\int_x \Phi_l(x)\ \Phi_j(x)dx = \delta_{ij}$ *(think FFT)*
  - Can be shown to be the same as a dot product and Pos Def kernel

Backpropagation NN



Common Kernels - Gaussian

- Isotropic Gaussian

$$k(x,y) = e^{(-\frac{1}{2}||x-y||^2/\sigma^2)}$$

- Most widely used kernel.
- Implicit Φ is of infinite dimension but evaluating the kernel is of order N (dim of input space).
- Obviously a good idea to normalize the inputs.
- "optimal" σ via cross validation over a reasonable range of values
- Many just use the default σ that the package gives them set w.r.t. the norm of the data! Plenty of examples show that this can be very suboptimal.
- Very obvious tie-in as kernel as similarity matrix.
- Generally, this kernel gives good results and is used when a simple distance metric makes sense as a measure of similarity between a pair of observations

Common Kernels - Polynomial

- Polynomial

$$k(x,y) = (x+y+1)^d$$

  - Probably 2nd most widely used kernel.
  - Φ dimensions are combinatorial in *d* and *N* but **k(x,y)** is of order N
  - Informative Example: Let d=2 and N=2 → $x=[x_1,x_2]$, $y=[y_1,y_2]$
    - → $\Phi(x) = \left[1, \sqrt{2x_1}, \sqrt{2x_2}, x_1^2, x_2^2, \sqrt{2x_1x_2}\right]$
    - $k(\Phi(x), \Phi(y)) = (x+y+1)^2 = \langle\Phi(x), \Phi(y)\rangle$
    - Imagine if d or N equal just 3…or both were. Very long vectors but kernel trick dot product of order N
- d typically chosen by cross-validating over a reasonable range of values.
- Much harder to visualize then Gaussian. This might be a kernel to try blindly after the Gaussian is tried.

Kernels – Testing them

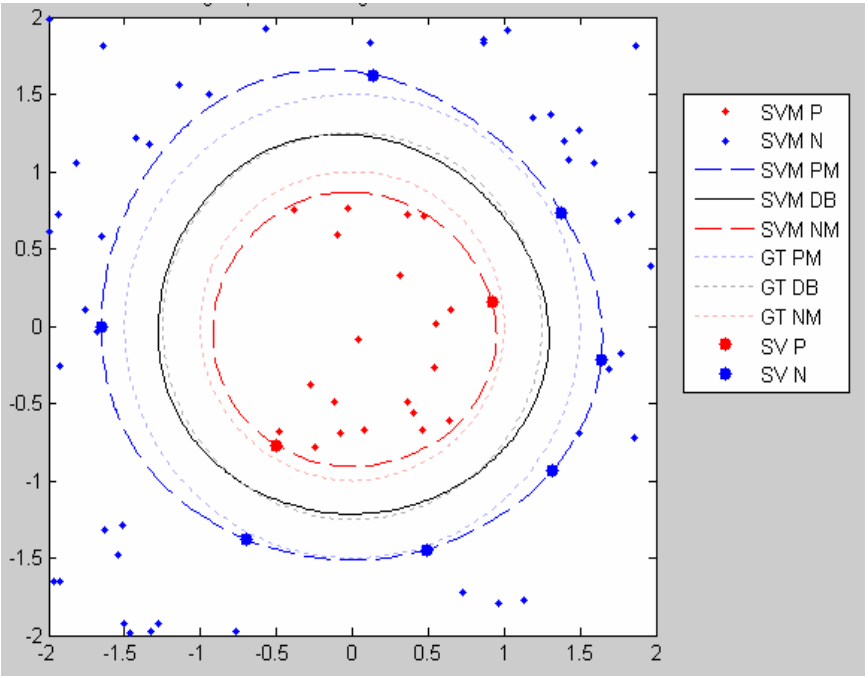
- Most kernels are off-the-shelf and so there's no need to test them to see if they valid w.r.t. any one of the formulations.
- If you have a custom kernel then either
  - Find the dot product kernel formulation's Φ function.
  - A mathematical proof that it's PD or Mercer or some equivalent.
  - Empirical: Show the eigenvalues of your Gram matrix are ≥ 0. Note that that this doesn't prove that the kernel is positive definite but it will suffice for the convexity required for the optimizer's convergence since the Gram matrix is all the optimizer will see. Note that for large matrices this could be computationally intractable and you could get slightly negative eigenvalues just from numerical issues. Probably could get away with do this via sampling subset Gram Matrices over night?
  - SVM has worked for some kernels that violate the constraints anyway!
  - *If anyone knows of another practical method please tell me.*

