Perceptron
Q: We pick the best hyperparameters by learning on the training data and evaluating error on the validation error. For our final model, should we then learn from training + validation?

A: Yes.

Let's assume that \{train-original\} is the original training data, and \{test\} is the provided test dataset.

1. Split \{train-original\} into \{train-subset\} and \{validation\}.
2. Pick the hyperparameters that when training on \{train-subset\} give the lowest error on \{validation\}. Call these hyperparameters \{best-hyper\}.
3. Retrain a new model using \{best-hyper\} on \{train-original\} = \{train-subset\} \cup \{validation\}.
4. Report test error by evaluating on \{test\}.

Alternatively, you could replace Step 1/2 with the following:
Pick the hyperparameters that give the lowest cross-validation error on \{train-original\}. Call these hyperparameters \{best-hyper\}. 
Reminders

• Homework 2: Decision Trees
  – Out: Wed, Sep 05
  – Due: Wed, Sep 19 at 11:59pm

• Homework 3: KNN, Perceptron, Lin.Reg.
  – Out: Wed, Sep 19
  – Due: Wed, Sep 26 at 11:59pm
THE PERCEPTRON ALGORITHM
Perceptron: History

Imagine you are trying to build a new machine learning technique... your name is Frank Rosenblatt... and the year is 1957.

FIGURE 5
DESIGN OF TYPICAL UNITS
Imagine you are trying to build a new machine learning technique... your name is Frank Rosenblatt... and the year is 1957

*The New Yorker*, December 6, 1958 P. 44

Talk story about the perceptron, a new electronic brain which hasn't been built, but which has been successfully simulated on the I.B.M. 704. Talk with Dr. Frank Rosenblatt, of the Cornell Aeronautical Laboratory, who is one of the two men who developed the prodigy; the other man is Dr. Marshall C. Yovits, of the Office of Naval Research, in Washington. Dr. Rosenblatt defined the perceptron as the first non-biological object which will achieve an organization of its external environment in a meaningful way. It interacts with its environment, forming concepts that have not been made ready for it by a human agent. If a triangle is held up, the perceptron's eye picks up the image & conveys it along a random succession of lines to the response units, where the image is registered. It can tell the difference betw. a cat and a dog, although it wouldn't be able to tell whether the dog was to the left or right of the cat. Right now it is of no practical use, Dr. Rosenblatt conceded, but he said that one day it might be useful to send one into outer space to take in impressions for us.
Linear Models for Classification

Key idea: Try to learn this hyperplane directly

Looking ahead:
- We’ll see a number of commonly used Linear Classifiers
- These include:
  - Perceptron
  - Logistic Regression
  - Naïve Bayes (under certain conditions)
  - Support Vector Machines

Directly modeling the hyperplane would use a decision function:

\[ h(x) = \text{sign}(\theta^T x) \]

for:

\[ y \in \{-1, +1\} \]
In-Class Exercise

Draw a picture of the region corresponding to:
\[ w_1 x_1 + w_2 x_2 + b > 0 \]
where \( w_1 = 2, w_2 = 3, b = 6 \)

Draw the vector \( w = [w_1, w_2] \)
Visualizing Dot-Products

Chalkboard:

– vector in 2D
– line in 2D
– adding a bias term
– definition of orthogonality
– vector projection
– hyperplane definition
– half-space definitions
Linear Models for Classification

Key idea: Try to learn this hyperplane directly

Looking ahead:
• We’ll see a number of commonly used Linear Classifiers
• These include:
  – Perceptron
  – Logistic Regression
  – Naïve Bayes (under certain conditions)
  – Support Vector Machines

Directly modeling the hyperplane would use a decision function:

\[ h(x) = \text{sign}(\theta^T x) \]

for:

\[ y \in \{-1, +1\} \]
Online vs. Batch Learning

**Batch Learning**
Learn from all the examples at once

**Online Learning**
Gradually learn as each example is received
Online Learning

Examples

1. **Stock market** prediction (what will the value of Alphabet Inc. be tomorrow?)

