Machine Learning

4771

Instructor: Tony Jebara

Topic 5

- Generalization Guarantees
- VC-Dimension
- Nearest Neighbor Classification (infinite VC dimension)
- Structural Risk Minimization
- Support Vector Machines

Empirical Risk Minimization

- Example: non-pdf linear classifiers $f\left(x;\theta\right) = sign\left(\theta^T x + \theta_0\right) \in \left\{-1,1\right\}$
- Recall ERM: $R_{emp}\left(\theta\right) = \frac{1}{N}\sum_{i=1}^{N} L\left(y_i, f\left(x_i;\theta\right)\right) \quad \in \left[0,1\right]$
- Have loss function: quadratic: $L\left(y,x,\theta\right) = \frac{1}{2}\left(y - f\left(x;\theta\right)\right)^2$

  linear: $L\left(y,x,\theta\right) = \left|y - f\left(x;\theta\right)\right|$

  binary: $L\left(y,x,\theta\right) = step\left(-yf\left(x;\theta\right)\right)$
- Empirical $R_{emp}\left(\theta\right)$ *approximates* the true risk (expected error)

$$R\left(\theta\right) = E_P\left\{L\left(x,y,\theta\right)\right\} = \int_{X\times Y} P\left(x,y\right) L\left(x,y,\theta\right) dx\, dy \in \left[0,1\right]$$

- But, we don't know the true P(x,y)!
- If infinite data, *law of large numbers* says:

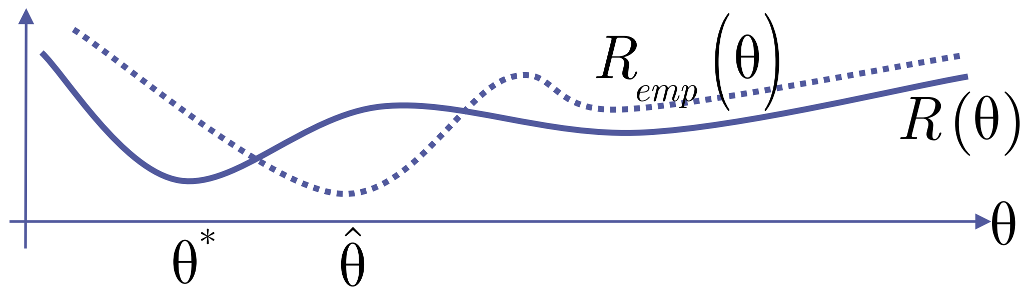
$$\lim_{N\to\infty}\ \min_\theta R_{emp}\left(\theta\right) = \min_\theta R\left(\theta\right)$$

- But, in general, can't make guarantees for ERM solution:

$$\arg\min_\theta R_{emp}\left(\theta\right) \neq \arg\min_\theta R\left(\theta\right)$$

Bounding the True Risk

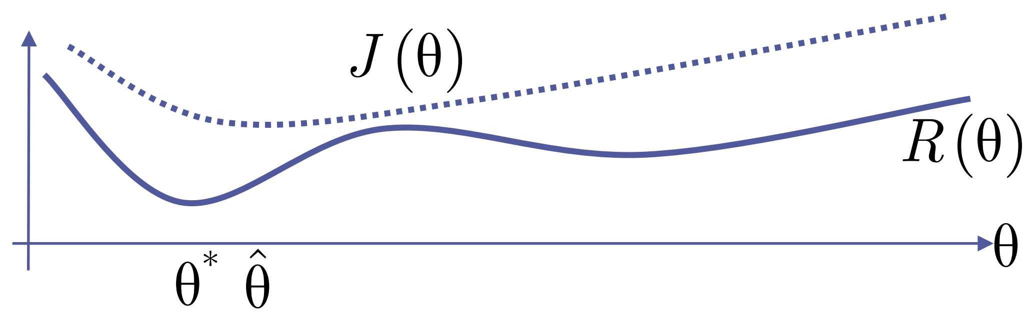
- ERM is inconsistent
  not guaranteed
  may do better on training than on test!



$$R(\hat{\theta}) \geq R_{emp}(\hat{\theta})$$

- Idea: add a prior or regularizer to $R_{emp}(\theta)$
- Define capacity or confidence = $C(\theta)$ which favors simpler θ

$$J(\theta) = R_{emp}(\theta) + C(\theta)$$



- If, $R(\theta) \leq J(\theta)$ we have bound $J(\theta)$ is a guaranteed risk
- After train, can guarantee future error rate is $\leq \min_\theta J(\theta)$