Kernels – Bottom Line

- Computing k(x,y) does NOT typically entail operating in or even knowing about Φ (That's the whole point!)
- Most people just use the gaussian kernel blindly to reasonably good but sub-optimal effect…but you shouldn't (given enough data to cross validate).
- Though people use this to good effect without doing much pre-processing of the data, that has been shown to yield suboptimal results. Many bad input features can seriously degrade the performance.
- Note that many machine learning paradigms use dot products (e.g. $<x_i,x_j>$) and so are can be extended to non linear forms (PCA→ KCA, NN→ KNN)
- Most SVM packages don't just allow you to hand the them the Gram matrix already computed which would seem to be the obvious way to handle user defined kernels. Some, however, allow you to give them a kernel definition. Perhaps this is to alleviate the need to store the N^2 Gram matrix (N=# of observations). They can compute it piecemeal on the fly but at greater computational expense. Frankly, they should support both options and let the user decide how much memory they can spare.

Example of Hard-Margin SVM

- P – Positive class i.e. inside circle.
- N – Negative class i.e. outside circle
- SVM P & SVM N are the points as classified.
- SVM PM, SVM DB, SVM NM are the SVM f(x) = 1,0, and -1 contours respectively. This corresponds to the Positive Margin, Decision Plane (it's a plane w in $\in \mathcal{H}$), and Negative Margin.
- GT PM, GT DB, GT NM are the corresponding Ground Truth contours (R=1, 1.25, 1.5)
- SV P, SV N are the support vectors for the Positive and Negative classes respectively.
- *Note – this isn't really a hard margin SVM but a Soft Margin SVM set to pretty much mimic a hard margin SVM*



EnVision Systems LLC

Soft Margin (i.e. non separable) SVM

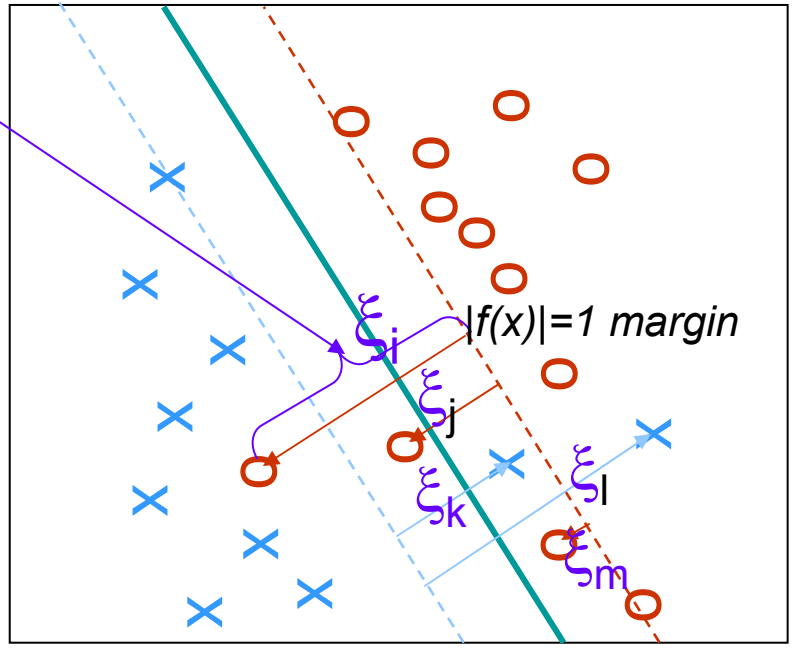
- Slack Variables - ξ_i - added to each point/constraint to "sop" up the error due to being on the wrong side of its margin.
- Observation i's constraint becomes

