10-315 INTRODUCTION TO MACHINE LEARNING (SCS MAJORS)
LECTURE 1: INTRODUCTION

LEILA WEHBE

CARNEGIE MELLON UNIVERSITY

MACHINE LEARNING DEPARTMENT

Lecture based on **chapter 4** from Hal Daumé III, on Kilian Weinberger's **lecture 3**, on Tom Mitchell's **lecture 1** and Matt Gormley's **lecture 1**.

LECTURE OUTCOMES

- Core concepts and problem definitions in Machine Learning
- Overview of applications

LINKS (USE THE VERSION YOU NEED)

- **Notebook**
- **PDF slides**

WELCOME TO 10-315 INTRO TO MACHINE LEARNING

Lectures: MW, 9:30-10:50am, GHC 4307
Recitations: F, 10:10-11:30am, GHC 4307

Instructor:
Leila Wehbe

WELCOME TO 10-315 INTRO TO MACHINE LEARNING

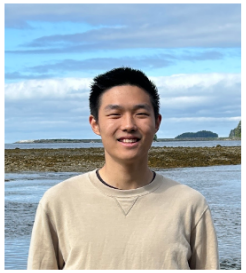
Education Associate:
Nichelle Phillips

WELCOME TO 10-315 INTRO TO MACHINE LEARNING

Teaching Assistants:

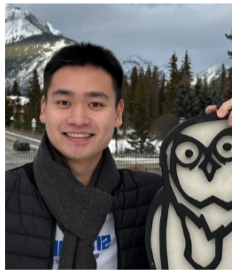
Yuki Minai

Derek Yuan



Ethan Wang

Jason Yuan



Jerick Shi

LINKS

- **Website** (includes **schedule** and links to lectures)
- **Piazza**
- **Syllabus**

SCHEDULE:

- Exam 1: October 9th
- Exam 2: December 2nd (non-comprehensive)
- Lectures on Monday and Wednesday, Recitation most Fridays.
    - EXCEPTION: no lecture on Labor Day (September 2).
    - EXCEPTION: no lecture on September 9th, instead we will have lecture that week on Wed and Fri (11th and 13th).

Mon	Aug-26	Lec 1	Intro - learning paradigms - function approximation				
Wed	Aug-28	Lec 2	Perceptron - learning linear separators - margins				
Fri	Aug-30	Reci 1					
Mon	Sep-02	--	Labor Day				
Wed	Sep-04	Lec 3	Decision trees - overfitting				
Fri	Sep-06	Reci 2					
Mon	Sep-09	--	No class - class on Friday				
Wed	Sep-11	Lec 4	K-nearest neighbors				
Fri	Sep-13	Lec 5	Bayes Rule - MLE - MAP				
Mon	Sep-16	Lec 6	Naive Bayes				
Wed	Sep-18	Lec 7	Optimization for ML				
Fri	Sep-20	Reci 4					

- Homework 0 out this Thursday.

HIGHLIGHTS OF COURSE LOGISTICS

- 7 HW assignments (60%)
- 2 exams (see schedule, 20% each)
- Homework assignments will be submitted on gradescope.
- 8 late days in total, maximum 3 per assignment.
- Collaboration is ok if you only talk to each other, and then write / implement separately.
- **Collaboration should be disclosed.** There is a dedicated section for each homework assignment.
    - What happens if you disclose / don't disclose?
- What happens if you copy code from someone else (even if you change it)?
- What happens if you use generative AI to create your code?

AIV RISK DUE TO GENERATIVE AI USE

Nichelle has put together a list of common reasons students turned to ChatGPT for help and alternative actions which are less likely to raise flags. It is our hope that you learn from this collective experience and complete this course responsibly.

- Reason: I used ChatGPT because I was in a time crunch.
- Alternative: This is the most common reason students turn to ChatGPT. We suggest starting your assignments as early as possible. You will not know how much time an assignment will take to complete until you attempt it. If you find that will not meet the deadline due to an emergency, reach out to the course EA before the assignment is due or as soon as you are able.

AIV RISK DUE TO GENERATIVE AI USE

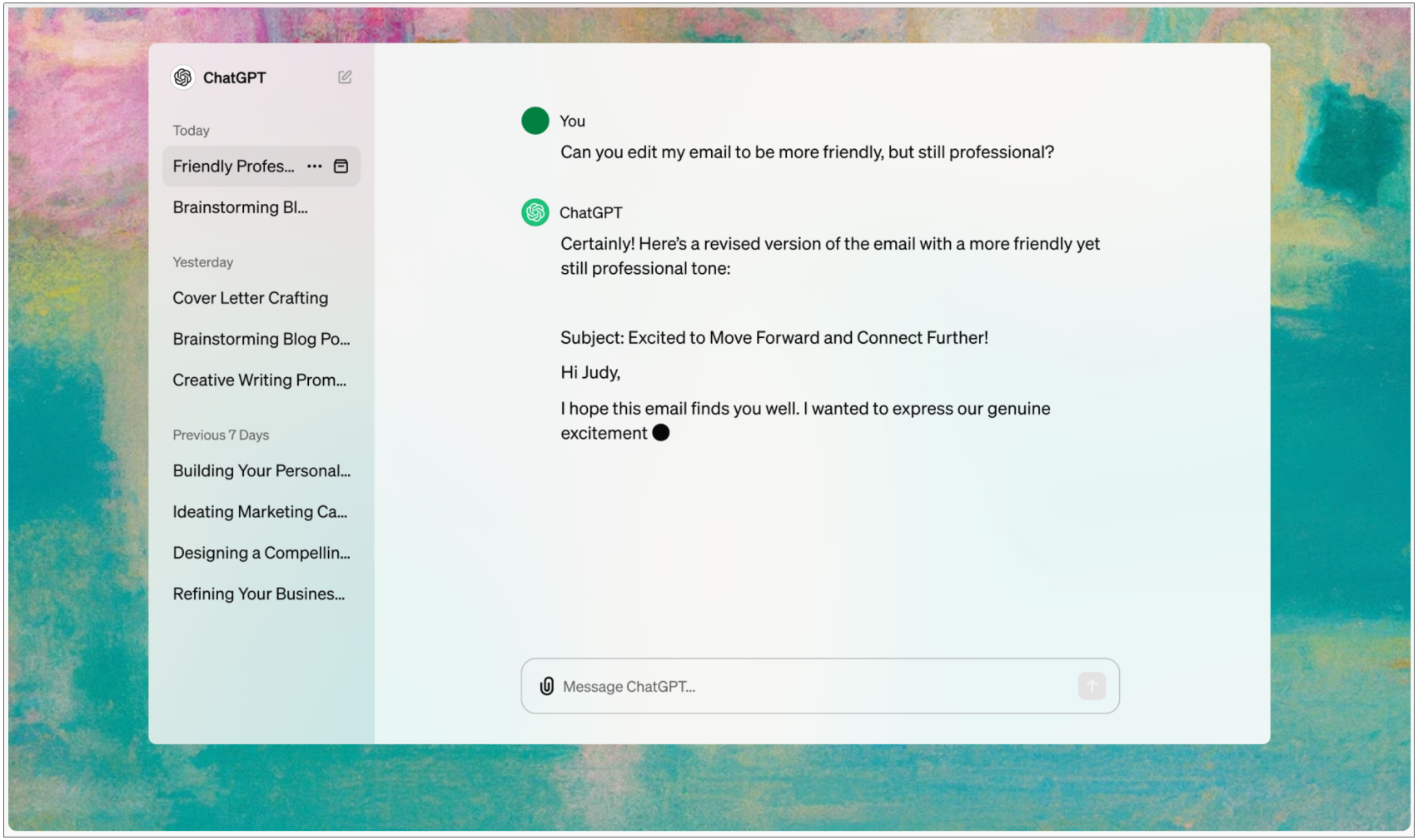
- Reason: I used ChatGPT to look up numpy functions.
- Alternative: We suggest you use numpy.org. Again, if you still intend to use ChatGPT, which is not recommended, be sure to prepend your prompt with “don’t give me any code in any language”. Reminder: this advice is not a guarantee that you will not be flagged for a potential AIV.
- Reason: I used ChatGPT to debug my code.
- Alternative: Bring detailed pseudo code to office hours that describes your implementation design. If you do not have pseudo code, the TA will not look at your code, but instead ask you to sketch out pseudo code at the chalkboard and discuss it from there. After discussing at a high-level if your 10 minutes have not expired, the TA may have time to look at your code. Reminder: This is not a programming course; you are expected to know how to debug code. Giving your code to ChatGPT will result in an AIV.

WHAT IS MACHINE LEARNING?