Bound the True Risk with VC

- But, how to find a guarantee? Difficult, but there is one…
- Theorem (Vapnik): with probability 1-η where η is a number between [0,1], the following bound holds:

$$R(\theta) \leq J(\theta) = R_{emp}(\theta) + \frac{2h\log\left(\frac{2eN}{h}\right) + 2\log\left(\frac{4}{\eta}\right)}{N}\left(1 + \sqrt{1 + \frac{NR_{emp}(\theta)}{h\log\left(\frac{2eN}{h}\right) + \log\left(\frac{4}{\eta}\right)}}\right)$$

N = number of data points

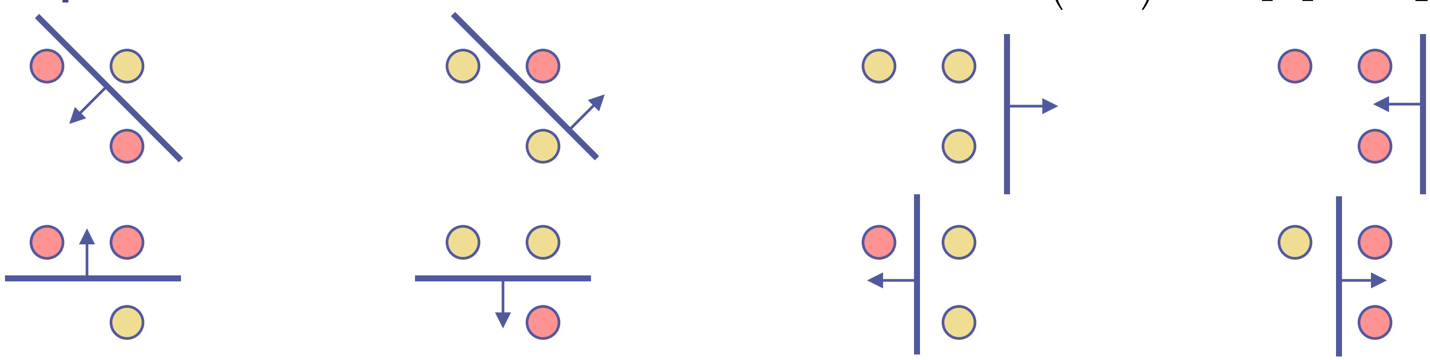
h = Vapnik-Chervonenkis (VC) dimension (1970's)

= capacity of the classifier class $f(.;\theta)$

- Note, above is independent of the true P(x,y)
- A *worst-case scenario* bound, guaranteed for all P(x,y)
- VC dimension not just the # of parameters a classifier has
- VC measures # of different datasets it can classify perfectly
- Structural Risk Minimization: minimize risk bound J(θ)

VC Dimension & Shattering

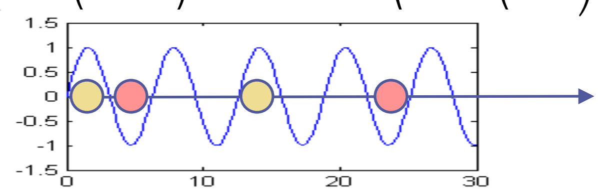
- How to compute h or VC for a family of functions $f\left(.;\theta\right)$
  h = # of training points that can be shattered
- Recall, classifier maps input to output $f\left(x;\theta\right) \rightarrow y \in \left\{-1,1\right\}$
- Shattering: I pick h points & place them at $x_1,\ldots,x_h$
  You challenge me with 2^h possible labelings $y_1,\ldots,y_h \in \left\{\pm 1\right\}^h$
  VC dimension is maximum # of points I can place which a $f\left(x;\theta\right)$ can correctly classify for arbitrary labeling $y_1,\ldots,y_h$
- Example: for 2d linear classifier h=3 $f\left(x;\theta\right) = x_1\theta_1 + x_2\theta_2 + \theta_0$



can't ever shatter 4 points! or 3 points on a straight line…

VC Dimension & Shattering

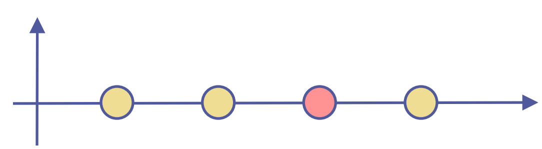
- More generally for higher dimensional linear classifiers, a hyperplane in $\mathbb{R}^d$ shatters any set of linearly independent points. Can choose d+1 linearly indep. points so h=d+1
- Note: VC is *not necessarily proportional* to # of parameters
- Example: sinusoidal 1d classifier $f(x;\theta) = sign(\sin(\theta x))$



number of parameters=1
…but… h=infinity!