$$y_i(<w,x_i>+b) \geq 1- \xi_i \text{ s.t. } \xi_i \geq 0$$

  - This is really two constraints.
  - $\xi_i \geq 0$ means slacks only for points on the wrong side of their margin.
- Now, in addition to w, must minimize ξ otherwise can fit anything.

$$\min_{w,\xi}(||w||^2 + \mathbf{C}\ \Omega_c(\xi))$$

- Function Ω_c is convex to yield a convex (e.g. quadratic) optimization problem ($\Rightarrow$ no local minimums) that admits a dual formulation.



- Higher C means the optimization increasingly cares about minimizing the slack variables and so getting a large margin, via minimizing ||w||, will suffer to fit all the points which therefore decreases generalization.

ENVISION SYSTEMS LLC

2-Norm Slack Variables

- Let $\Omega(\xi) = \|\xi\|^2$
- Primal Optimization:

$$\text{Min}_{w,b,\xi}(½\|w\|^2+C\|\xi\|^2) \text{ s.t. } y_i(<w, \Phi(x_i)>+b)\geq 1-\xi_i,\ \xi_i \geq 0,\ \forall_i$$

- Note that if $\xi_i < 0$ since $\xi_i^2 > 0$, $\Rightarrow \xi_i \geq 0$ is a redundant constraint
- So full primal lagrangian is

$$L(w, b, \xi, \alpha) = ½\|w\|^2 + \sum_i \alpha_i(1- \xi_i - y_i(<w, \Phi(x_i)>+b))$$

- For dual formulation must min w,b,ξ via $\{\nabla L_{\mathbf{w}}, \nabla L_{\mathbf{b}}, \nabla L_{\xi}\}=0$

$$w^*=\sum_i \alpha_i y_i \Phi(x_i),\ \sum_i \alpha_i y_i = 0,\ \mathbf{C\xi_i = \alpha_i}$$

- Substituting (and lots of algebra) above into primal yields the dual optimization form

$$q(\alpha) = \max_{\alpha>0}(-½\sum_{ij} \alpha_i \alpha_j y_i y_j (k(x_i,x_j)+ \delta_{ij}/C) + \sum_i \alpha_i)$$

$$= \max_{\alpha>0}(-½ v^t(K+\mathbf{I/C})v+\sum_i \alpha_i)$$

Where $v = \alpha$.* v, $\mathbf{I}$ is the identity matrix, and $\alpha > 0$

Analysis of 2-norm Slack Var SVM

- Hard margin vs. 2-norm Soft margin dual

$$\max_{\alpha>0}(\tfrac{1}{2}v^tKv+\Sigma_i\alpha_i) \text{ vs. } \max_{\alpha>0}(\tfrac{1}{2}v^t(K\mathbf{+I/C})v+\Sigma_i\alpha_i)$$

- Only difference from hard margin is I/C term.
  - Makes diagonal of Gram Matrix, K, larger, more stable (increases eigenvalues by 1/C)
  - $\lim_{C\to 0} K+I/C = I/C = I/0 = I\infty$
  - $\lim_{C\to \infty} K+I/C = K$ yielding hard margin form
- **2-norm slack variable minimization very sensitive to outliers (for the same reason that LMS is sensitive to outliers).**
  - Conceptually, the system will do anything to attenuate large ξ_i because given a quadratic cost say $\xi_j=n\xi_i \Rightarrow \text{cost}(\xi_j)/\text{cost}(\xi_i) = \text{cost}(n\xi_i)/\text{cost}(\xi_i) = (n\xi_i)^2/\xi_i^2 = n^2$
  - We didn't go through all this margin stuff to be nailed by some outliers!