2. **Email** classification (distribution of both spam and regular mail changes over time, but the target function stays fixed - last year's spam still looks like spam)

3. **Recommendation** systems. Examples: recommending movies; predicting whether a user will be interested in a new news article

4. **Ad placement** in a new market
Online Learning

For $i = 1, 2, 3, \ldots$:

- **Receive** an unlabeled instance $x^{(i)}$
- **Predict** $y' = h_{\theta}(x^{(i)})$
- **Receive** true label $y^{(i)}$
- **Suffer loss** if a mistake was made, $y' \neq y^{(i)}$
- **Update** parameters $\theta$

Goal:

- **Minimize** the number of mistakes
Perceptron

Chalkboard:
- (Online) Perceptron Algorithm
- Why do we need a bias term?
- Inductive Bias of Perceptron
- Limitations of Linear Models
Perceptron Algorithm: Example

Example: 
- $(-1,2) - \times$
- $(1,0) + \checkmark$
- $(1,1) + \times$
- $(-1,0) - \checkmark$
- $(-1,-2) - \times$
- $(1,-1) + \checkmark$

Perceptron Algorithm: (without the bias term)
- Set $t=1$, start with all-zeroes weight vector $w_1$.
- Given example $x$, predict positive iff $w_t \cdot x \geq 0$.
- On a mistake, update as follows:
  - Mistake on positive, update $w_{t+1} \leftarrow w_t + x$
  - Mistake on negative, update $w_{t+1} \leftarrow w_t - x$

$w_1 = (0,0)$
$w_2 = w_1 - (-1,2) = (1,-2)$
$w_3 = w_2 + (1,1) = (2,-1)$
$w_4 = w_3 - (-1,-2) = (3,1)$

Slide adapted from Nina Balcan
Background: Hyperplanes

Hyperplane (Definition 1):
\[ \mathcal{H} = \{ \mathbf{x} : \mathbf{w}^T \mathbf{x} = b \} \]

Hyperplane (Definition 2):
\[ \mathcal{H} = \{ \mathbf{x} : \mathbf{\theta}^T \mathbf{x} = 0 \text{ and } x_0 = 1 \} \]
\[ \mathbf{\theta} = [b, w_1, \ldots, w_M]^T \]

Half-spaces:
\[ \mathcal{H}^+ = \{ \mathbf{x} : \mathbf{\theta}^T \mathbf{x} > 0 \text{ and } x_0 = 1 \} \]
\[ \mathcal{H}^- = \{ \mathbf{x} : \mathbf{\theta}^T \mathbf{x} < 0 \text{ and } x_0 = 1 \} \]

Notation Trick: fold the bias \( b \) and the weights \( \mathbf{w} \) into a single vector \( \mathbf{\theta} \) by prepending a constant to \( \mathbf{x} \) and increasing dimensionality by one!
(Online) Perceptron Algorithm

**Data:** Inputs are continuous vectors of length $M$. Outputs are discrete.

$(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \ldots$

where $x \in \mathbb{R}^M$ and $y \in \{+1, -1\}$

**Prediction:** Output determined by hyperplane.

$$\hat{y} = h_{\theta}(x) = \text{sign}(\theta^T x)$$

Assume $\theta = [b, w_1, \ldots, w_M]^T$ and $x_0 = 1$

**Learning:** Iterative procedure:

- **initialize** parameters to vector of all zeroes
- **while** not converged
  - **receive** next example $(x^{(i)}, y^{(i)})$
  - **predict** $y' = h(x^{(i)})$
  - **if** positive mistake: **add** $x^{(i)}$ to parameters
  - **if** negative mistake: **subtract** $x^{(i)}$ from parameters

$$\text{sign}(a) = \begin{cases} 
1, & \text{if } a \geq 0 \\
-1, & \text{otherwise}
\end{cases}$$
Data: Inputs are continuous vectors of length $M$. Outputs are discrete. 

\[ (x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \ldots \]
where $x \in \mathbb{R}^M$ and $y \in \{+1, -1\}$

Prediction: Output determined by a hyperplane.