- "How can we build computer programs that automatically improve their performance through experience?"
    - Study of algorithms that
        - improve their performance **P**
        - at some task **T**
        - with experience **E**
    - well-defined learning task: (**P**,**T**,**E**)

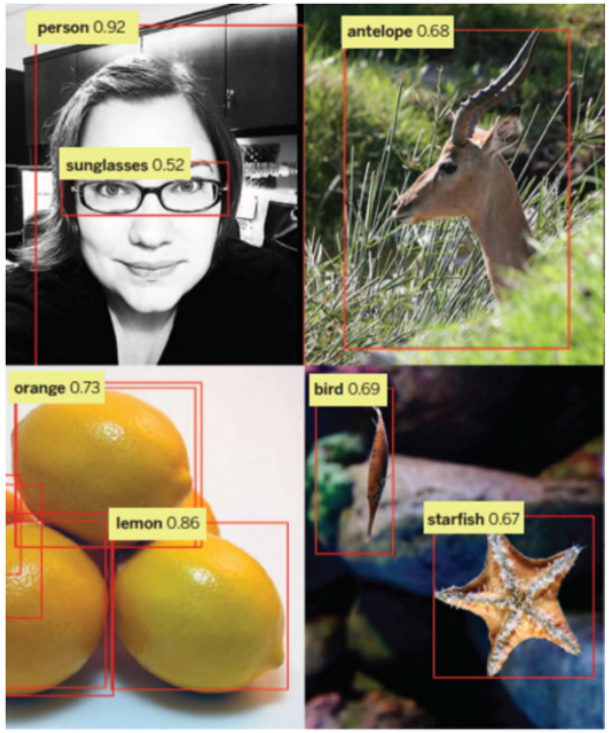
- How can we learn from data?
- How robust is what we learn? What types of assumptions do we make with different approaches? What are the guarantees? How do we pick an approach?

NATURAL LANGUAGE PROCESSING

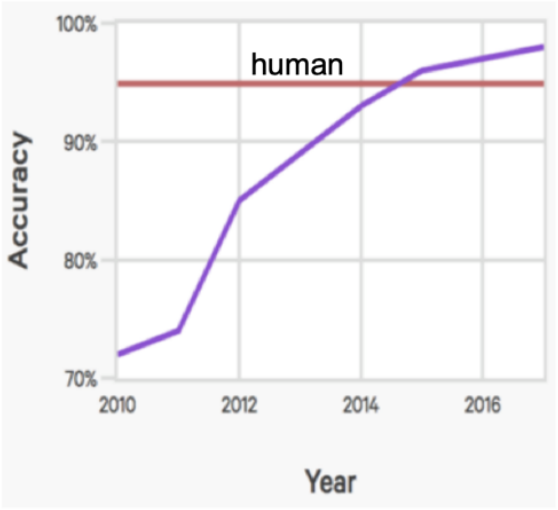


from **https://openai.com/chatgpt/**

COMPUTER VISION

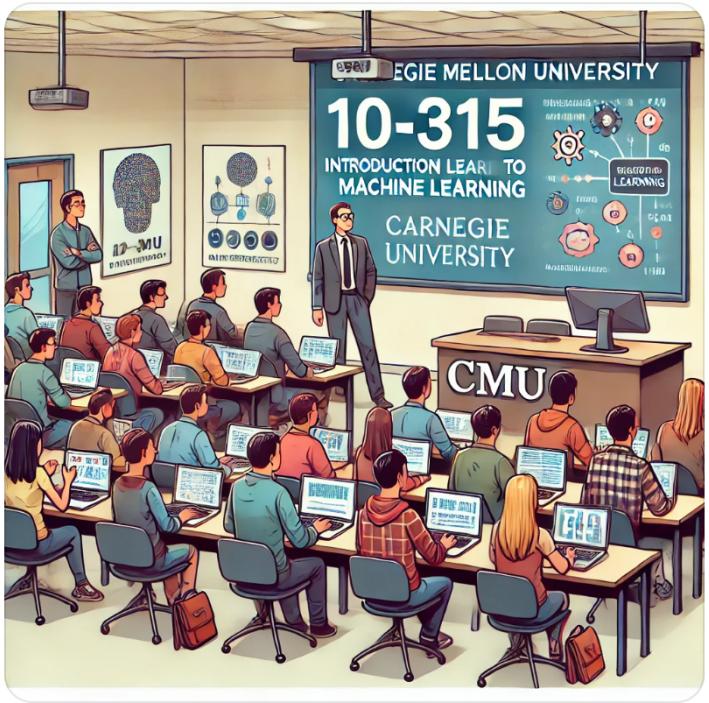


Imagenet Visual Recognition Challenge



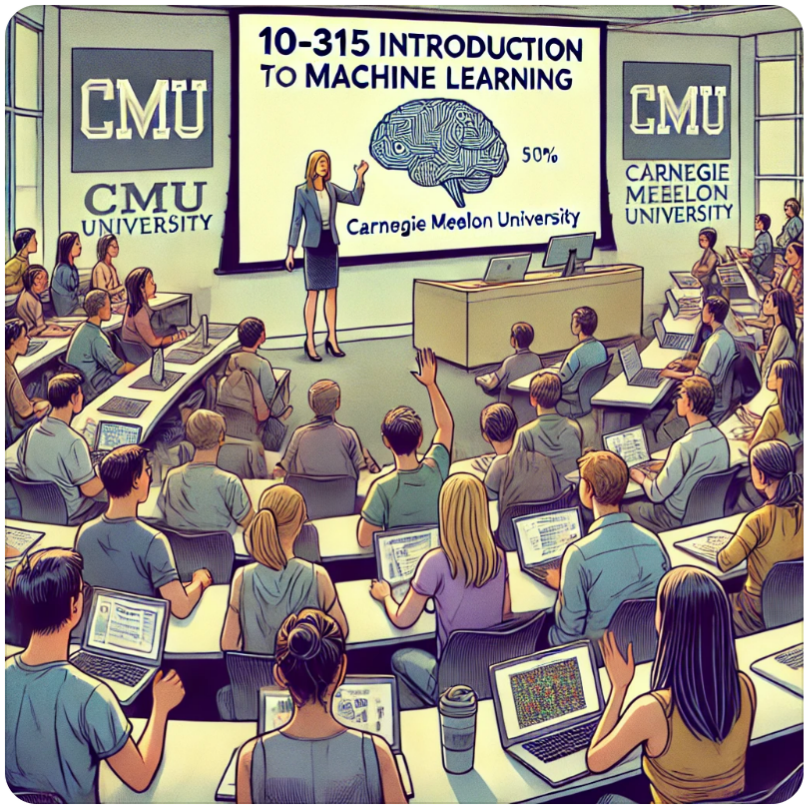
MULTIMODAL AI

can you draw an image that illustrate the first day of class for the 10315 introduction to machine learning for computer science majors at CMU?



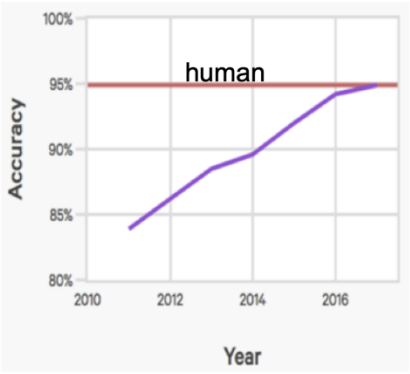
by chatgpt

MULTIMODAL AI -- SECOND TRY



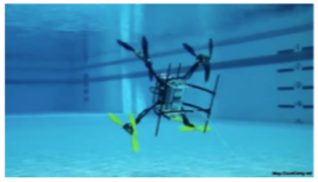
by chatgpt

SPEECH RECOGNITION



ROBOTS

Factories, Land, Air, Sea, Mines, Homes

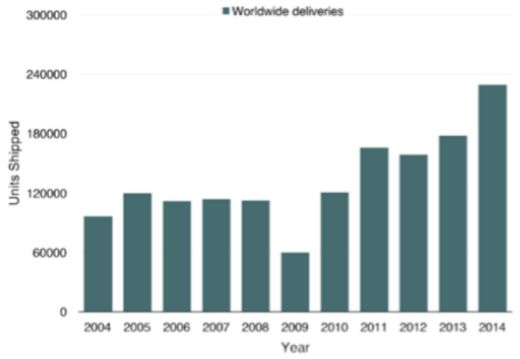


FIGURE 2.4 Worldwide shipping of robots over time. SOURCE: International Federation of Robotics, 2015.

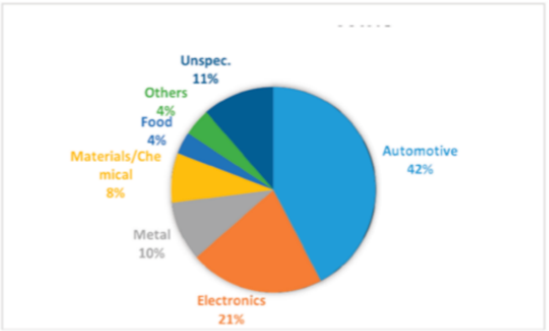
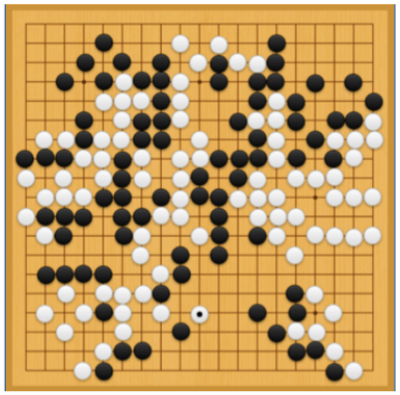


FIGURE 2.5 Robot application areas in 2015. SOURCE: Data from International Federation of Robotics, 2015.