1-Norm Slack Variables (C-SVC)

- Let $\Omega(\xi) = \Sigma\xi_i$
- Primal Optimization:

$$\text{Min}_{w,\xi}(\tfrac{1}{2}||w||^2+C\Sigma\xi_i) \text{ s.t. } y_i(<w, \Phi(x_i)>+b)\geq 1-\xi_i,\ \xi_i \geq 0,\ \forall_i$$

- Since $C\xi_i$ can be < 0, $\xi_i \geq 0$ ($-\xi_i \leq \mathbf{0}$) is **NOT** a redundant constraint

  $\Rightarrow$ Need a new lagrangian term with it's own lagrangian variables, "r_i"
- So full primal lagrangian is

$$L(w, b,\xi ,\alpha,r) = \tfrac{1}{2}||w||^2 +C\Sigma\xi_i + \Sigma_i\alpha_i (1- \xi_i - y_i(<w, \Phi(x_i)>+b)) - \Sigma_i r_i \xi_i$$

- Again, for dual formulation must min w,b,ξ via $\{\nabla L_{\mathbf{w}},\nabla L_{\mathbf{b}},\nabla L_{\xi}\}=0$

$$w^*=\Sigma_i\alpha_i y_i\Phi(x_i),\ \Sigma_i\alpha_i y_i=0,\ C-\alpha_i-r_i=0$$

- Substituting, etc., above into primal yields the dual optimization form

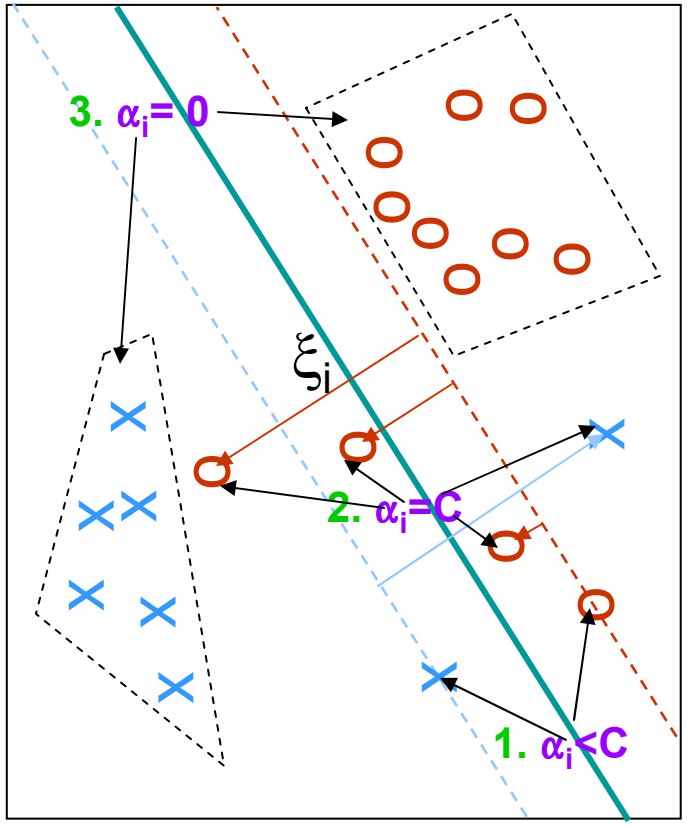
$$\mathbf{q(\alpha) = \max_{\alpha>0}(-\tfrac{1}{2}\Sigma_{ij}\alpha_i\alpha_j y_i y_j k(x_i,x_j) +\Sigma_i\alpha_i)}$$

$$\mathbf{= \max_{\alpha>0}(-\tfrac{1}{2}v^tKv+\Sigma_i\alpha_i)}$$

- Same as the hard margin form! (but has the $C-\alpha_i-r_i=0$ constraint)