\[ \hat{y} = h_\theta(x) = \text{sign}(\theta^T x) \]

Assume $\theta = [b, w_1, \ldots, w_M]$

Learning:

**Algorithm 1 Perceptron Learning Algorithm**

1: \textbf{procedure} PERCEPTRON($\mathcal{D} = \{(x^{(i)}, y^{(i)})\}$) 
2: \hspace{1em} $\theta \leftarrow 0$
3: \hspace{1em} \textbf{for} $i \in \{1, 2, \ldots\}$ \textbf{do}
4: \hspace{2em} $\hat{y} \leftarrow \text{sign}(\theta^T x^{(i)})$
5: \hspace{2em} \textbf{if} $\hat{y} \neq y^{(i)}$ \textbf{then}
6: \hspace{3em} $\theta \leftarrow \theta + y^{(i)} x^{(i)}$
7: \hspace{1em} \textbf{return} $\theta$

- Initialize parameters
- For each example
  - Predict
  - If mistake
  - Update parameters

**Implementation Trick:** same behavior as our "add on positive mistake and subtract on negative mistake" version, because $y^{(i)}$ takes care of the sign.
(Batch) Perceptron Algorithm

Learning for Perceptron also works if we have a fixed training dataset, D. We call this the “batch” setting in contrast to the “online” setting that we’ve discussed so far.

Algorithm 1 Perceptron Learning Algorithm (Batch)

1: procedure PERCEPTRON($\mathcal{D} = \{(x^{(1)}, y^{(1)}), \ldots, (x^{(N)}, y^{(N)})\}$)
2: $\theta \leftarrow 0$  ▷ Initialize parameters
3: while not converged do  ▷ For each example
4:     for $i \in \{1, 2, \ldots, N\}$ do  ▷ Predict
5:         $\hat{y} \leftarrow \text{sign}(\theta^T x^{(i)})$
6:         if $\hat{y} \neq y^{(i)}$ then  ▷ If mistake
7:             $\theta \leftarrow \theta + y^{(i)} x^{(i)}$  ▷ Update parameters
8: return $\theta$
Learning for Perceptron also works if we have a fixed training dataset, $D$. We call this the “batch” setting in contrast to the “online” setting that we’ve discussed so far.

**Discussion:**
The Batch Perceptron Algorithm can be derived in two ways.

1. By extending the online Perceptron algorithm to the batch setting (as mentioned above)
2. By applying **Stochastic Gradient Descent (SGD)** to minimize a so-called **Hinge Loss** on a linear separator
Extensions of Perceptron

• **Voted Perceptron**
  – generalizes better than (standard) perceptron
  – memory intensive (keeps around every weight vector seen during training, so each one can vote)

• **Averaged Perceptron**
  – empirically similar performance to voted perceptron
  – can be implemented in a memory efficient way (running averages are efficient)

• **Kernel Perceptron**
  – Choose a kernel $K(x', x)$
  – Apply the **kernel trick** to Perceptron
  – Resulting algorithm is **still very simple**

• **Structured Perceptron**
  – Basic idea can also be applied when $y$ ranges over an exponentially large set
  – Mistake bound **does not** depend on the size of that set
ANALYSIS OF PERCEPTRON
**Definition:** The margin of example \( x \) w.r.t. a linear sep. \( w \) is the distance from \( x \) to the plane \( w \cdot x = 0 \) (or the negative if on wrong side).
Geometric Margin

**Definition:** The margin of example $x$ w.r.t. a linear separator $w$ is the distance from $x$ to the plane $w \cdot x = 0$ (or the negative if on wrong side).

**Definition:** The margin $\gamma_w$ of a set of examples $S$ wrt a linear separator $w$ is the smallest margin over points $x \in S$. 

Slide from Nina Balcan
**Geometric Margin**

**Definition:** The margin of example $x$ w.r.t. a linear sep. $w$ is the distance from $x$ to the plane $w \cdot x = 0$ (or the negative if on wrong side).

**Definition:** The margin $\gamma_w$ of a set of examples $S$ wrt a linear separator $w$ is the smallest margin over points $x \in S$.