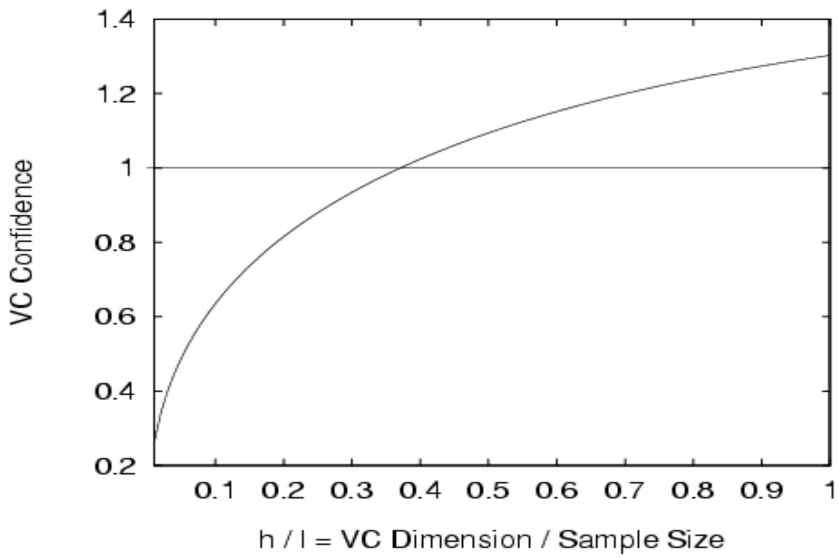
since I can choose: $x_i = 10^{-i} \qquad i = 1,\ldots,h$
no matter what labeling you challenge: $y_1,\ldots,y_h \in \{\pm 1\}^h$
using $\theta = \pi\left(1 + \sum_{i=1}^{h} \frac{1}{2}(1 - y_i)10^{-i}\right)$ shatters perfectly

But, as a side note, if I choose 4 equally spaced x's then cannot shatter



VC Dimension & Shattering

- Recall that VC dimension gives an upper bound
- We want to minimize h since that minimizes $C(\theta)$ & $J(\theta)$
- If can't compute h exactly but can compute h^+ can plug in h^+ in bound & still guarantee
- Also, sometimes bound is trivial
- Need h/N = 0.3 before $C(\theta)<1$ (recall $R(\theta)$ in [0,1])

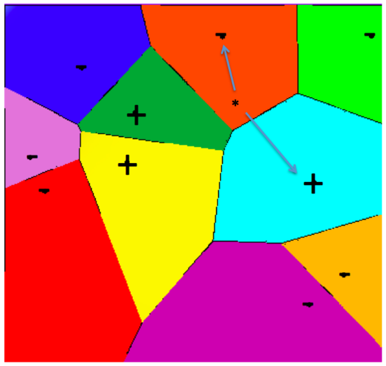


- Note: $h = low \Rightarrow good\ performance$ $\quad h = \infty \not\Rightarrow poor\ performance$

Nearest Neighbors & VC

- Consider Nearest Neighbors classification algorithm:

  Input a query example x
  Find training example x_i in $\{x_1,\ldots x_N\}$ closest to x
  Predict label for x as y_i of neighbor



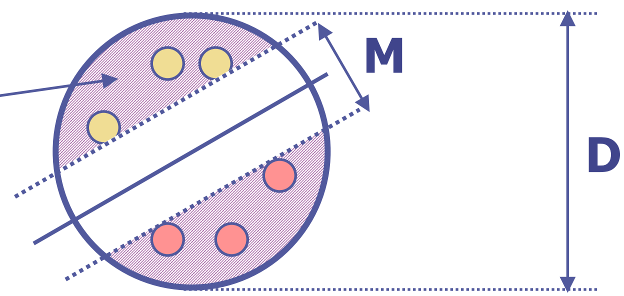
- Often use Euclidean distance $\left\| x - x_i \right\|$ to measure closeness
- Nearest neighbors shatters any set of points!
- So VC=infinity, C(θ)=infinity, guaranteed risk=infinity
- But still works well in practice

$$h = \infty \not\Rightarrow poor\, performance \quad h = low \Rightarrow good\, performance$$

