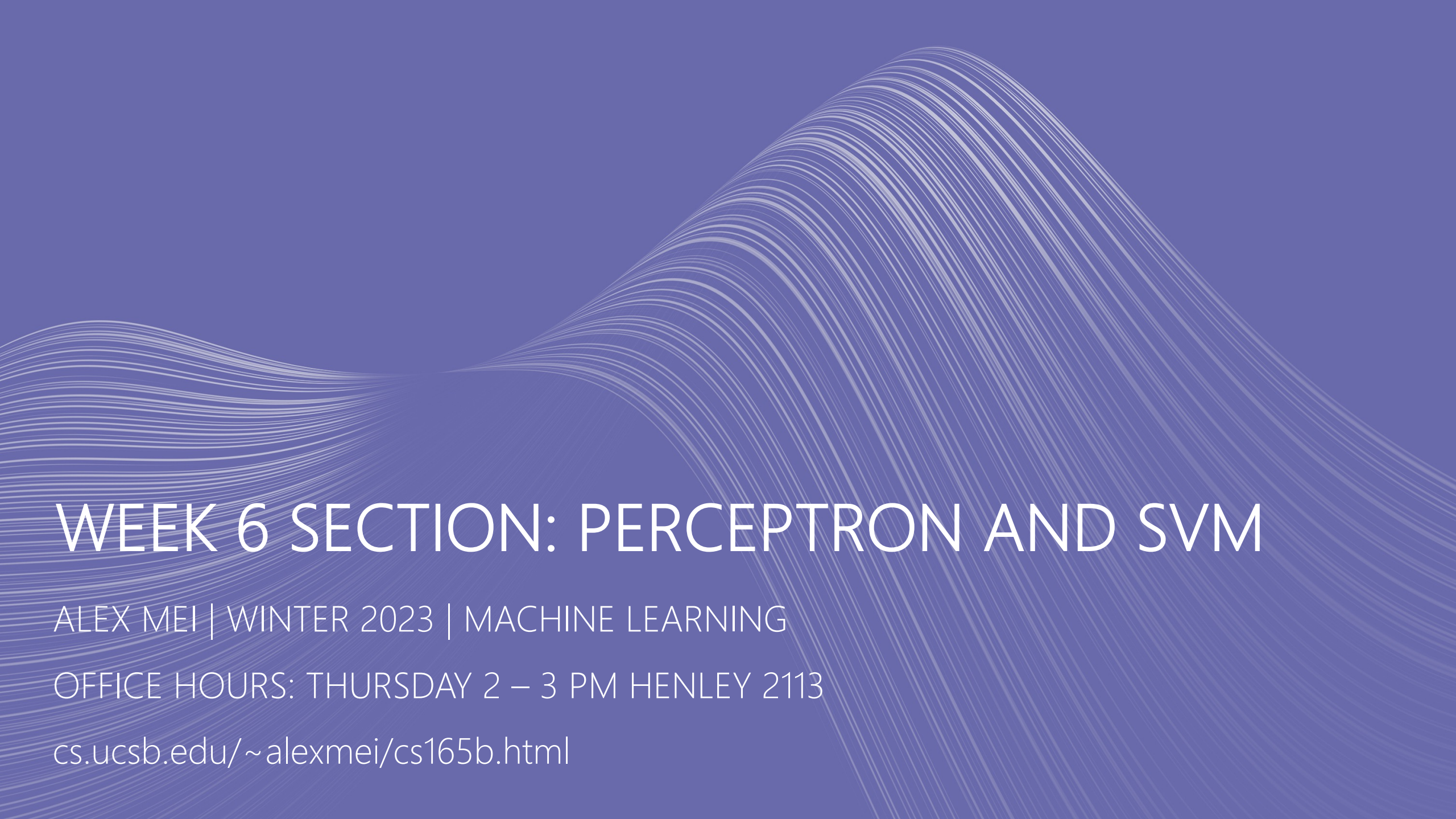


WEEK 6 SECTION: PERCEPTRON AND SVM

ALEX MEI | WINTER 2023 | MACHINE LEARNING