GAMES AND REASONING

Chess

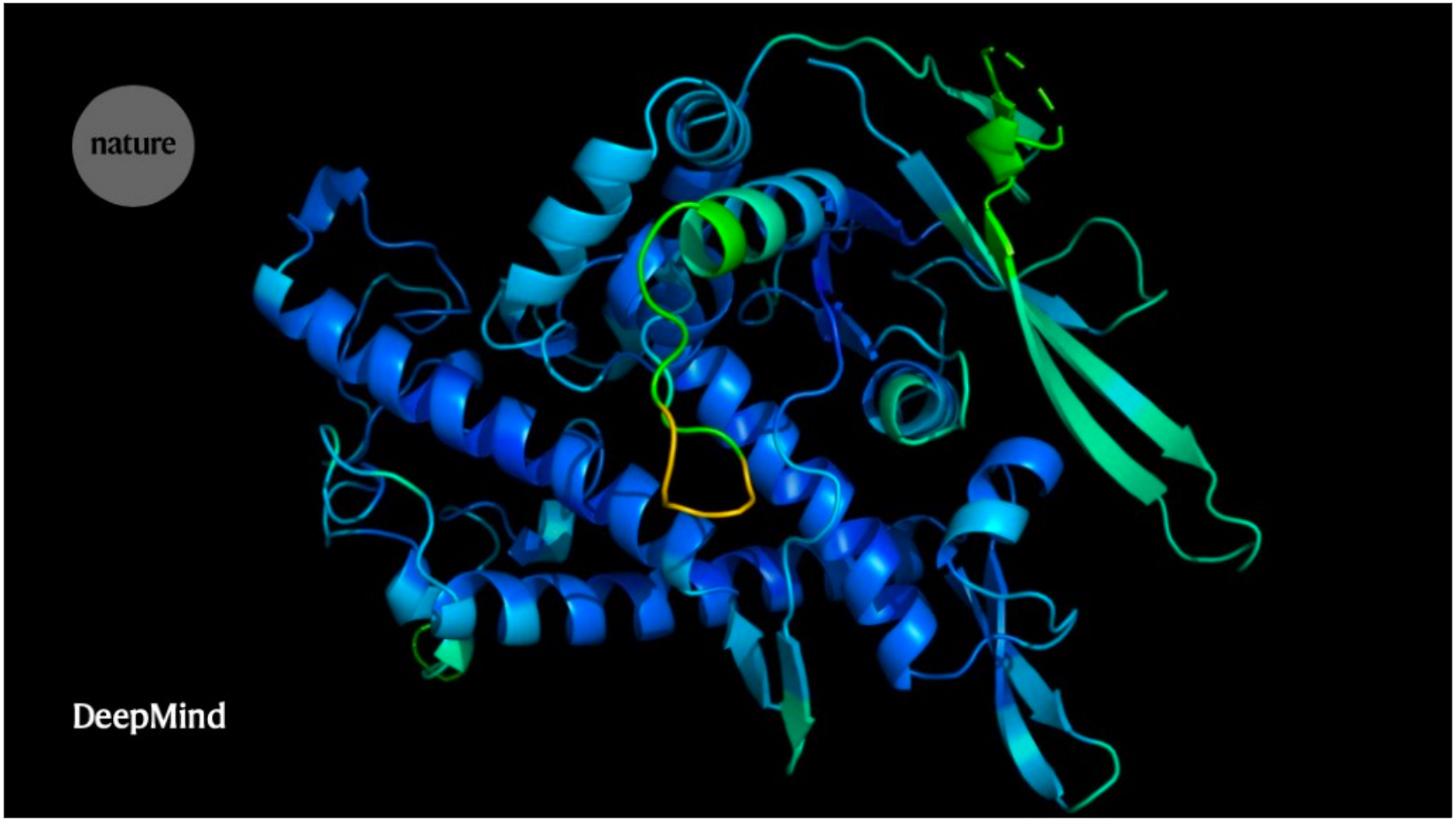


Go

Jeopardy

Poker

PROTEIN FOLDING

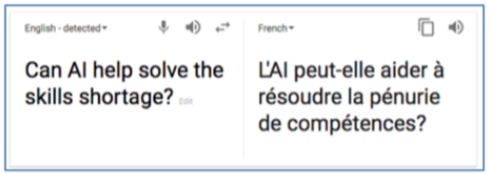


THE KEY: MACHINE LEARNING

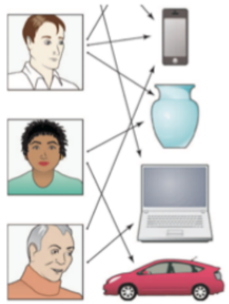
conversational agents

medical diagnosis

fraud detection



translation



recommendations

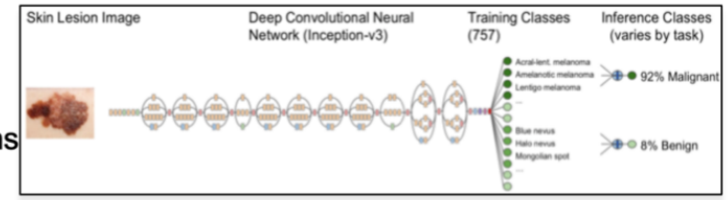
Many algorithms:

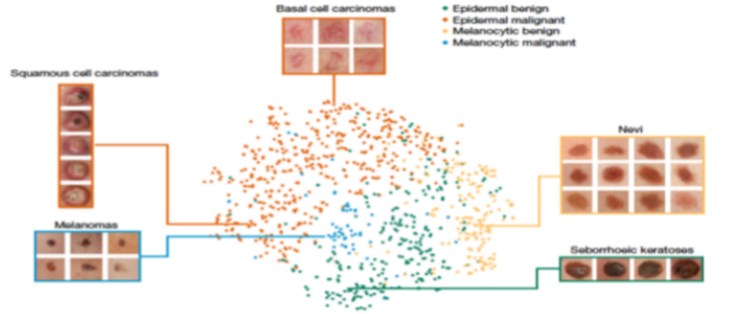
- Deep neural networks
- Bayesian networks
- Hidden Markov models
- Support Vector Machines
- Gaussian mixture model
- Expectation maximization
- ....

SKIN CANCER DIAGNOSIS

[Esteva et al., *Nature* 2017]

Trained on 129,450 skin images
plus 1.4 million standard photographs
Deep net Inception v3 architecture
Outperforms doctors





tsne visualization of final hidden layer

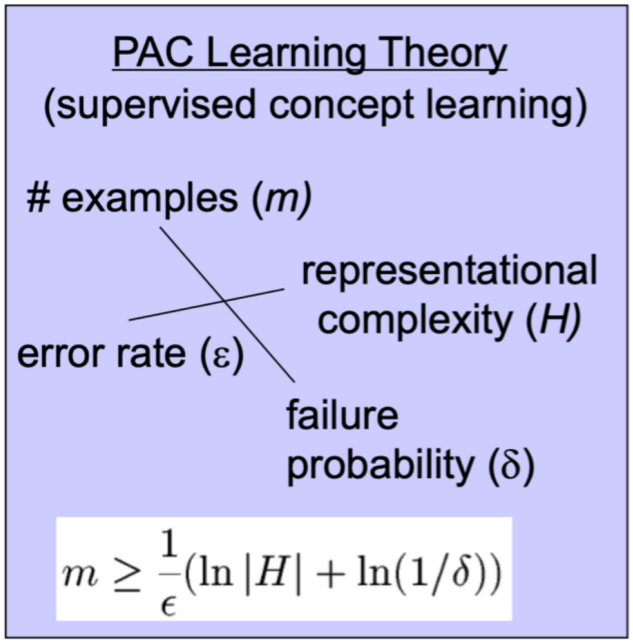
PREDICT CARDIOVASCULAR RISK FROM RETINAL PHOTOGRAPHS

[Poplin et al., *Nature Biomed Eng.* 2018]

Trained deep net on 284,335 retinal images
New approach to detecting risk factors and biometrics

	Accuracy
Age	within 3.26 years on average
Smoker?	71%
Systolic blood pressure	within 11 mmHg on average
Gender	97%
Major cardiac event within past 5 years?	70%