**Definition:** The margin $\gamma$ of a set of examples $S$ is the maximum $\gamma_w$ over all linear separators $w$. 

Slide from Nina Balcan
**Def:** For a *binary classification* problem, a set of examples $S$ is **linearly separable** if there exists a linear decision boundary that can separate the points.

Case 1: 

```
+  
-  
+  
```

Case 2: 

```
-  
+  
+  
```

Case 3: 

```
+  
+  
+  
```

Case 4: 

```
+  
-  
+  
```
Analysis: Perceptron

Perceptron Mistake Bound

**Guarantee:** If data has margin $\gamma$ and all points inside a ball of radius $R$, then Perceptron makes $\leq \left(\frac{R}{\gamma}\right)^2$ mistakes.

(Normalized margin: multiplying all points by 100, or dividing all points by 100, doesn’t change the number of mistakes; algo is invariant to scaling.)
Analysis: Perceptron

Perceptron Mistake Bound

**Guarantee:** If data has margin $\gamma$ and all points inside a ball of radius $R$, then Perceptron makes $\leq \left(\frac{R}{\gamma}\right)^2$ mistakes.

(Normalized margin: multiplying all points by 100, or dividing all points by 100, doesn’t change the number of mistakes; algo is invariant to scaling.)

**Def:** We say that the (batch) perceptron algorithm has **converged** if it stops making mistakes on the training data (perfectly classifies the training data).

**Main Takeaway:** For **linearly separable** data, if the perceptron algorithm cycles repeatedly through the data, it will **converge** in a finite # of steps.
**Perceptron Mistake Bound**

**Theorem 0.1** (Block (1962), Novikoff (1962)).

Given dataset: \( \mathcal{D} = \{ (x^{(i)}, y^{(i)}) \}_{i=1}^{N} \).

Suppose:

1. **Finite size inputs:** \( ||x^{(i)}|| \leq R \)
2. **Linearly separable data:** \( \exists \theta^* \) s.t. \( ||\theta^*|| = 1 \) and \( y^{(i)}(\theta^* \cdot x^{(i)}) \geq \gamma, \forall i \)

Then: The number of mistakes made by the Perceptron algorithm on this dataset is

\[
k \leq (R/\gamma)^2
\]
Proof of Perceptron Mistake Bound:

We will show that there exist constants A and B s.t.

\[ Ak \leq \|\theta^{(k+1)}\| \leq B\sqrt{k} \]
Analysis: Perceptron

**Theorem 0.1** (Block (1962), Novikoff (1962)).

Given dataset: \( D = \{(x(i), y(i))\}_{i=1}^{N} \).

Suppose:

1. **Finite size inputs:** \( ||x^{(i)}|| \leq R \)
2. **Linearly separable data:** \( \exists \theta^* \) s.t. \( ||\theta^*|| = 1 \) and \( y^{(i)}(\theta^* \cdot x^{(i)}) \geq \gamma, \forall i \)

Then: The number of mistakes made by the Perceptron algorithm on this dataset is

\[
k \leq \left(\frac{R}{\gamma}\right)^2
\]

**Algorithm 1** Perceptron Learning Algorithm (Online)

1: **procedure** PERCEPTRON\( (D = \{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \ldots\}) \)  
2: \( \theta \leftarrow 0, k = 1 \)  
3: **for** \( i \in \{1, 2, \ldots\} \) **do**  
4: \hspace{1em} **if** \( y^{(i)}(\theta^{(k)} \cdot x^{(i)}) \leq 0 \) **then**  
5: \hspace{2em} \( \theta^{(k+1)} \leftarrow \theta^{(k)} + y^{(i)} x^{(i)} \)  
6: \hspace{1em} \( k \leftarrow k + 1 \)  
7: **return** \( \theta \)
Proof of Perceptron Mistake Bound:
Part 1: for some A,  \( Ak \leq \|\theta^{(k+1)}\| \)

\[ \theta^{(k+1)} \cdot \theta^* = (\theta^{(k)} + y^{(i)} x^{(i)}) \cdot \theta^* \]