Analysis of 1-norm Slack Var SVM (C-SVC)

- From KKT: Given u the lagrangians on $g(x) \leq 0$ then $u_i g_i(x) = 0$
  - g(x) is active $\Rightarrow$ $g_i(x)=0$ or inactive $\Rightarrow$ $u_i=0$
- Given $C-\alpha_i-r_i=0$ and, by KKT, $\alpha_i \geq 0$ & $r_i \geq 0$
  - $\alpha_i \in [0,C]$, i.e. limits outlier effect on $f(x)= \sum_{s \in SV} \alpha_s y_s k(x_s,x)+b$
  - Given r_i $(-\xi_i) \leq 0$ then $\xi_i=0 \Rightarrow r_i>0 \Rightarrow \alpha_i<C$ or $\xi_i>0 \Rightarrow r_i=0 \Rightarrow \alpha_i=C$
  1. if $\xi=0 \Rightarrow y_i(<w, \Phi(x_i)>+b) = 1$ **so on margin points have $\alpha_i<C$**
  2. If $\xi>0 \Rightarrow$ **points on the wrong side of their margin have $\alpha_i=C$**
- Also α_i $(1- \xi_i - y_i(<w, \Phi(x_i)>+b)) \leq 0$
  3. so given $\xi \geq 0 \Rightarrow y_i(<w, \Phi(x_i)>+b) > 1- \xi_i$ **so points on the correct side of the margin $\Rightarrow \alpha_i= 0$.**



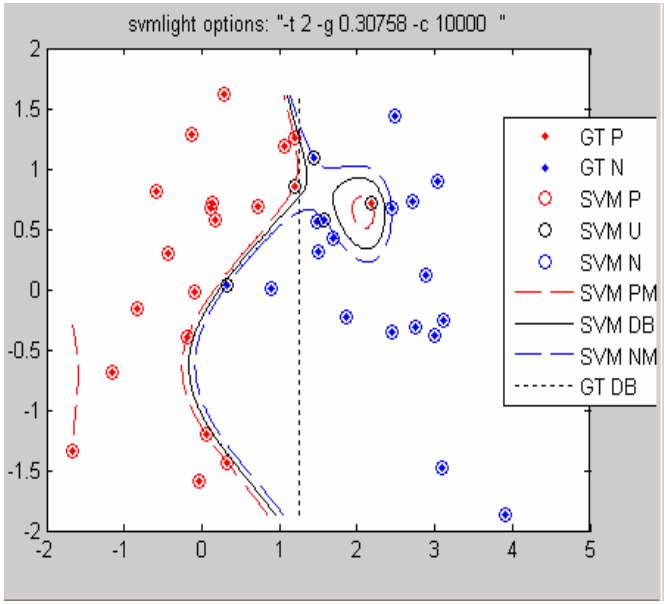
• Note that all "wrong" points (2) have equal weight regardless of whether they are "just off" their proper margin or far on the other side of the wrong margin $\Rightarrow$ **very insensitive to outliers**

Notes on setting C-SVC's "C"