MACHINE LEARNING THEORY

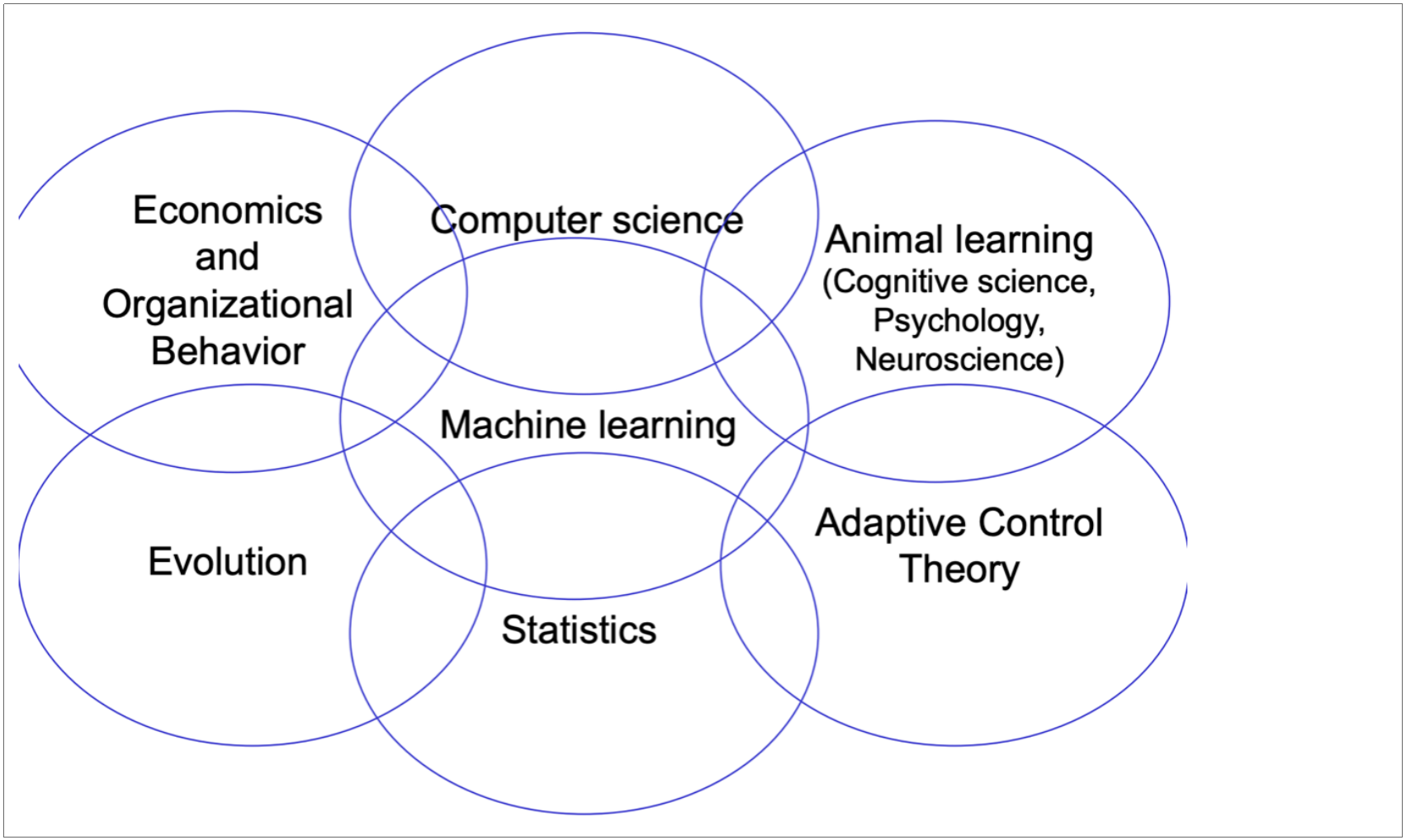


Other theories for

- Reinforcement skill learning
- Semi-supervised learning
- Active student querying
- …

… also relating:

- # of mistakes during learning
- learner's query strategy
- convergence rate
- computational demands
- asymptotic performance
- bias, variance, Bayesian priors
- VC dimension


Economics
and
Organizational
Behavior
Computer science
Animal learning
(Cognitive science,
Psychology,
Neuroscience)
Machine learning
Evolution
Statistics
Adaptive Control
Theory

SOCIAL IMPACTS OF MACHINE LEARNING

- Better, evidence-based, decision making in many domains
    - Medical diagnosis, Credit card fraud detection, Online tutoring, Anticipating equipment failures, Marketing, Legal sentencing, …
- Created breakthroughs in AI, with huge impact on society
    - Computer vision, speech, text processing, self-driving cars, games, …
- Raises new issues
    - Explainability
    - Bias
    - Privacy
    - If big data is key to successful ML, who controls access to the data?
    - …

WE WILL COVER IN THIS COURSE

Algorithms:

- Decision trees
- Bayes classifiers
- Logistic regression
- Deep neural networks
- Graphical models
- Expectation maximization
- Support Vector Machines
- Kernel regression
- PCA
- Reinforcement learning

Concepts:

- Statistical estimation
- Overfitting
- Representation learning
- Probabilistic models
- Maximum margin models
- Probably approximately correct learning
- VC dimension
- Role of unlabeled data
- Optimization

MACHINE LEARNING ALGORITHMS

- Supervised learning
    - Classification
    - Regression

SUPERVISED LEARNING PROBLEM STATEMENT

The goal is to learn a function c^* that maps input variables X to output variables y, based on a set of labeled training examples.

- classification: y is binary or multiclass
- regression: y is continuous

Training Data: Given a training set of n labeled examples:$\{(X_1, y_1), (X_2, y_2), \ldots, (X_n, y_n)\}$, where $X_i \in \mathcal{X}$ represents the input features and $y_i \in \mathcal{Y}$ represents the corresponding labels, the goal is to estimate the optimal function c^* that best predicts the labels for new, unseen data.

Hypothesis Space: The function c^* is chosen from a family of hypotheses $\mathcal{H}$. That is, $c^* \in \mathcal{H}$, where $\mathcal{H}$ represents the set of all possible functions that could map inputs to outputs.

Learning Rule: A learning rule is applied to select the optimal function c^* from the hypothesis space $\mathcal{H}$. The learning rule is typically defined based on an optimization algorithm that seeks to minimize a cost function over the training data.

SUPERVISED LEARNING PROBLEM STATEMENT (CONTINUED)

Loss Function: A loss function $L(y, \hat{y})$ quantifies the error between the predicted value $\hat{y} = c(X)$ and the actual value y for a single data point.

- Examples of Loss Functions
  - **0-1 Loss (for Classification)**: The 0-1 loss function is used in classification tasks and is defined for. single point as:

    $$L(y, \hat{y}) = \begin{cases} 0, & \text{if } y = \hat{y} \\ 1, & \text{if } y \neq \hat{y} \end{cases}$$

    This loss function counts the number of incorrect predictions, and the goal is to minimize the number of errors.
  - The loss over a dataset (also refered to as the error rate): $\frac{1}{n} \sum_{i=1}^{n} L(y_i, \hat{y}_n)$

- **Mean Squared Error (MSE for Regression)**: The MSE loss function is commonly used in regression tasks and is defined for a single point as:

  $$L(y, \hat{y}) = (y - \hat{y})^2$$

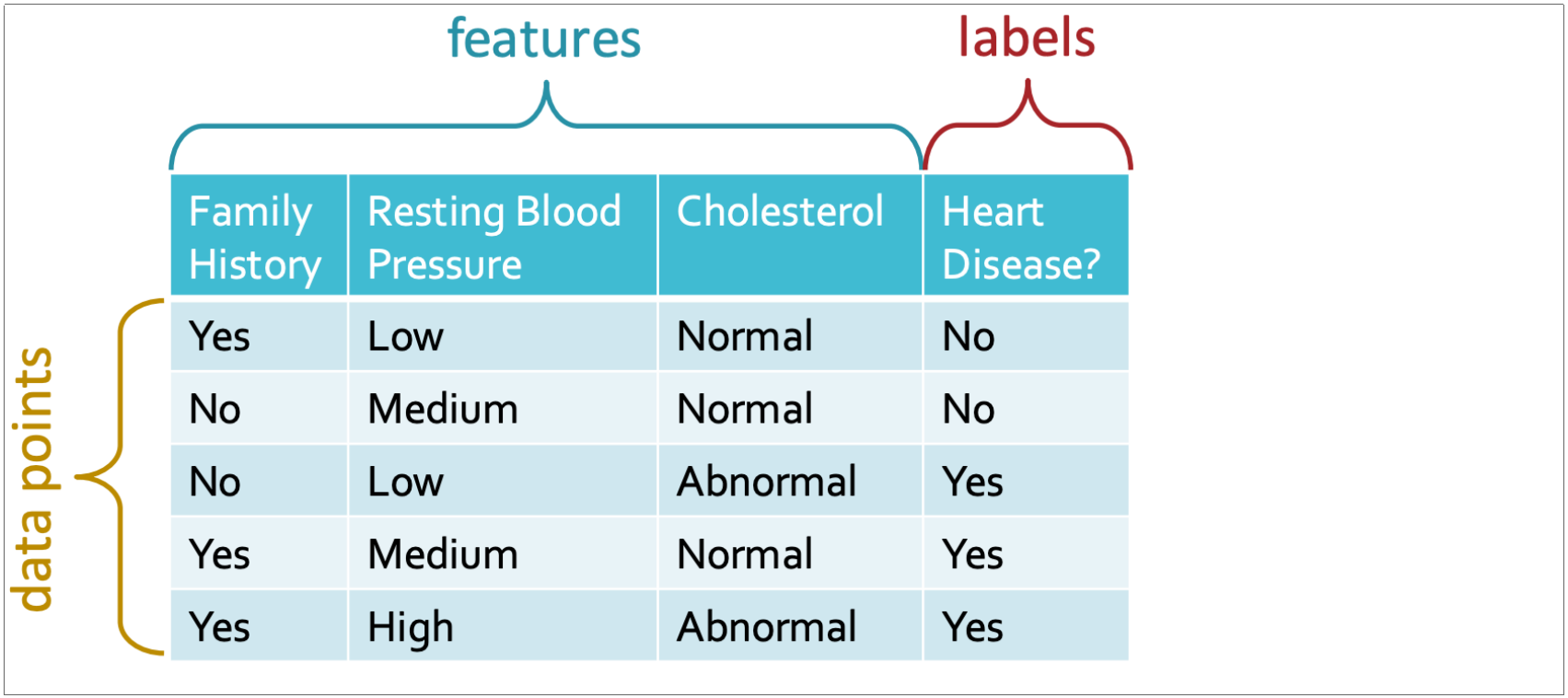
  - The MSE is the average of squared differences over all training examples:

$\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$

Test Data: A set of m labeled examples:$\{(X_j, y_j), j \in 1 \ldots m\}$, which is sampled from the same distribution as the training set.

