Lecture 3: The Perceptron Algorithm
Perceptron algorithm

Algorithm for learning a “large margin” linear separator in \( \mathbb{R}^d \).

Online setting:

- Examples arrive one at a time.
- Given \( x \), predict label \( y \).
- Told correct answer.

Goal: bound number of mistakes under assumption there exists \( w^* \) such that \( w^* \cdot x \geq 1 \) on positives and \( w^* \cdot x \leq -1 \) on negatives.

Perceptron alg: makes at most \( ||w^*||^2 \max( ||x||^2 ) \) mistakes.
Perceptron algorithm

Perceptron alg makes \( \leq \|w^*\|^2 \max( \|x\|^2 ) \) mistakes if \( \exists w^* \) with \( w^* \cdot x \geq 1 \) on all positives and \( w^* \cdot x \leq -1 \) on all negatives.

How to think about this:

- \( \frac{w^* \cdot x}{\|w^*\|} \) is distance of \( x \) to hyperplane \( w^* \cdot x = 0 \).
- Our assumption is equivalent to assuming exists a separator of margin \( \gamma = \frac{1}{\|w^*\|} \).
- If points all lie in a ball of radius \( R \), then mistake bound is at most \( R^2/\gamma^2 \).
- Notice this is scale-invariant.
Perceptron algorithm

Perceptron alg makes \( \leq \|w^*\|^2 \max(\|x\|^2) \) mistakes if \( \exists w^* \) with \( w^* \cdot x \geq 1 \) on all positives and \( w^* \cdot x \leq -1 \) on all negatives.

Algorithm:

- Initialize \( w = 0 \). Predict positive if \( w \cdot x > 0 \), else predict negative.
- Mistake on positive: \( w \leftarrow w + x \).
- Mistake on negative: \( w \leftarrow w - x \).
Perceptron algorithm

Example:
(0.01,1) -
(1,1) +
(1,0) +
(0.01,1) -
(1,1) +
(1,0) +

Algorithm:
Initialize $w = \vec{0}$. Use $w \cdot x > 0$.

- Mistake on pos: $w \leftarrow w + x$.
- Mistake on neg: $w \leftarrow w - x$. 
Analysis

Perceptron alg makes at most $\|w^*\|^2 R^2$ mistakes if $\exists w^*$ with $w^* \cdot x \geq 1$ on all positives and $w^* \cdot x \leq -1$ on all negatives, and all $\|x\| \leq R$.

Proof: consider $w \cdot w^*$ and $\|w\|$

- Each mistake increases $w \cdot w^*$ by at least 1.
  \[(w + x) \cdot w^* = w \cdot w^* + x \cdot w^* \geq w \cdot w^* + 1.\]
  So after $M$ mistakes, $w \cdot w^* \geq M$.

- Each mistake increases $w \cdot w$ by at most $R^2$.
  \[(w + x) \cdot (w + x) = w \cdot w + 2(w \cdot x) + x \cdot x \leq w \cdot w + R^2.\]
  So, after $M$ mistakes, $\|w\|^2 \leq MR^2$, so $\|w\| \leq \sqrt{MR}$.

Since $\frac{w \cdot w^*}{\|w^*\|} \leq \|w\|$, get $\frac{M}{\|w^*\|} \leq \sqrt{MR}$ so $\sqrt{M} \leq \|w^*\| R$. 
Lower bound

Perceptron alg makes at most \( \|w^*\|^2 R^2 \) mistakes if \( \exists w^* \) with \( w^* \cdot x \geq 1 \) on all positives and \( w^* \cdot x \leq -1 \) on all negatives, and all \( \|x\| \leq R \).

In general it’s not possible to get \( < \frac{R^2}{\gamma^2} \) mistakes with a deterministic algorithm.

Proof: consider \( \frac{R^2}{\gamma^2} \) coordinate vectors scaled to length \( R \).

\[
w^* = (\pm x_1 \pm x_2 \pm \cdots \pm x_{\frac{R^2}{\gamma^2}}) / R
\]

\( |w^* \cdot x| = 1 \) for all the input vectors, so can force all mistakes.

\[
\|w^*\| = \sqrt{\frac{R^2}{\gamma^2}} \frac{1}{R} = \frac{1}{\gamma}, \text{ so all margins are } \gamma \text{ as desired.} 
\]
What if no perfect separator?