by Perceptron algorithm update

\[ = \theta^{(k)} \cdot \theta^* + y^{(i)} (\theta^* \cdot x^{(i)}) \]

\[ \geq \theta^{(k)} \cdot \theta^* + \gamma \]

by assumption

\[ \Rightarrow \theta^{(k+1)} \cdot \theta^* \geq k \gamma \]

by induction on \( k \) since \( \theta^{(1)} = 0 \)

\[ \Rightarrow \|\theta^{(k+1)}\| \geq k \gamma \]

since \( \|w\| \times \|u\| \geq w \cdot u \) and \( \|\theta^*\| = 1 \)

Cauchy-Schwartz inequality
Proof of Perceptron Mistake Bound:
Part 2: for some $B$, $\|\theta^{(k+1)}\| \leq B\sqrt{k}$

$\|\theta^{(k+1)}\|^2 = \|\theta^{(k)} + y^{(i)}x^{(i)}\|^2$

by Perceptron algorithm update

$= \|\theta^{(k)}\|^2 + (y^{(i)})^2\|x^{(i)}\|^2 + 2y^{(i)}(\theta^{(k)} \cdot x^{(i)})$

$\leq \|\theta^{(k)}\|^2 + (y^{(i)})^2\|x^{(i)}\|^2$

since $k$th mistake $\Rightarrow y^{(i)}(\theta^{(k)} \cdot x^{(i)}) \leq 0$

$= \|\theta^{(k)}\|^2 + R^2$

since $(y^{(i)})^2\|x^{(i)}\|^2 = \|x^{(i)}\|^2 = R^2$ by assumption and $(y^{(i)})^2 = 1$

$\Rightarrow \|\theta^{(k+1)}\|^2 \leq kR^2$

by induction on $k$ since $(\theta^{(1)})^2 = 0$

$\Rightarrow \|\theta^{(k+1)}\| \leq \sqrt{kR}$
Proof of Perceptron Mistake Bound:
Part 3: Combining the bounds finishes the proof.

\[ k \gamma \leq \left\| \theta^{(k+1)} \right\| \leq \sqrt{kR} \]

\[ \Rightarrow k \leq \left( \frac{R}{\gamma} \right)^2 \]

The total number of mistakes must be less than this.
Analysis: Perceptron

What if the data is not linearly separable?

1. Perceptron will **not converge** in this case (it can’t!)
2. However, Freund & Schapire (1999) show that by projecting the points (hypothetically) into a higher dimensional space, we can achieve a similar bound on the number of mistakes made on **one pass** through the sequence of examples.

**Theorem 2.** Let \( \langle (x_1, y_1), \ldots, (x_m, y_m) \rangle \) be a sequence of labeled examples with \( \|x_i\| \leq R \). Let \( u \) be any vector with \( \|u\| = 1 \) and let \( \gamma > 0 \). Define the deviation of each example as

\[
d_i = \max\{0, \gamma - y_i(u \cdot x_i)\},
\]

and define \( D = \sqrt{\sum_{i=1}^{m} d_i^2} \). Then the number of mistakes of the online perceptron algorithm on this sequence is bounded by

\[
\left( \frac{R + D}{\gamma} \right)^2.
\]
Summary: Perceptron

- Perceptron is a **linear classifier**
- **Simple learning algorithm**: when a mistake is made, add / subtract the features
- Perceptron will converge if the data are **linearly separable**, it will **not** converge if the data are **linearly inseparable**
- For linearly separable and inseparable data, we can **bound the number of mistakes** (geometric argument)
- **Extensions** support nonlinear separators and structured prediction
Perceptron Learning Objectives

You should be able to...

• Explain the difference between online learning and batch learning
• Implement the perceptron algorithm for binary classification [CIML]
• Determine whether the perceptron algorithm will converge based on properties of the dataset, and the limitations of the convergence guarantees
• Describe the inductive bias of perceptron and the limitations of linear models
• Draw the decision boundary of a linear model
• Identify whether a dataset is linearly separable or not
• Defend the use of a bias term in perceptron