EXAMPLE CLASSIFIER

Given the dataset:



Family History	Resting Blood Pressure	Cholesterol	Heart Disease?
Yes	Low	Normal	No
No	Medium	Normal	No
No	Low	Abnormal	Yes
Yes	Medium	Normal	Yes
Yes	High	Abnormal	Yes

- Compute the majority vote classifier, what is the training error rate?
- Decision stump on Family History

EXAMPLE CLASSIFIER

test dataset

Family History	Resting Blood Pressure	Cholesterol	Heart Disease?	**Predictions**
No	Low	Normal	No	Yes
No	High	Abnormal	Yes	Yes
Yes	Medium	Abnormal	Yes	Yes

- What is the test error rate for both?

MACHINE LEARNING ALGORITHMS

- Supervised learning
    - Classification
    - Regression
- Unsupervised learning
    - Specific case: self-supervised learning

SELF-SUPERVISED LEARNING PROBLEM STATEMENT

The model learns to predict part of the input data from other parts of the input data.
Training Data: Given a set of unlabeled data points:$\{X_1, X_2, \ldots, X_n\}$, where $X_i \in \mathcal{X}$ represents the input features, the model is trained to predict some aspect of X from other aspects of X.
Unlike supervised learning, there are no explicit labels y_i provided by humans.
Examples:

- Language modeling: given words in a sequence, predict the next word.
- Image inpainting: fill-in the missing parts of an image.
- Contrastive learning: Learn to distinguish between different transformations or augmentations of the same data point X_i. The task is to bring representations of similar pairs (e.g., different augmentations of the same image) closer in the feature space while pushing representations of dissimilar pairs apart.

MACHINE LEARNING ALGORITHMS

- Supervised learning
    - Classification
    - Regression
- Unsupervised learning
    - Specific case: self-supervised learning
- Reinforcement learning

REINFORCEMENT LEARNING PROBLEM STATEMENT

RL is a type of machine learning where an agent learns to make decisions by interacting with an environment. The goal is for the agent to learn a policy that maximizes cumulative rewards over time.

Agent: the learner or decision-maker that interacts with the environment. At each time step, the agent observes the current **state** s_t, takes an **action** a_t, and receives a **reward** r_t from the environment.

Policy: A **policy** $\pi(a|s)$ is a mapping from states to probabilities of selecting each possible action. The goal of reinforcement learning is to find an optimal policy π^* that maximizes the expected cumulative reward over time.

$$\pi^* = \arg\max_{\pi} \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r_t\right]$$

where $\gamma \in [0, 1]$ is the discount factor that balances immediate and future rewards.

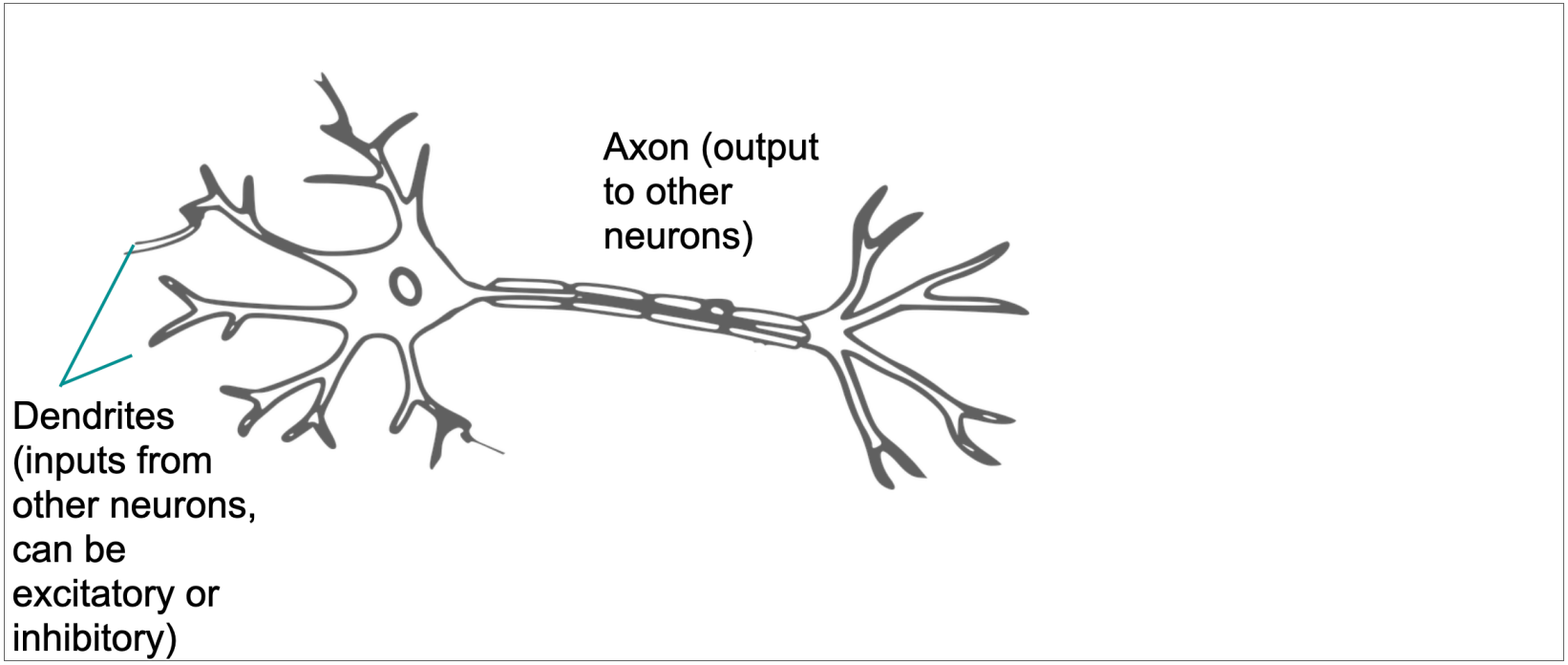
Learning algorithm: How to learn the policy? We will explore multiple approches later in the course.

CLASS OUTLINE:

- Supervised learning:
    - Perceptron

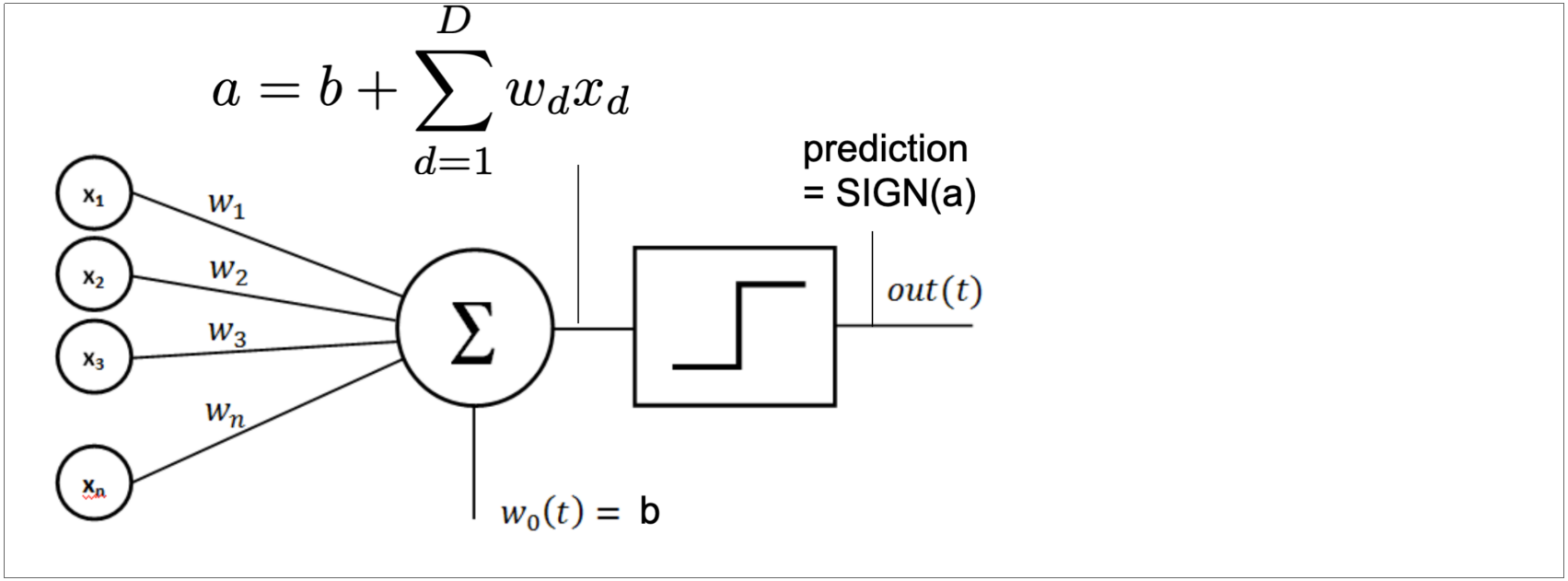
THE PERCEPTRON

- Introduced by Rosenblatt in 1958
- Inspired by real neurons



THE PERCEPTRON

- Introduced by Rosenblatt in 1958
- Inspired by real neurons



THE PERCEPTRON

- Assume data is binary
- Assume data is linearly separable:
    - there exist a hyperplane that perfectly divides the two classes