In this case, a mistake could cause $|w \cdot w^*|$ to drop. The hinge-loss of $w^*$ on positive $x$ is $\max(0, 1 - w^* \cdot x)$: the amount by which the inequality $w^* \cdot x \geq 1$ is not satisfied.

The hinge-loss of $w^*$ on negative $x$ is $\max(0, 1 + w^* \cdot x)$: the amount by which the inequality $w^* \cdot x \leq -1$ is not satisfied.
What if no perfect separator?

In this case, a mistake could cause $|w \cdot w^*|$ to drop.

Theorem: on any sequence of examples $S$, the Perceptron algo makes at most $\min_{w^*}[||w^*||^2 R^2 + 2L_{hinge}(w^*, S)]$ mistakes.

$L_{hinge}(w^*, S) = \text{total hinge loss of } w^* \text{ on set } S. $

Equivalently: how far you would have to move all the points to have them on the correct side by $\gamma$, in units of $\gamma$. 
What if no perfect separator?

In this case, a mistake could cause $|w \cdot w^*|$ to drop.

Theorem: on any sequence of examples $S$, the Perceptron algo makes at most $\min_{w^*} \left[ \|w^*\|^2 R^2 + 2L_{hinge}(w^*, S) \right]$ mistakes.

Proof sketch:

- After $M$ mistakes, $w \cdot w^* \geq M - L_{hinge}(w^*, S)$.
- Still have: after $M$ mistakes, $\|w\|^2 \leq MR^2$.
- Again use fact that $(w \cdot w^*)^2 \leq \|w\|^2 \|w^*\|^2$.
- Solve: $(M - L_{hinge})^2 \leq MR^2 \|w^*\|^2$. Do some algebra.

\[
M^2 - 2ML_{hinge} + L_{hinge}^2 \leq MR^2 \|w^*\|^2
\]
\[
M \leq R^2 \|w^*\|^2 + 2L_{hinge} - L_{hinge}^2/M.
\]
Kernel functions

What if the decision boundary between positive and negatives (e.g., spam and non-spam email) looks more like a circle than a linear separator?

Idea: Kernel functions / “kernel trick”:

- A pairwise function \( K(x, x') \) is a kernel if there exists a function \( \phi \) from input space to a new space (of possibly much higher dimension) such that \( K(x, x') = \phi(x) \cdot \phi(x') \).

- Example: \( K(x, x') = (1 + x \cdot x')^2 \).

- Verify this is a kernel for special case that examples in \( \mathbb{R}^2 \):

\[
K(x, x') = (1 + x_1 x'_1 + x_2 x'_2)^2 = 1 + 2x_1 x'_1 + 2x_2 x'_2 + x_1^2 x'_1^2 + 2x_1 x_2 x'_1 x'_2 + x_2^2 x'_2^2 = \phi(x) \cdot \phi(x') \text{ for } \phi(x) = (1, \sqrt{2} x_1, \sqrt{2} x_2, x_1^2, \sqrt{2} x_1 x_2, x_2^2).
\]
Kernel functions

What if the decision boundary between positive and negatives (e.g., spam and non-spam email) looks more like a circle than a linear separator?

Idea: Kernel functions / “kernel trick”:

- If can modify Perceptron so that only interacts with data via taking dot-products, and then replace $x \cdot x'$ with $K(x, x')$, then algorithm will act as if data was in higher-dimensional $\phi$-space.
- Called “kernelizing” the algorithm.
- E.g., for $\phi(x) = (1, \sqrt{2}x_1, \sqrt{2}x_2, x_1^2, \sqrt{2}x_1x_2, x_2^2)$, the weight vector $w^* = (-100, 0, 0, 1, 0, 1)$ gives a circle of radius 10 as decision boundary $w^* \cdot \phi(x) = 0$. 
Kernel functions

What if the decision boundary between positive and negatives (e.g., spam and non-spam email) looks more like a circle than a linear separator?

Idea: Kernel functions / “kernel trick”:

- If can modify Perceptron so that only interacts with data via taking dot-products, and then replace $x \cdot x'$ with $K(x, x')$, then algorithm will act as if data was in higher-dimensional $\phi$-space.

- How to kernelize Perceptron?

- Easy: weight vector always a sum of previous examples (or their negations), e.g., $w = x^{(1)} + x^{(3)} - x^{(6)}$. So, to predict on new $x$, just compute $w \cdot x = x^{(1)} \cdot x + x^{(3)} \cdot x - x^{(6)} \cdot x$. Now replace dot-product with kernel.