VC Dimension & Large Margins

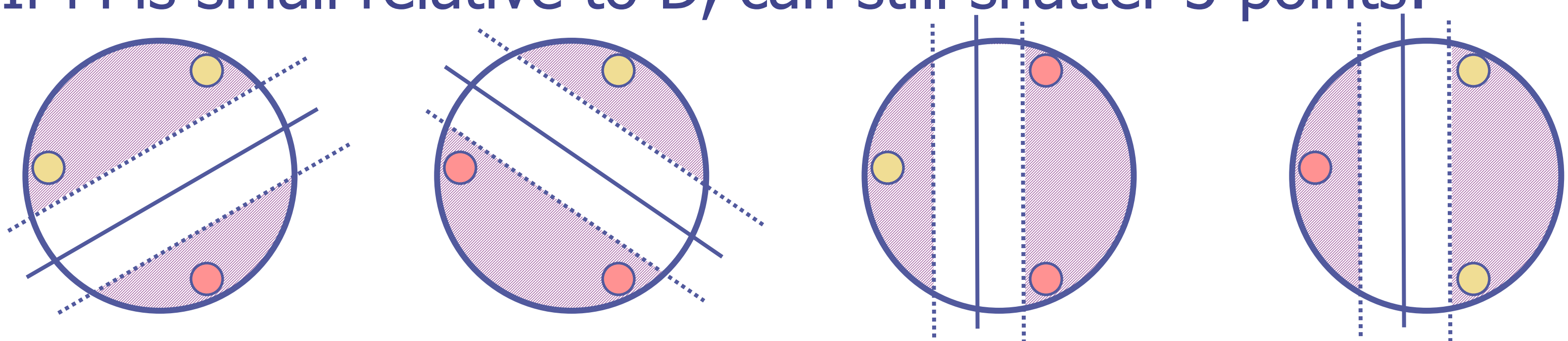
- Linear classifiers are too big a function class since h=d+1
- Can reduce VC dimension if we restrict them
- Constrain linear classifiers to data living inside a sphere
- Gap-Tolerant classifiers: a linear classifier whose activity is constrained to a sphere & outside a margin

Only count errors in shaded region Elsewhere have $L(x,y,\theta)=0$



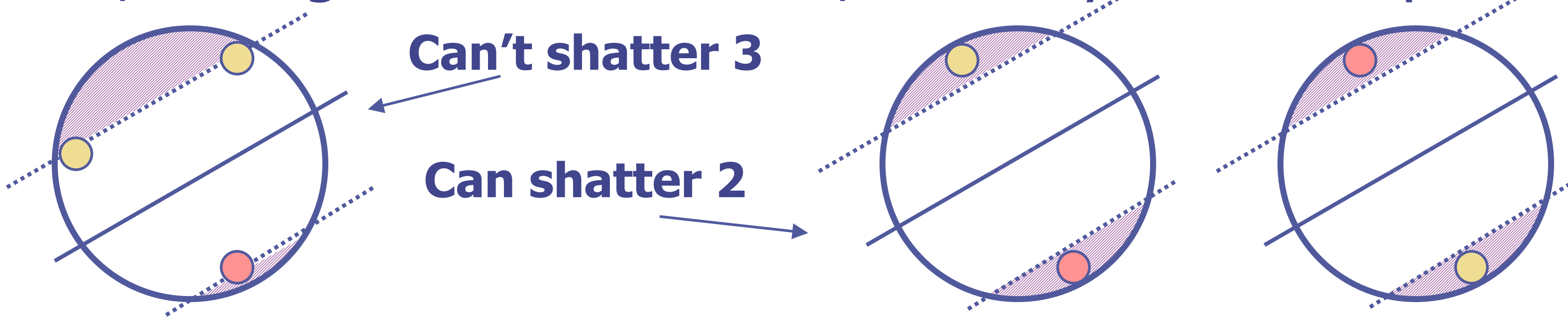
M=margin
D=diameter
d=dimensionality

- If M is small relative to D, can still shatter 3 points:



VC Dimension & Large Margins

- But, as M *grows* relative to D, can only shatter 2 points!



- For hyperplanes, as M grows vs. D, shatter fewer points!
- VC dimension h goes down if gap-tolerant classifier has larger margin, general formula is: $h \leq \min\left\{ceil\left[\frac{D^2}{M^2}\right], d\right\} + 1$
- Before, just had h=d+1. Now we have a smaller h
- If data is anywhere, D is infinite and back to h=d+1
- Typically real data is bounded (by sphere), D is fixed
- Maximizing M reduces h, improving guaranteed risk $J(\theta)$
- Note: $R(\theta)$ doesn't count errors in margin or outside sphere

Structural Risk Minimization

- Structural Risk Minimization: minimize risk bound $J(\theta)$ reducing empirical error & reduce VC dimension h

$$R(\theta) \leq J(\theta) = R_{emp}(\theta) + \frac{2h\log\left(\frac{2eN}{h}\right) + 2\log\left(\frac{4}{\eta}\right)}{N}\left(1 + \sqrt{1 + \frac{NR_{emp}(\theta)}{h\log\left(\frac{2eN}{h}\right) + \log\left(\frac{4}{\eta}\right)}}\right)$$