REFRESHER:

- Recall how to define a hyperplane:
    - a subspace whose dimension is one less than that of its ambient space
    - if x is d-dimensional:
        - $\langle \mathbf{x}, w \rangle + b = 0$

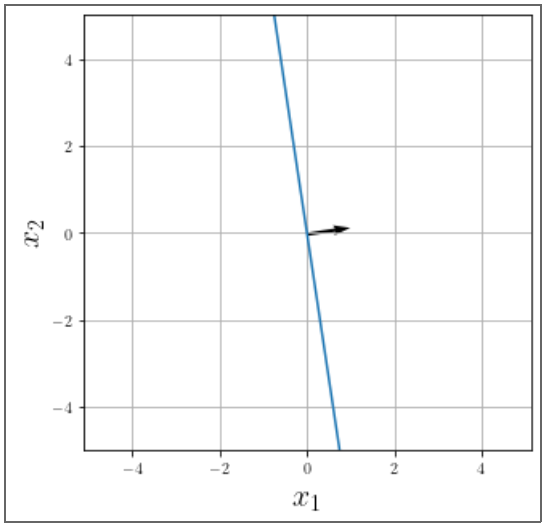
In [34]:

```
import matplotlib.pyplot as plt
%matplotlib inline
import numpy as np
plt.rcParams['text.usetex'] = True
import warnings
warnings.filterwarnings('ignore')
```

In [44]:

```
def plot_line(ax,xlims, w, do_norm=True):
    x1 = np.linspace(xlims[0],xlims[1],1000)
    x2_plot = (- w[2] - w[0]*x1)/w[1]  #  w[0]*x1 + w[1]*x2 + w[2] = 0
    ax.plot(x1,x2_plot)
    origin = x1[np.array(x1.shape[0]/2).astype(int)], x2_plot[int(x1.shape[0]/2)]
    nn = np.linalg.norm(w) if do_norm else 1
    ax.quiver(*origin, w[0]/nn,w[1]/nn, color='k',angles='xy', scale_units='xy', scale=1)
    ax.axis('equal')
    ax.axis([xlims[0],xlims[1],xlims[0],xlims[1]])
    ax.set_xlabel(r'$x_1$',fontsize=20); ax.set_ylabel(r'$x_2$',fontsize=20)
    ax.grid()

w = [4,0.6]
b = 0
f, ax = plt.subplots(figsize=(5,5))
plot_line(ax, [-5,5], [w[0],w[1],b])
```



THE PERCEPTRON

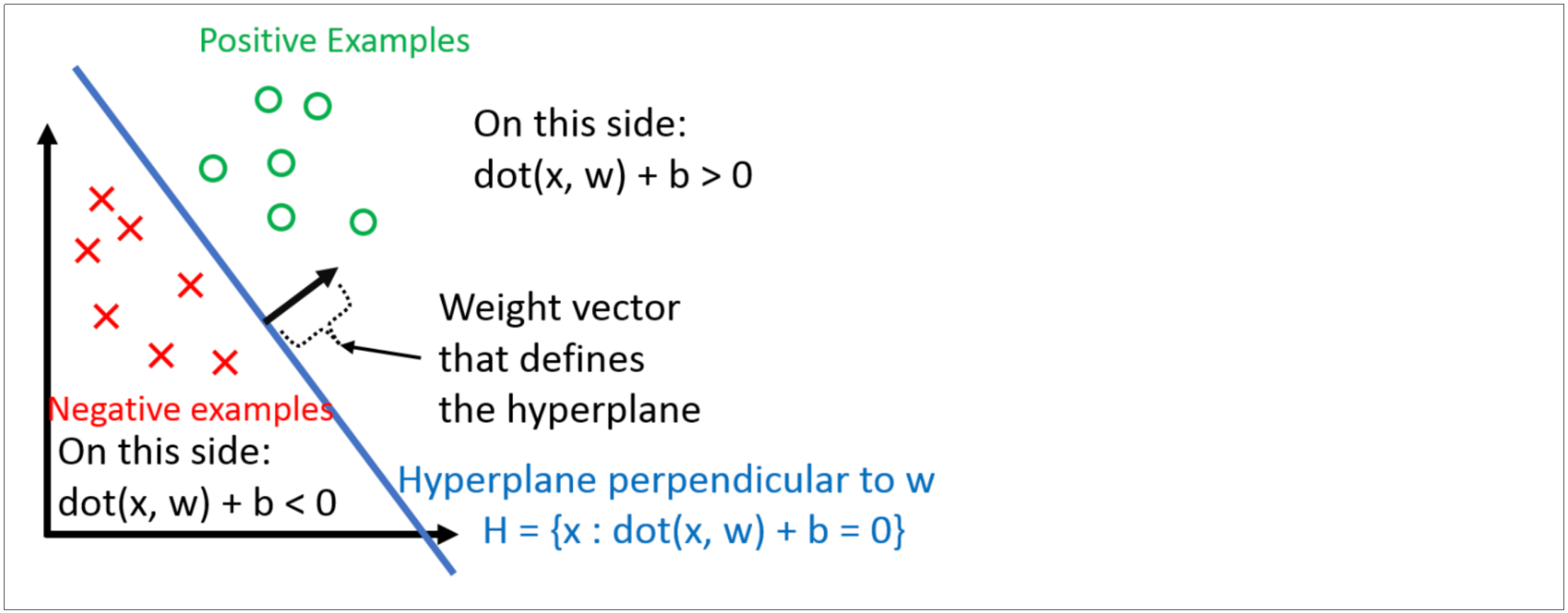
- Assume data is binary
- Assume data is linearly separable:
    - there exist a hyperplane that perfectly divides the two classes

$$\exists \mathbf{w}, b \quad \text{s.t.} \quad \forall (\mathbf{x}_i, y_i) \in D, \tag{1}$$

$$y_i(\mathbf{w}^\top \mathbf{x}_i + b) > 0 \tag{2}$$

$$\exists \mathbf{w}, b \quad \text{s.t.} \quad \forall (\mathbf{x}_i, y_i) \in D, \tag{3}$$

$$y_i(\mathbf{w}^\top \mathbf{x}_i + b) > 0 \tag{4}$$



source

What is a separable dataset?

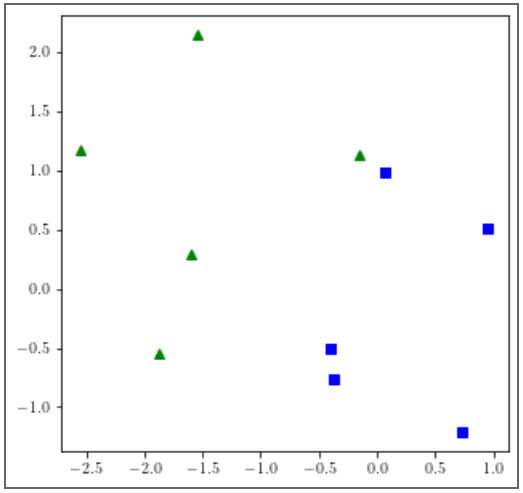
In [61]:

```
from sklearn import datasets

X, y = datasets.make_blobs(n_samples=10, centers=np.array([[-1,1],[1,-1]]),
                           n_features=2, center_box=(0, 10))
y[y==0] = -1

def plot_dataset(ax,X,y):
    ax.plot(X[:, 0][y == -1], X[:, 1][y == -1], 'g^')
    ax.plot(X[:, 0][y == 1], X[:, 1][y == 1], 'bs');

f, ax = plt.subplots(figsize=(5,5))
plot_dataset(ax,X,y)
```



SIMPLIFYING w AND b

- We can write $\mathbf{x}_i$ as:

$$\mathbf{x_i}' = \begin{bmatrix} \mathbf{x_i} \\ 1 \end{bmatrix} \tag{5}$$

- and incorporate b into $\mathbf{w}$:

$$\mathbf{w}' = \begin{bmatrix} \mathbf{w} \\ b \end{bmatrix} \tag{6}$$

- The same hyperplane is now defined by $\mathbf{w'}^\top \mathbf{x}' = 0$
- Why does this work?
- We will use $\mathbf{w}$ and $\mathbf{x}$ to refer to these vectors in the rest of the lecture.