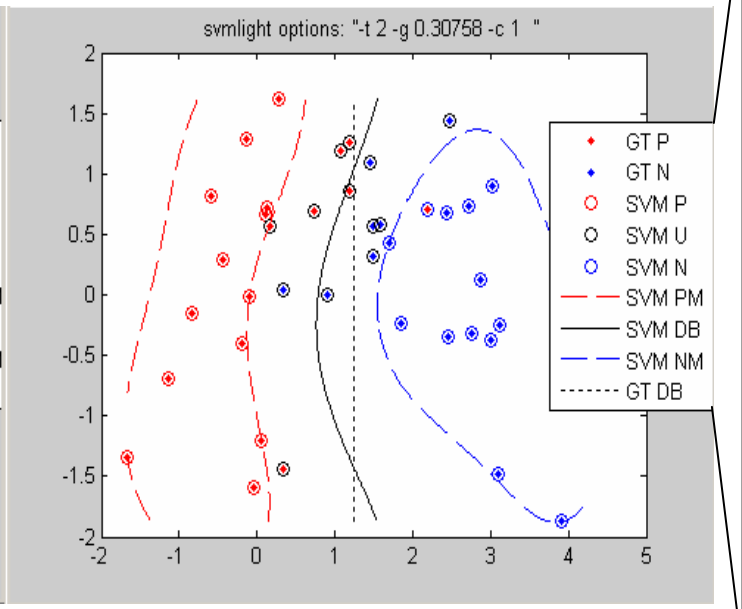
- Value of C can make a significant difference.
  - Lower C yields more generalization (wider margins).
  - In a typical regularization framework we have cost=$Risk_{emp}$ + $\lambda||w||^2$ and adjusting λ up penalizes weight magnitudes yielding increased generalization at the expense of empirical risk by adjusting the relative contribution of the two terms to the total cost. C-SVC's cost = $||w||^2$ + $C\Omega(\xi)$ where ξ is $Risk_{emp}$. Like Adjusting λ, adjusting C governs the relative importance of $Risk_{emp}$ and regularization but since it's a factor on the $Risk_{emp}$ lower values mean better generalization. In fact, λ=1/C.
- How to choose C
  - Typically, via cross validation in logarithm increments.
  - Hastie et. al. have an algorithm such that running SVM once (with one C) with some extra bookkeeping will allow the computation of α's for other values of C without re-optimization (the α' are piecewise linear in C).
- This is in addition to cross-validating over any kernel parameters

Example: 2-D Gausian Data Using SVMLight

- Data generated from two gaussian distributions with (μ_1,σ_1)=([0,0], 1) & (μ_2,σ_2)=([2.5,0], 1).
- 20 samples per class.
- "-t 2" $\Rightarrow$ Gaussian kernel
- "-g 0.30758" $\Rightarrow$ Gaussian kernel with 0.30758 = 1/ σ^2



Less regularization
"-c 10000" $\Rightarrow$ C = 10000
Notice the overfitting of the outliers and the small margin