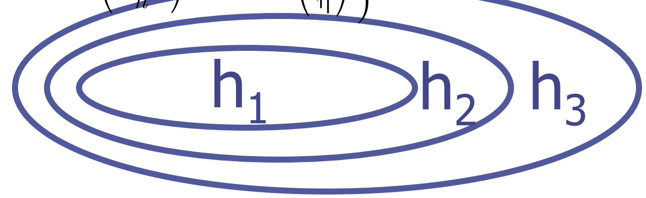
for each model i in list of hypothesis

1) compute its $h = h_i$

2) $\theta^* = \arg\min_\theta R_{emp}(\theta)$

3) compute $J(\theta^*, h_i)$

choose model with lowest $J(\theta^*, h_i)$



Space of different Classifiers or Hypotheses

- Or, directly optimize over both $(\theta^*, h) = \arg\min_{\theta,h} J(\theta, h)$
- If possible, min empirical error while also minimizing VC
- For gap-tolerant linear classifiers, minimize $R_{emp}(\theta)$ *while* maximizing margin, support vector machines do just that!

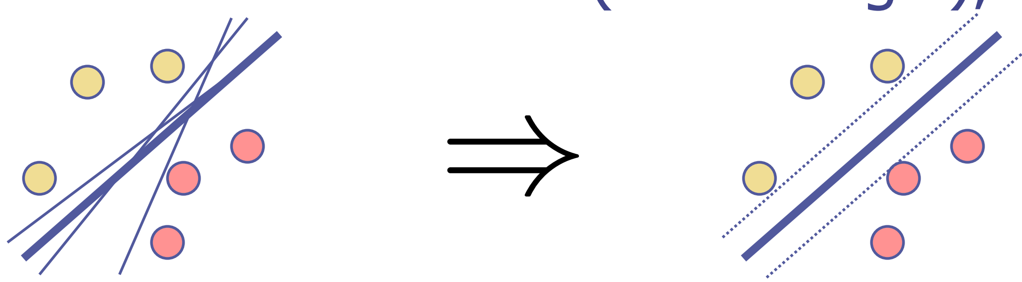
Support Vector Machines

- Support vector machines are (in the simplest case) linear classifiers that do structural risk minimization (SRM)
- Directly maximize margin to reduce guaranteed risk $J(\theta)$
- Assume first the 2-class data is linearly separable:

$$have \ \left\{\left(x_1,y_1\right),\ldots,\left(x_N,y_N\right)\right\} \ \ where \ x_i \in \mathbb{R}^D \ and \ y_i \in \left\{-1,1\right\}$$

$$f\left(x;\theta\right) = sign\left(w^T x + b\right)$$

- Decision boundary or hyperplane given by $w^T x + b = 0$
- Note: can scale w & b while keeping same boundary
- Many solutions exist which have empirical error $R_{emp}(\theta)=0$
- Want widest or thickest one (max margin), also it's unique!



Side Note: Constraints

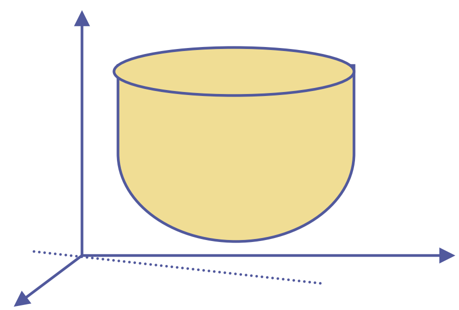
- How to minimize a function subject to equality constraints?

$$\min_{x_1,x_2} f\left(\vec{x}\right) = \min_{x_1,x_2} b_1 x_1 + b_2 x_2 + \tfrac{1}{2} H_{11} x_1^2 + H_{12} x_1 x_2 + \tfrac{1}{2} H_{22} x_2^2$$

$$= \min_{\vec{x}} \vec{b}^T \vec{x} + \tfrac{1}{2} \vec{x}^T H \vec{x}$$

$$\Rightarrow \frac{\partial f}{\partial \vec{x}} = \vec{b} + H\vec{x} = 0$$

$$\Rightarrow \vec{x} = -H^{-1} b$$

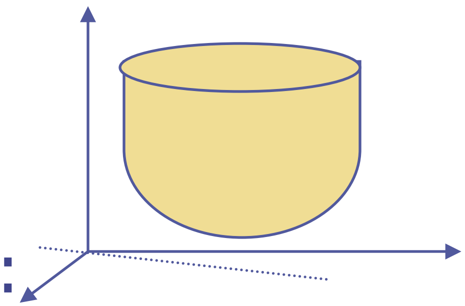


- Only walk on $x_1=2x_2$ or… $x_1-2x_2=0$…
- Use Lagrange Multipliers, for each constraint, subtract it times a lambda variable. Lambda blows up the minimization if we don't satisfy the constraint:

$$\min_{x_1,x_2} \max_{\lambda} f\left(\vec{x}\right) - \lambda\left(equality\, condition = 0\right)$$

$$= \min_{x_1,x_2} \max_{\lambda} b_1 x_1 + b_2 x_2 + \tfrac{1}{2} H_{11} x_1^2 + H_{12} x_1 x_2 + \tfrac{1}{2} H_{22} x_2^2 - \lambda\left(x_1 - 2x_2\right)$$