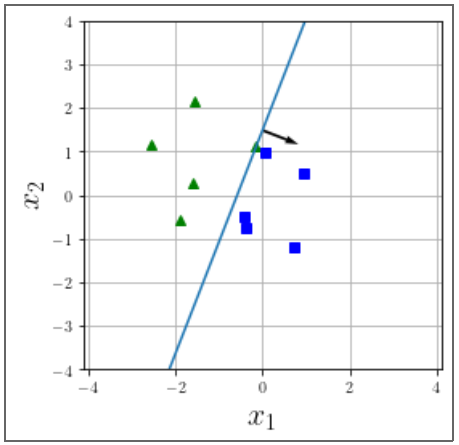
THE PERCEPTRON TRAINING ALGORITHM

```
Initialize w⃗ = 0⃗                          // Initialize w⃗. w⃗ = 0⃗ misclassifies everything.
while TRUE do                              // Keep looping
    m = 0                                  // Count the number of misclassifications, m
    for (x_i, y_i) ∈ D do                  // Loop over each (data, label) pair in the dataset, D
        if y_i(w⃗^T · x⃗_i) ≤ 0 then          // If the pair (x⃗_i, y_i) is misclassified
            w⃗ ← w⃗ + yx⃗                    // Update the weight vector w⃗
            m ← m + 1                      // Counter the number of misclassification
        end if
    end for
    if m = 0 then                          // If the most recent w⃗ gave 0 misclassifications
        break                              // Break out of the while-loop
    end if
end while                                  // Otherwise, keep looping!
```

In [62]:

```
def perceptron_train(X,y,MaxIter=20):
    w = np.zeros((X.shape[1]))
    for i in range(MaxIter):
        m = 0
        for (xi,yi) in zip(X,y):
            if yi*w.T.dot(xi)<=0:
                w = w + yi*xi
                m = m + 1
        if m==0:
            break
    return w

f,ax = plt.subplots(figsize=(4,4))
plot_dataset(ax,X,y)
Xprime = np.hstack([X,np.ones((X.shape[0],1))])
w = perceptron_train(Xprime,y)
plot_line(ax,[-4,4], w)
```



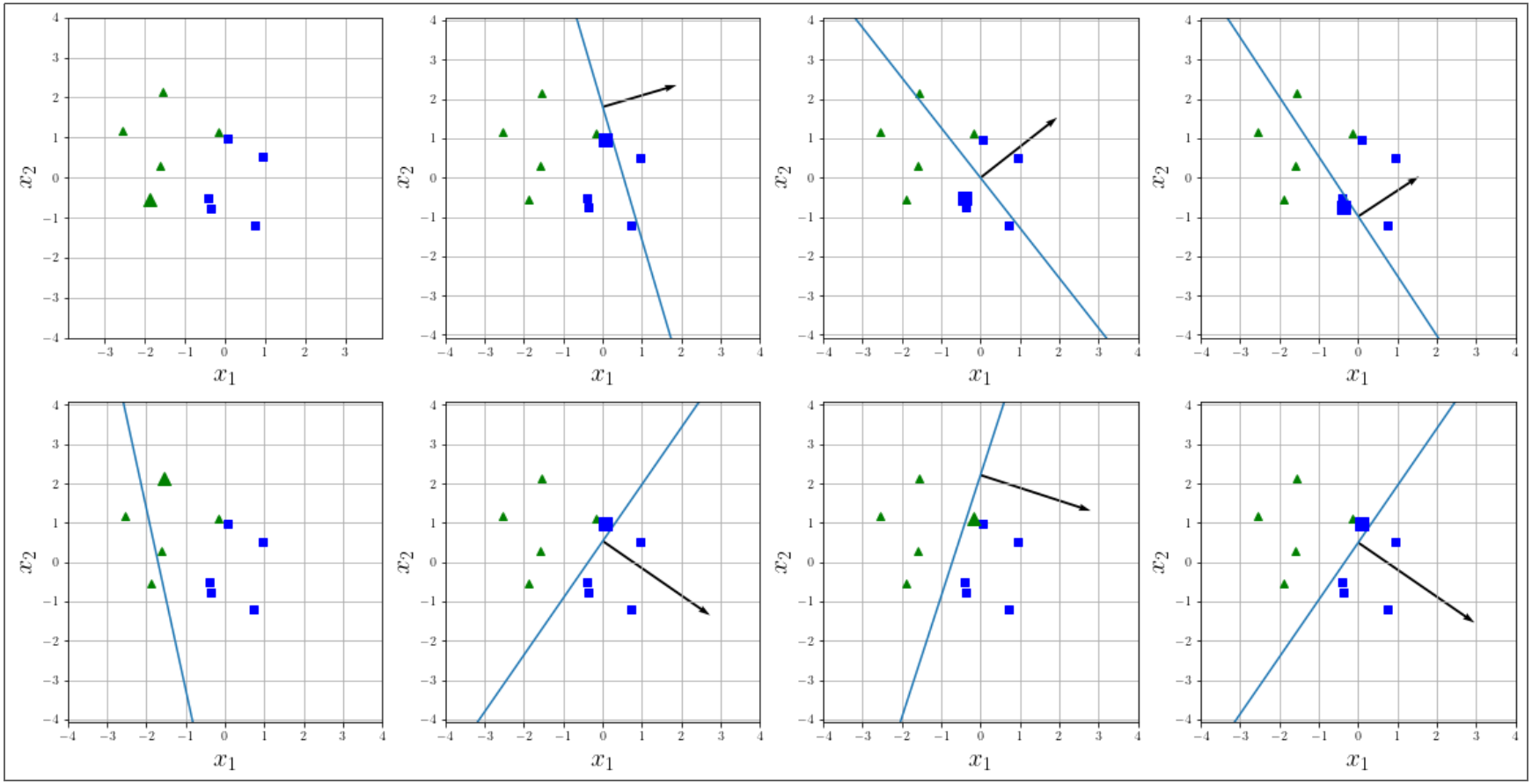
In [63]:

```
def perceptron_train_and_plot(axs,X,y,MaxIter=20):
    w = np.zeros((X.shape[1]))
    plot_dataset(axs[0],X,y)
    plot_line(axs[0],[-4,4], w)
    plt_cnt = 0
    for i in range(MaxIter):
        m = 0
        for (xi,yi) in zip(X,y):
            if yi*w.T.dot(xi)<=0:
                w = w + yi*xi
                m = m + 1
                plt_cnt = plt_cnt + 1
                try:
                    if yi == -1:
                        axs[plt_cnt-1].plot([xi[0]],[xi[1]],'g^',markersize=10)
                    else:
                        axs[plt_cnt-1].plot([xi[0]],[xi[1]],'bs',markersize=10)
                    plot_dataset(axs[plt_cnt],X,y)
                    plot_line(axs[plt_cnt],[-4,4], w, do_norm=False)
                except:
                    print('not enough subplots')
        if m==0:
            break
    return w
```

In [64]:

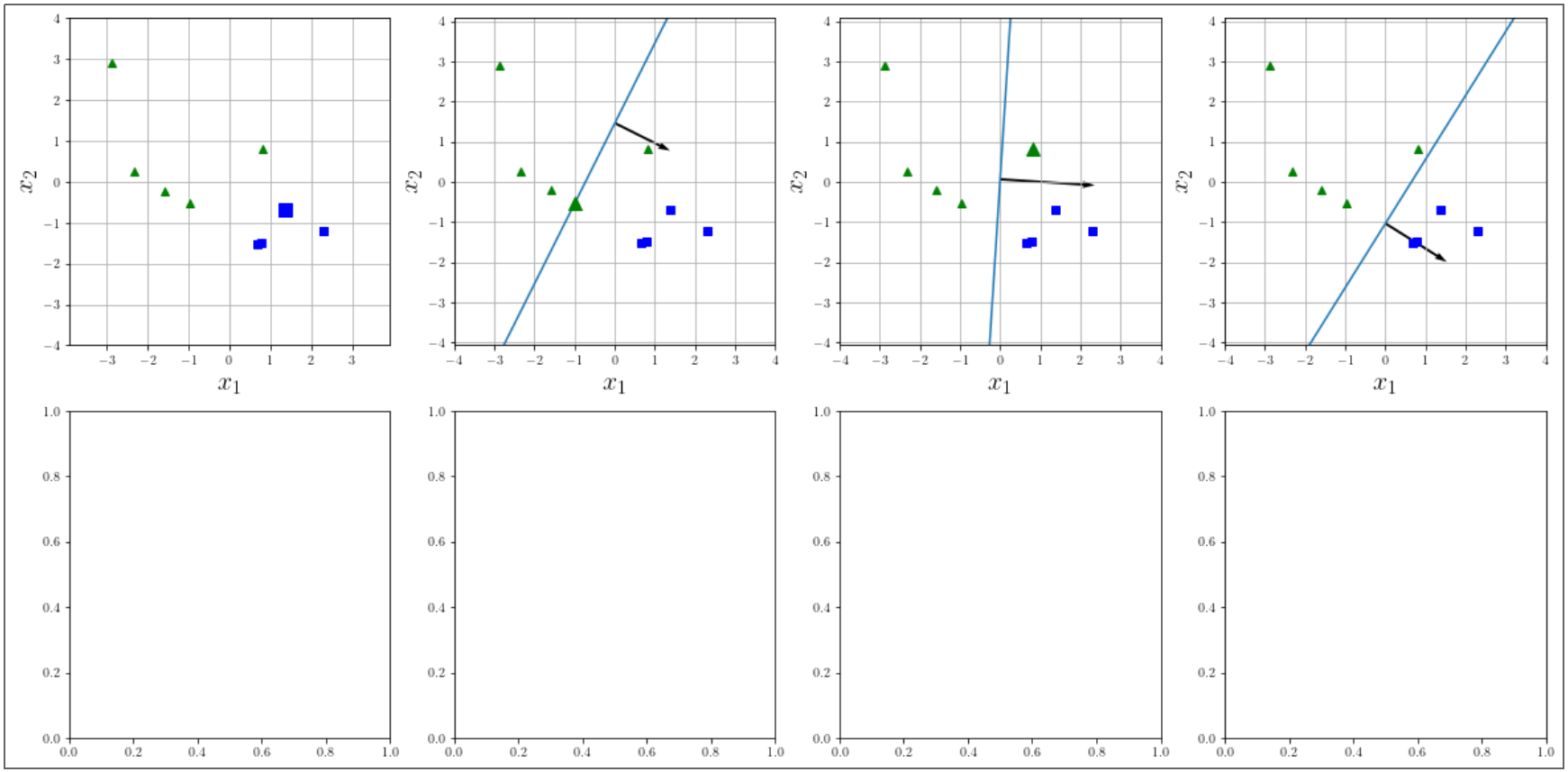
```
f,axs = plt.subplots(nrows=2,ncols=4,figsize=(20,10))
w = perceptron_train_and_plot(axs.reshape(-1),Xprime,y)
```

```
not enough subplots
not enough subplots
not enough subplots
not enough subplots
not enough subplots
```



