Linear Separability: More formally, Using Margin

- Data linearly separable, if there exists
  - a vector $\mathbf{w}$, $||\mathbf{w}|| = 1$
  - a margin

  Such that
  - for all points
    - $y^{(i)} = +1$ if $\mathbf{w}^T \mathbf{x}^{(i)} > \gamma$
    - $y^{(i)} = -1$ if $\mathbf{w}^T \mathbf{x}^{(i)} < -\gamma$

  and $\gamma$ is the margin.
Perceptron Analysis: Linearly Separable Case

- Theorem [Block, Novikoff]:
  - Given a sequence of labeled examples: \((x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}) \ldots, (x^{(N)}, y^{(N)})\)
  - Each feature vector has bounded norm: 
  - If dataset is linearly separable:

\[
\exists \mathbf{w}, \| \mathbf{w} \| = 1, \quad \forall i, y^{(i)} \mathbf{w} \cdot x^{(i)} \geq 0, \text{ for } i \leq N
\]

- Then the number of mistakes made by the online perceptron on this sequence is bounded by

\[
\left( \frac{2}{\delta} \right)^2 \approx \text{constant, does not depend on } N
\]

Beyond Linearly Separable Case

- Perceptron algorithm is super cool!
  - No assumption about data distribution!
    - Could be generated by an oblivious adversary, no need to be iid
  - Makes a fixed number of mistakes, and it’s done for ever!
    - Even if you see infinite data

- However, real world not linearly separable
  - Can’t expect never to make mistakes again
  - Analysis extends to non-linearly separable case
  - Very similar bound, see Freund & Schapire
  - Converges, but ultimately may not give good accuracy (make many many mistakes)
What if the data is not linearly separable?

Use features of features of features of features...

\[ \Phi(x) : \mathbb{R}^m \rightarrow F \]

Feature space can get really large really quickly!

Higher order polynomials

\[ \text{num. terms} = \binom{d + m - 1}{d} = \frac{(d + m - 1)!}{d!(m - 1)!} \]

Even though dimensions of \( \phi(x) \) are huge, fit model very well very quickly

d = 6, m = 100

about 1.6 billion terms
Perceptron Revisited

- Given weight vector $w^{(t)}$, predict point $x$ by:

  - Mistake at time $t$: $w^{(t+1)} = w^{(t)} + y^{(t)} x^{(t)}$
  
  - Thus, write weight vector in terms of mistaken data points only:
    - Let $M^{(t)}$ be time steps up to $t$ when mistakes were made:

- Prediction rule now:

- When using high dimensional features:

Dot-product of polynomials

$\Phi(u) \cdot \Phi(v) = \text{polynomials of degree exactly } d$
Finally the Kernel Trick!!!
(From Kernelized Perceptron)

- Every time you make a mistake, remember $(x^{(t)}, y^{(t)})$

- Kernelized Perceptron prediction for $x$:

$$\text{sign}(w^{(t)} \cdot \phi(x)) = \sum_{j \in M^{(t)}} y^{(j)} \phi(x^{(j)}) \cdot \phi(x)$$

$$= \sum_{j \in M^{(t)}} y^{(j)} \phi(x^{(j)}) \cdot \phi(x)$$

Polynomial kernels

- All monomials of degree $d$ in $O(d)$ operations:

$$\Phi(u) \cdot \Phi(v) = (u \cdot v)^d = \text{polynomials of degree exactly } d$$

- How about all monomials of degree up to $d$?
  - Solution 0:
  - Better solution:
Common kernels

- Polynomials of degree exactly $d$
  \[ K(u, v) = (u \cdot v)^d \]
- Polynomials of degree up to $d$
  \[ K(u, v) = (u \cdot v + 1)^d \]
- Gaussian (squared exponential) kernel
  \[ K(u, v) = \exp\left(-\frac{||u - v||}{2\sigma^2}\right) \]
- Sigmoid
  \[ K(u, v) = \tanh(\eta u \cdot v + \nu) \]

What you need to know

- Notion of online learning
- Perceptron algorithm
- Mistake bounds and proofs
- The kernel trick
- Kernelized Perceptron
- Derive polynomial kernel
- Common kernels
- In online learning, report averaged weights at the end
Linear classifiers – Which line is better?
Pick the one with the largest margin!

“confidence” = \( y^j(w \cdot x^j + w_0) \)

Maximize the margin

\[
\max_{\gamma, w, w_0} \gamma \quad \text{s.t.} \quad y^j(w \cdot x^j + w_0) \geq \gamma, \forall j \in \{1, \ldots, N\}
\]
But there are many planes...

Review: Normal to a plane

$$x_j = \bar{x}_j + \lambda \frac{w}{||w||}$$
A Convention: Normalized margin –
Canonical hyperplanes

\[ x_j = \bar{x}_j + \lambda \frac{w}{||w||} \]

Margin maximization using canonical hyperplanes

Unnormalized problem:
\[ \max_{\gamma,w,0} \gamma \]
\[ y_j(w \cdot x_j + w_0) \geq \gamma, \forall j \in \{1, \ldots, N\} \]

Normalized Problem:
\[ \min_{w,0} \frac{||w||^2}{2} \]
\[ y_j(w \cdot x_j + w_0) \geq 1, \forall j \in \{1, \ldots, N\} \]
Support vector machines (SVMs)

$$\min_{w, w_0} \|w\|_2^2$$

$$y_j (w \cdot x_j + w_0) \geq 1, \forall j \in \{1, \ldots, N\}$$

- Solve efficiently by many methods, e.g.,
  - quadratic programming (QP)
  - Well-studied solution algorithms
  - Stochastic gradient descent
- Hyperplane defined by support vectors

What if the data is not linearly separable?

Use features of features of features of features....
What if the data is still not linearly separable?

\[ \min_{w, w_0} \|w\|_2^2 \]
\[ y_j^i (w \cdot x^j + w_0) \geq 1, \forall j \]

- If data is not linearly separable, some points don’t satisfy margin constraint:

- How bad is the violation?

- Tradeoff margin violation with \( \|w\| \):

---

SVMs for Non-Linearity Separable meet my friend the Perceptron...

- Perceptron was minimizing the hinge loss:

\[ \sum_{j=1}^{N} (-y_j^i (w \cdot x^j + w_0))_+ \]

- SVMs minimizes the regularized hinge loss!!

\[ \|w\|_2^2 + C \sum_{j=1}^{N} (1 - y_j^i (w \cdot x^j + w_0))_+ \]
### Stochastic Gradient Descent for SVMs

- **Perceptron minimization:**
  \[ \sum_{j=1}^{N} \left( -y_j (w \cdot x_j + w_0) \right)_+ \]
- **SGD for Perceptron:**
  \[ w^{(t+1)} \leftarrow w^{(t)} + \alpha \left[ y^{(i)} (w^{(t)} \cdot x^{(i)}) \leq 0 \right] y^{(i)} x^{(i)} \]
- **SVMs minimization:**
  \[ ||w||^2 + C \sum_{j=1}^{N} \left( 1 - y_j (w \cdot x_j + w_0) \right)_+ \]
- **SGD for SVMs:**

### What you need to know

- Maximizing margin
- Derivation of SVM formulation
- Non-linearly separable case
  - Hinge loss
  - A.K.A. adding slack variables
- SVMs = Perceptron + L2 regularization
- Can optimize SVMs with SGD
  - Many other approaches possible