Side Note: Constraints



- Cost minimization with equality constraints:
  1) Subtract each constraint times an extra variable
     (a Lagrange multiplier λ, like an adversary variable)
  2) Take partials with respect to x and set to zero
  3) Plug solution into constraint to find lambda

$$x_1 - 2x_2 = \vec{x}^T \begin{bmatrix} 1 \\ -2 \end{bmatrix}$$

$$\min_{\vec{x}} \max_{\lambda} f(\vec{x}) - \lambda\left(equality\,condition = 0\right)$$

$$= \min_{\vec{x}} \max_{\lambda} b^T\vec{x} + \tfrac{1}{2}\vec{x}^T H\vec{x} - \lambda\left(x_1 - 2x_2\right)$$

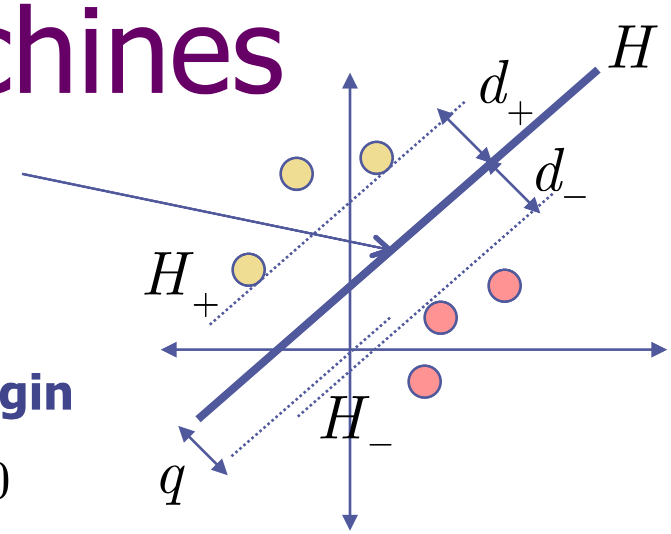
$$\Rightarrow \frac{\partial f}{\partial \vec{x}} = \vec{b} + H\vec{x} - \lambda\begin{bmatrix} 1 \\ -2 \end{bmatrix} = 0 \quad \Rightarrow \vec{x} = H^{-1}\lambda\begin{bmatrix} 1 \\ -2 \end{bmatrix} - H^{-1}b$$

$$\Rightarrow \left(H^{-1}\lambda\begin{bmatrix} 1 \\ -2 \end{bmatrix} - H^{-1}b\right)^T\begin{bmatrix} 1 \\ -2 \end{bmatrix} = 0 \Rightarrow \lambda = \frac{b^T H^{-1}\begin{bmatrix} 1 \\ -2 \end{bmatrix}}{\begin{bmatrix} 1 \\ -2 \end{bmatrix}^T H^{-1}\begin{bmatrix} 1 \\ -2 \end{bmatrix}}$$

Support Vector Machines

- Define:

$$w^T x + b = 0$$



H_+=positive margin hyperplane
H_- =negative margin hyperplane
q =distance from decision plane to origin

$$q = \min_x \left\| \vec{x} - \vec{0} \right\| \quad subject\, to \quad w^T x + b = 0$$

$$\min_x \tfrac{1}{2} \left\| \vec{x} - \vec{0} \right\|^2 - \lambda \left(w^T x + b \right)$$

1) grad $\frac{\partial}{\partial x} \left(\frac{1}{2} x^T x - \lambda \left(w^T x + b \right) \right) = 0$

$$x - \lambda w = 0$$

$$x = \lambda w$$

2) plug into constraint

$$w^T x + b = 0$$

$$w^T \left(\lambda w \right) + b = 0$$

$$\lambda = -\frac{b}{w^T w}$$

3) Sol'n $\hat{x} = -\left(\frac{b}{w^T w} \right) w$

4) distance $q = \left\| \hat{x} - \vec{0} \right\| = \left\| -\frac{b}{w^T w} w \right\| = \frac{|b|}{w^T w} \sqrt{w^T w} = \frac{|b|}{\|w\|}$

