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 $f(x; \theta) = \text{sign}(\theta^T x + \theta_0) \in \{-1, 1\}$
- Recall ERM: 
  $$R_{emp}(\theta) = \frac{1}{N} \sum_{i=1}^{N} L(y_i, f(x_i; \theta)) \in [0, 1]$$
- Have loss function: quadratic: 
  $$L(y, x, \theta) = \frac{1}{2} (y - f(x; \theta))^2$$
  linear: 
  $$L(y, x, \theta) = |y - f(x; \theta)|$$
  binary: 
  $$L(y, x, \theta) = \text{step}(-yf(x; \theta))$$
- Empirical $R_{emp}(\theta)$ approximates the true risk (expected error) 
  $$R(\theta) = E_P \left\{ L(x, y, \theta) \right\} = \int_{X \times Y} P(x, y) L(x, y, \theta) \, dx \, dy \in [0, 1]$$
- But, we don’t know the true $P(x, y)$!
- If infinite data, law of large numbers says: 
  $$\lim_{N \to \infty} \min_{\theta} R_{emp}(\theta) = \min_{\theta} R(\theta)$$
- But, in general, can’t make guarantees for ERM solution: 
  $$\arg \min_{\theta} R_{emp}(\theta) \neq \arg \min_{\theta} R(\theta)$$
Bounding the True Risk

- ERM is inconsistent, not guaranteed, may do better on training than on test!
- Idea: add a prior or regularizer to $R_{emp}(\theta)$
- Define capacity or confidence $C(\theta)$ which favors simpler $\theta$

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

- 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 $1 - \eta$ where $\eta$ is a number between $[0,1]$, the following bound holds:

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

$N$ = 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 $J(\theta)$
VC Dimension & Shattering

• How to compute $h$ or VC for a family of functions $f(.; \theta)$
  
  $h = \#$ of training points that can be shattered

• Recall, classifier maps input to output  
  $f(x; \theta) \rightarrow y \in \{-1, 1\}$

• Shattering: I pick $h$ points & place them at
  
  You challenge me with $2^h$ possible labelings  
  $x_1, \ldots, x_h, y_1, \ldots, y_h \in \{\pm 1\}^h$
  
  VC dimension is maximum \# of points I can place which
  a $f(x; \theta)$ can correctly classify for arbitrary labeling $y_1, \ldots, y_h$

• Example: for 2d linear classifier $h=3$
  
  $f(x; \theta) = x_1 \theta_1 + x_2 \theta_2 + \theta_0$

  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

  number of parameters=1

  ...but... $h=\infty$!

  since I can choose: $x_i = 10^{-i}$  
  no matter what labeling you challenge: $y_1, \ldots, y_h \in \{\pm 1\}^h$

  using $\theta = \pi \left(1 + \sum_{i=1}^{h} \frac{1}{2} \left(1 - y_i\right) 10^{-i} \right)$  
  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 $\quad h = \infty \Rightarrow$ 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 $\|x - x_i\|$ to measure closeness
• Nearest neighbors shatters any set of points!  
• So VC=infinity, $C(\theta)=\infty$, guaranteed risk=infinity  
• But still works well in practice

  $h = \infty \nRightarrow poor \ performance \quad h = low \Rightarrow good \ performance$
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 \( L(x,y,\theta) = 0 \)

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! 
  
  ![Can’t shatter 3, Can shatter 2](image)

• 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:

$$ h \leq \min \left\{ \text{ceil} \left( \frac{D^2}{M^2} \right), d \right\} + 1 $$

• 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 $J(\theta^*, h_i)$

- 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:
  
  \[ \text{have } \{ (x_1, y_1), \ldots, (x_N, y_N) \} \text{ where } x_i \in \mathbb{R}^D \text{ and } y_i \in \{-1, 1\} \]

  \[ f(x; \theta) = \text{sign}(w^T x + b) \]

• 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 $R_{\text{emp}}(\theta) = 0$
• Want widest or thickest one (max margin), also it’s unique!
Side Note: Constraints

• How to minimize a function subject to equality constraints?