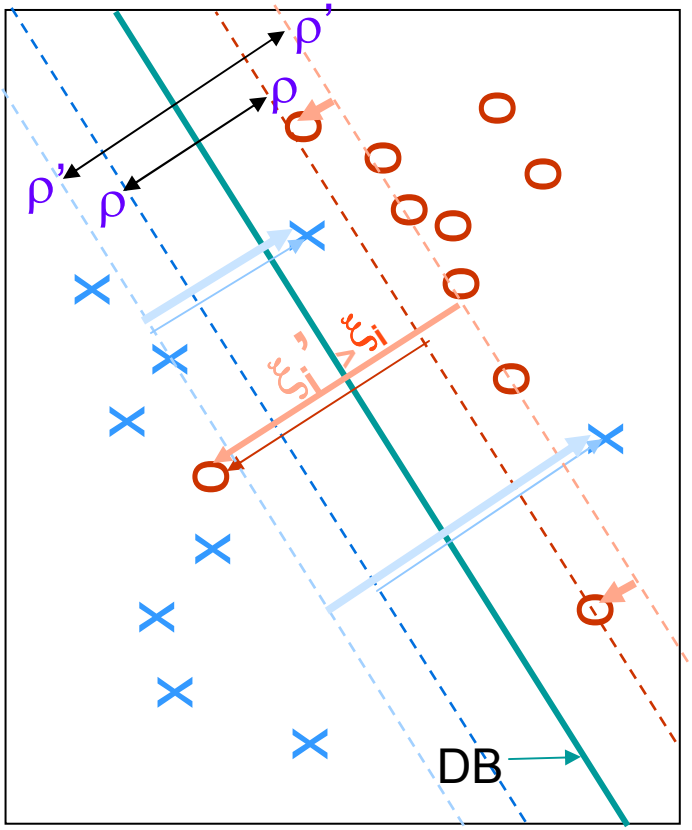


More regularization
"-c 1" $\Rightarrow$ C = 1

- •GT- Ground Truth
- •SVM P,U,N - The Positive, Undecided, and Negative class points as classified by the SVM
- • SVM PM, DB, NM – SVM's Positive class Margin , Decision Boundary, and Negative class Margin.
- • GT DB – The Ground Truth Decision Boundary

1-Norm Slack Variables (ν-SVC)

- A more intuitive easier to control reformulation than adjusting C.
- Move margins to meet a specified fractional classification error (wrong side of margin) goal, $\nu \in (0,1]$
- Margin becomes explicit optimization variable ρ.
- For a fixed DB, varying ρ changes the number of points between the margins
- A higher p $\Rightarrow$ a larger (i.e. better) margin but also $\Rightarrow$ more points in the margin and larger slack variables, ξ_i, which will work against an optimization of slack variables.
- $\Rightarrow$ need equilibrium for optimal ρ and ξ_i



As the margin (darker red/blue), specified, by ρ increases (to ρ' above, lighter red/blue), more points fall between the margins and any already there have increased ξ_i

1-Norm Slack Variables (ν-SVC)

- Primal Optimization:

$\text{Min}_{w,b,\xi,\rho}(\frac{1}{2}||w||^2 - \nu\rho + N^{-1}\xi_i))$ s.t. $y_i(<w,\Phi(x_i)>+b) \geq \rho - \xi_i$, $\xi_i \geq 0$, $\rho \geq 0$, $\forall_i$

  - N – # of points (so $N^{-1}\xi_i = E_{\xi i}$) *(some C-SVC's also use this vs $\Sigma\xi_i$)*
  - $-\nu\rho$ - maximizing ρ will maximize the margin and act as a dynamic normalization for w. Since we need to minimize this yields $-\rho$. ν is a positive constant so that doesn't change this and it yields useful properties (next slide).

- So primal lagrangian is

$L(w,b,\xi,\rho,\alpha,r,\delta) = \frac{1}{2}||w||^2 - \nu\rho + N^{-1}\xi_i + \sum_i\alpha_i(\rho - \xi_i - y_i(<w, \Phi(x_i)>+b)) - \sum_i r_i \xi_i - \delta\rho$

- For dual formulation must min w,b,ξ, ρ via $\{\nabla L_{\mathbf{w}}, \nabla L_{\mathbf{b}}, \nabla L_{\xi}, \nabla L_{\rho}\}=0$

$w^* = \Sigma_i\alpha_i y_i \Phi(x_i)$, $\Sigma_i\alpha_i y_i = 0$, $N^{-1} = \alpha_i + r_i$, $\sum_i\alpha_i - \delta = \nu$