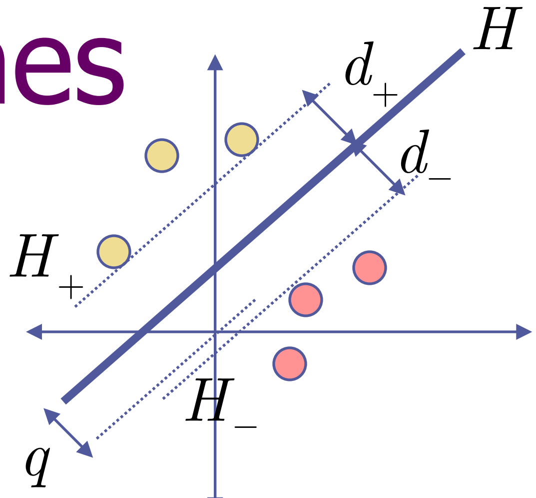
5) Define without loss of generality since can scale b & w

$$H \rightarrow w^T x + b = 0$$

$$H_+ \rightarrow w^T x + b = +1$$

$$H_- \rightarrow w^T x + b = -1$$

Support Vector Machines



$$H_+ \rightarrow w^T x + b = +1$$
$$H_- \rightarrow w^T x + b = -1$$

- The constraints on the SVM for $R_{emp}(\theta)=0$ are thus:

$$w^T x_i + b \geq +1 \quad \forall y_i = +1$$
$$w^T x_i + b \leq -1 \quad \forall y_i = -1$$

- Or more simply: $y_i\left(w^T x_i + b\right) - 1 \geq 0$
- The margin of the SVM is:

$$m = d_+ + d_-$$

- Distance to origin: $H \rightarrow q = \frac{|b|}{\|w\|} \quad H_+ \rightarrow q_+ = \frac{|b-1|}{\|w\|} \quad H_- \rightarrow q_- = \frac{|-1-b|}{\|w\|}$
- Therefore: $d_+ = d_- = \frac{1}{\|w\|}$ and margin $m = \frac{2}{\|w\|}$
- Want to max margin, or equivalently minimize: $\|w\|$ *or* $\|w\|^2$
- SVM Problem: $\min \frac{1}{2}\|w\|^2 \quad subject\,to \quad y_i\left(w^T x_i + b\right) - 1 \geq 0$
- This is a quadratic program!
- Can plug this into a matlab function called "qp()", done!

Side Note: Optimization Tools

• A hierarchy of Matlab optimization packages to use:

Linear Programming $\min_{\vec{x}} \vec{b}^T \vec{x} \;\; s.t. \; \vec{c}_i^T \vec{x} \geq \alpha_i \;\; \forall i$

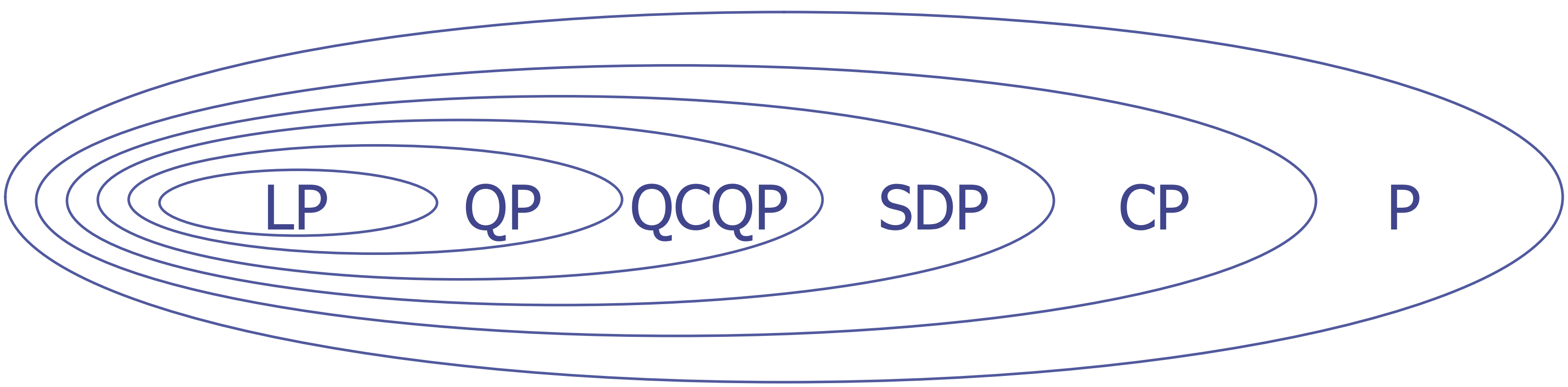
<Quadratic Programming $\min_{\vec{x}} \frac{1}{2} \vec{x}^T H \vec{x} + \vec{b}^T \vec{x} \;\; s.t. \; \vec{c}_i^T \vec{x} \geq \alpha_i \;\; \forall i$