In [68]:

```
X, y = datasets.make_blobs(n_samples=10, centers=np.array([[-1,1],[1,-1]]),
                           n_features=2, center_box=(0, 10))
y[y==0] = -1
Xprime = np.hstack([X,np.ones((X.shape[0],1))])
f,axs = plt.subplots(nrows=2,ncols=4,figsize=(20,10))
w = perceptron_train_and_plot(axs.reshape(-1),Xprime,y)
```



WHY DOES THIS WORK?

- what happens if you missclassify a positive example? (i.e. $\mathbf{w^k}^\top \mathbf{x_i} < 0$)

$$\mathbf{w^{k+1}} = \mathbf{w^k} + \mathbf{x_i}$$

$$\begin{aligned} \mathbf{w^{k+1}}^\top \mathbf{x_i} &= (\mathbf{w^k} + \mathbf{x_i})^\top \mathbf{x_i} && (7) \\ &= \mathbf{w^k}^\top \mathbf{x_i} + \mathbf{x_i}^\top \mathbf{x_i} && (8) \\ &> \mathbf{w^k}^\top \mathbf{x_i} && (9) \end{aligned}$$

- The prediction becomes more positive (vice versa for a negative example).
- Thus the boundary is getting closer to correctly classiying $\mathbf{x_i}$.

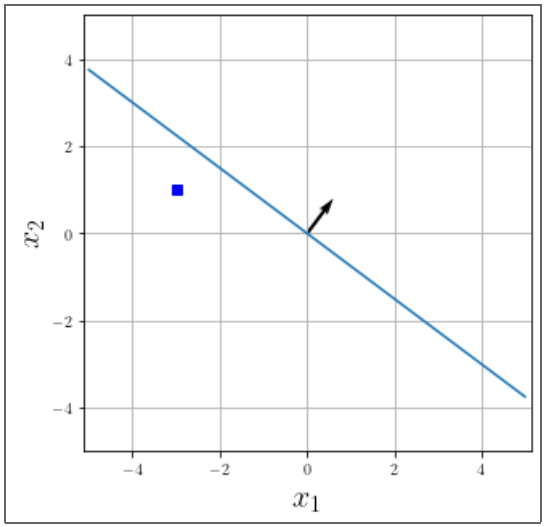
HOW MANY ITERATIONS ARE NEEDED WITH A DATASET WITH ONE EXAMPLE?

In [55]:

```
f, ax = plt.subplots(figsize=(5,5))
plot_line(ax, [-5,5], [3,4,0])
ax.plot([-3],[1],'bs')
```

Out[55]:

```
[<matplotlib.lines.Line2D at 0x11c503c88>]
```



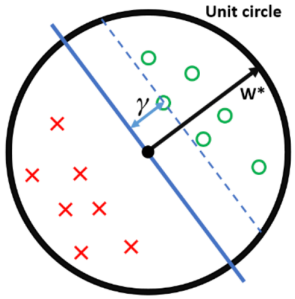
CONVERGENCE OF THE PERCEPTRON ALGORITHM

The perceptron algorithm converges in $\frac{1}{\gamma^2}$ updates if the data is linearly separable.

γ is the margin of the problem instance (defined on next slide).

NOTION OF MARGIN

- Assume there exists $\mathbf{w}^*$ such that $\forall(\mathbf{x_i}, y_i) \in D, \quad y_i(\mathbf{x_i}^\top \mathbf{w}^*) > 0$
- Also assume we rescale $\mathbf{w}^*$ and $\mathbf{x_i}$s such that:
    - $||\mathbf{w}^*|| = 1$ and $||\mathbf{x_i}|| \leq 1 \ \forall \mathbf{x_i}$ (how?)
- The margin γ of the hyperplane $\mathbf{w}^*$ is the minimum distance between one of the points and the hyperplane:
    - $\gamma = \min_{(\mathbf{x_i}, y_i) \in \mathcal{D}} |\mathbf{x_i}^\top \mathbf{w}^*|$ (since $\mathbf{w}^*$ is unit norm)



source

THEOREM

- Given:
    - All $\mathbf{x}_i$s are within the unit sphere
    - There exists a separating hyperplane $\mathbf{w}^*$, with $||\mathbf{w}^*|| = 1$
    - γ is the margin of hyperplane $\mathbf{w}^*$
- If all of the above holds, then the Perceptron algorithm makes at most $\frac{1}{\gamma^2}$ mistakes.
- Would we want a large margin or a small margin?
- What types of datasets will converge quickly?

PROOF

source

Keeping what we defined above, consider the effect of an update ($\mathbf{w}$ becomes $\mathbf{w} + y\mathbf{x}$) on the two terms $\mathbf{w}^\top \mathbf{w}^*$ and $\mathbf{w}^\top \mathbf{w}$. We will use two facts:

- $y(\mathbf{x}^\top \mathbf{w}) \leq 0$: This holds because $\mathbf{x}$ is misclassified by $\mathbf{w}$ - otherwise we wouldn't make the update.
- $y(\mathbf{x}^\top \mathbf{w}^*) > 0$: This holds because $\mathbf{w}^*$ is a separating hyper-plane and classifies all points correctly.

1. Consider the effect of an update on $\mathbf{w}^\top \mathbf{w}^*$:

$$(\mathbf{w} + y\mathbf{x})^\top \mathbf{w}^* = \mathbf{w}^\top \mathbf{w}^* + y(\mathbf{x}^\top \mathbf{w}^*) \geq \mathbf{w}^\top \mathbf{w}^* + \gamma$$

The inequality follows from the fact that, for $\mathbf{w}^*$, the distance from the hyperplane defined by $\mathbf{w}^*$ to $\mathbf{x}$ must be at least γ (i.e. $y(\mathbf{x}^\top \mathbf{w}^*) = |\mathbf{x}^\top \mathbf{w}^*| \geq \gamma$).

This means that for each update, $\mathbf{w}^\top \mathbf{w}^*$ grows by **at least** γ.

2. Consider the effect of an update on $\mathbf{w}^\top \mathbf{w}$:

$$(\mathbf{w} + y\mathbf{x})^\top (\mathbf{w} + y\mathbf{x}) = \mathbf{w}^\top \mathbf{w} + \underbrace{2y(\mathbf{w}^\top \mathbf{x})}_{<0} + \underbrace{y^2(\mathbf{x}^\top \mathbf{x})}_{0 \leq \ \leq 1} \leq \mathbf{w}^\top \mathbf{w} + 1$$

The inequality follows from the fact that

- $2y(\mathbf{w}^\top \mathbf{x}) < 0$ as we had to make an update, meaning $\mathbf{x}$ was misclassified
- $0 \leq y^2(\mathbf{x}^\top \mathbf{x}) \leq 1$ as $y^2 = 1$ and all $\mathbf{x}^\top \mathbf{x} \leq 1$ (because $\|\mathbf{x}\| \leq 1$).

3. Now we know that after M updates the following two inequalities must hold:

   (1) $\mathbf{w}^\top \mathbf{w}^* \geq M\gamma$

   (2) $\mathbf{w}^\top \mathbf{w} \leq M$.

   We can then complete the proof:

$$
\begin{aligned}
M\gamma &\leq \mathbf{w}^\top \mathbf{w}^* && \text{By (1)} \\
&= \|\mathbf{w}\| \cos(\theta) && \text{by definition of inner-product, where } \theta \text{ is the angle between } \mathbf{w} \text{ and } \mathbf{w}^*. \\
&\leq ||\mathbf{w}|| && \text{by definition of cos, we must have } \cos(\theta) \leq 1. \\
&= \sqrt{\mathbf{w}^\top \mathbf{w}} && \text{by definition of } \|\mathbf{w}\| \\
&\leq \sqrt{M} && \text{By (2)} \\
& \\
&\Rightarrow M\gamma \leq \sqrt{M} \\
&\Rightarrow M^2\gamma^2 \leq M \\
&\Rightarrow M \leq \frac{1}{\gamma^2} && \text{And hence, the number of updates } M \text{ is bounded from above by a constant.}
\end{aligned}
$$

- What happens if the data is not separable?
- Does the order matter?