- Substituting, etc., above into primal yields the dual optimization form

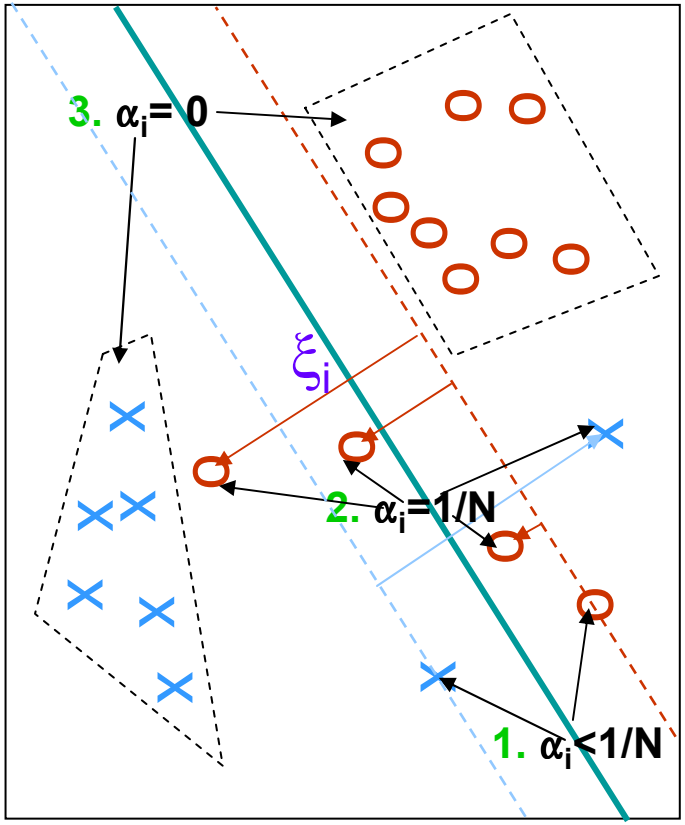
$$\mathbf{q(\alpha) = \max_{\alpha>0}(-\frac{1}{2}\Sigma_{ij}\alpha_i\alpha_j y_i y_j k(x_i,x_j))}$$

$$\mathbf{= \max_{\alpha>0}(-\frac{1}{2}v^t K v)}$$

- Same as hard margin without $\Sigma_i\alpha_i$ term

Analysis of 1-norm Slack Var SVM (ν-SVC)

- Take as a given that we must have a non-zero margin so $\rho > 0$
- Given the KKT $\Rightarrow$ ρ's lagrangian, $\delta = 0$.
- So $\sum_i \alpha_i - \delta = \nu \Rightarrow \sum_i \alpha_l = \nu$
- SV $\equiv y_i(\langle w, \Phi(x_i)\rangle + b) = \rho - \xi_i \Rightarrow \alpha_i > 0$
  - KKT $\Rightarrow$ Non SVs (3) have $\alpha_i = 0$
- Given the KKT & $N^{-1} = \alpha_i + r_i$
  - ν is the maximum fraction of margin error points $\xi_i > 0$ (margin errors) (2) $\Rightarrow r_i = 0 \Rightarrow N^{-1} = \alpha_i$ but since $\sum_i \alpha_i = \nu \Rightarrow$ for just margin errors we have $\sum_i N_i = \nu$
  - ν is the smallest fraction of SVs. For SVs we have $\alpha_i > 0$ and so given $\sum_i \alpha_i = \nu$, KKT
  - For margin points, $y_i(\langle w, \Phi(x_i)\rangle + b) = \rho$, as a SV $\alpha_i > 0$ and since $\xi_i = 0 \Rightarrow r_i > 0$ and so $\alpha_i < N^{-1}$ (1)



• As the 1-norm C-SVC, all the margin errors (aka outliers) have equal weight, here 1/N

Representer Theorem and Regularization
in english

- Given f being any (learning machine) function that can be created by a linear combination of any given kernel, k, evaluated using ALL the points in the input space, f evaluated on a training set yielding empirical risk $Risk_{emp}$, and Ω a monotonically increasing function on $[0, \infty)$, with the regularized risk, $Risk_{reg}(f)$, that regularizes $Risk_{emp}$ w.r.t. f, expressed as

$$Risk_{Reg}(f) = Risk_{emp} + \lambda\, \Omega(\, ||f||)$$

And the corresponding optimal f, f*, s.t.

$$f^* = \min_f[Risk_{Reg}(f)]$$

then f* is also a linear combination of k over the training set.

- In a nutshell, this just says that if you want to regularize your optimization cost function (to achieve greater generality), you can add the regularization term (as Ω constained above) and yet the optimal result, f*, can still be obtained with just the training set even though f isn't constrained to the training set