<Quadratically Constrained Quadratic Programming

<Semidefinite Programming

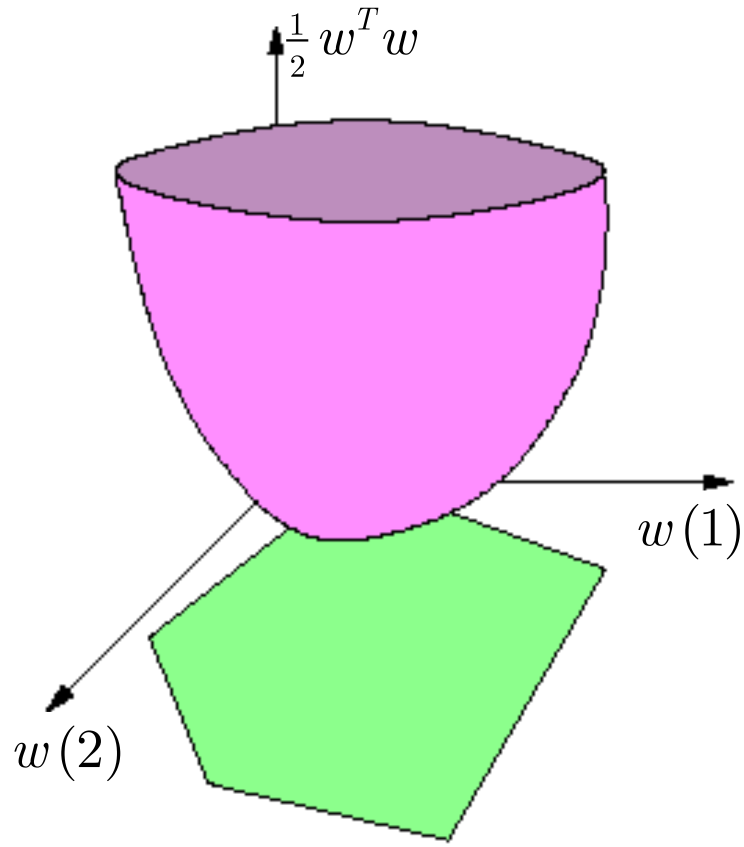
<Convex Programming

<Polynomial Time Algorithms



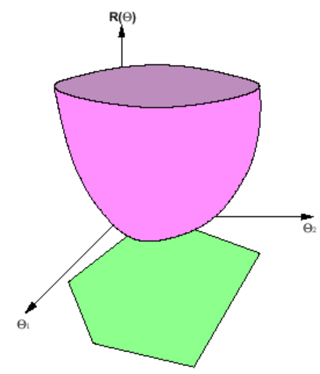
Side Note: Optimization Tools

- Each data point adds $y_i\left(w^T x_i + b\right) - 1 \geq 0$ linear inequality to QP
- Each point cuts a half plane of allowable SVMs and reduces green region
- The SVM is closest point to the origin that is still in the green region
- The preceptron algorithm just puts us randomly in green region
- QP runs in cubic polynomial time
- There are D values in the w vector
- Needs $O(D^3)$ run time… But, there is a DUAL SVM in $O(N^3)$!



SVM in Dual Form

- We can also solve the problem via convex duality
- Primal SVM problem L_P: $\min \frac{1}{2}\|w\|^2 \quad \textit{subject to} \quad y_i\left(w^T x_i + b\right) - 1 \geq 0$
- This is a quadratic program, quadratic cost function with multiple linear inequalities (these carve out a convex hull)

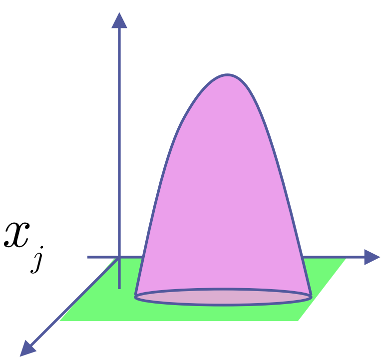


- Subtract from cost each inequality times an α Lagrange multiplier, take derivatives of w & b:

$$L_P = \min_{w,b} \max_{\alpha \geq 0} \frac{1}{2}\|w\|^2 - \sum_i \alpha_i \left(y_i\left(w^T x_i + b\right) - 1\right)$$

$$\frac{\partial}{\partial w} L_P = w - \sum_i \alpha_i y_i x_i = 0 \rightarrow w = \sum_i \alpha_i y_i x_i$$

$$\frac{\partial}{\partial b} L_P = -\sum_i \alpha_i y_i = 0$$



- Plug back in, dual: $L_D = \sum_i \alpha_i - \frac{1}{2}\sum_i \sum_j \alpha_i \alpha_j y_i y_j x_i^T x_j$
- Also have constraints: $\sum_i \alpha_i y_i = 0 \;\; \& \;\; \alpha_i \geq 0$
- Above L_D must be maximized! convex duality... also qp()