\[
\begin{align*}
\min_{x_1, x_2} f(\vec{x}) &= \min_{x_1, x_2} b_1 x_1 + b_2 x_2 + \frac{1}{2} H_{11} x_1^2 + H_{12} x_1 x_2 + \frac{1}{2} H_{22} x_2^2 \\
&= \min_{\vec{x}} \vec{b}^T \vec{x} + \frac{1}{2} \vec{x}^T H \vec{x} \\
\Rightarrow \frac{\partial f}{\partial \vec{x}} &= \vec{b} + H \vec{x} = 0 \\
\Rightarrow \vec{x} &= -H^{-1} \vec{b}
\end{align*}
\]

• 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:

\[
\begin{align*}
\min_{x_1, x_2} \max_{\lambda} f(\vec{x}) - \lambda &\big(equality\ condition = 0\big) \\
= \min_{x_1, x_2} \max_{\lambda} b_1 x_1 + b_2 x_2 + \frac{1}{2} H_{11} x_1^2 + H_{12} x_1 x_2 + \frac{1}{2} H_{22} x_2^2 - \lambda \big(x_1 - 2x_2\big)
\end{align*}
\]
Side Note: Constraints

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

$$\begin{align*}
\min_{\bar{x}} \max_{\lambda} f(\bar{x}) - \lambda \left( \text{equality condition} = 0 \right) \\
= \min_{\bar{x}} \max_{\lambda} b^T \bar{x} + \frac{1}{2} \bar{x}^T H \bar{x} - \lambda \left( x_1 - 2x_2 \right) \\
\Rightarrow \frac{\partial f}{\partial \bar{x}} = \bar{b} + H \bar{x} - \lambda \left[ \begin{array}{c} 1 \\ -2 \end{array} \right] = 0 \Rightarrow \bar{x} = H^{-1} \lambda \left[ \begin{array}{c} 1 \\ -2 \end{array} \right] - H^{-1} b \\
\Rightarrow \left( H^{-1} \lambda \left[ \begin{array}{c} 1 \\ -2 \end{array} \right] - H^{-1} b \right)^T \left[ \begin{array}{c} 1 \\ -2 \end{array} \right] = 0 \Rightarrow \lambda = \frac{b^T H^{-1} \left[ \begin{array}{c} 1 \\ -2 \end{array} \right] \left[ \begin{array}{c} 1 \\ -2 \end{array} \right]^T}{\left[ \begin{array}{c} 1 \\ -2 \end{array} \right]^T H^{-1} \left[ \begin{array}{c} 1 \\ -2 \end{array} \right]} \\
\end{align*}$$

$$x_1 - 2x_2 = \bar{x}^T \left[ \begin{array}{c} 1 \\ -2 \end{array} \right]$$
Support Vector Machines

- Define:
  \[ w^T x + b = 0 \]
  
  \[ H_+ = \text{positive margin hyperplane} \]
  
  \[ H_- = \text{negative margin hyperplane} \]
  
  \[ q = \text{distance from decision plane to origin} \]

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

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

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

\[ x - \lambda w = 0 \]

2) Plug into constraint

\[ w^T x + b = 0 \]

\[ w^T (\lambda w) + b = 0 \]

\[ \lambda = -\frac{b}{w^T w} \]

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

- The constraints on the SVM for $R_{\text{emp}}(\theta) = 0$ 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: $y_i(w^T x_i + b) - 1 \geq 0$

- The margin of the SVM is:
  \[
  m = d_+ + d_-
  \]

- Distance to origin: $H \rightarrow q = \frac{|b|}{\|w\|}$

- 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:
  \[
  \min \frac{1}{2}\|w\|^2 \quad \text{subject to} \quad y_i(w^T x_i + b) - 1 \geq 0
  \]

- 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  \( \min_{\vec{x}} \vec{b}^T \vec{x} \quad \text{s.t.} \quad \vec{c}_i^T \vec{x} \geq \alpha_i \quad \forall i \)

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

< Quadratically Constrained Quadratic Programming

< Semidefinite Programming

< Convex Programming

< Polynomial Time Algorithms
Side Note: Optimization Tools

• Each data point adds $y_i(w^T x_i + b) - 1 \geq 0$ 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 \( L_P \):
  \[
  \min \frac{1}{2} \|w\|^2 \quad \text{subject to} \quad y_i \left( w^T x_i + b \right) - 1 \geq 0
  \]

• 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 \( \alpha \) 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:
  \[
  \sum_i \alpha_i y_i = 0 \quad \text{&} \quad \alpha_i \geq 0
  \]

• Above \( L_D \) must be maximized! convex duality... also \( \text{qp()} \)