OFFICE HOURS: THURSDAY 2 – 3 PM HENLEY 2113

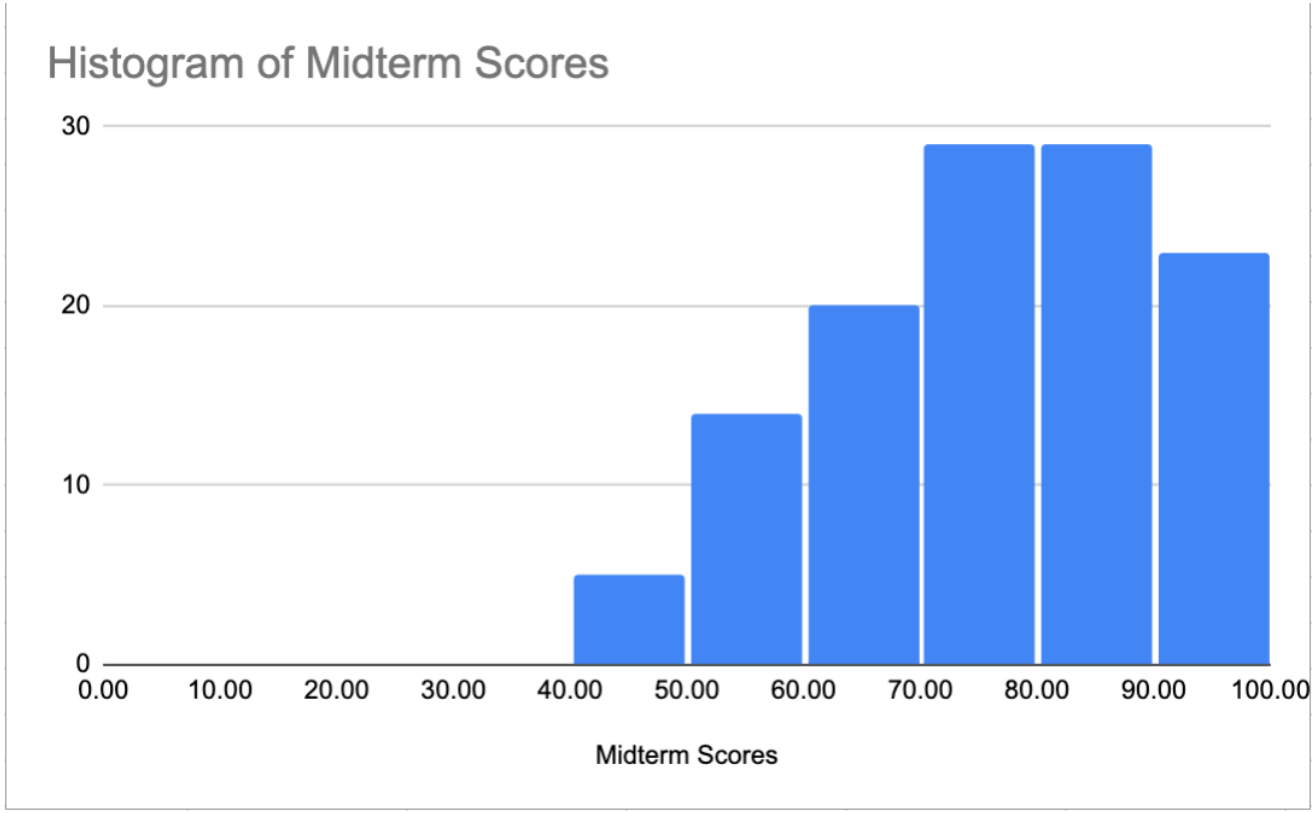
cs.ucsb.edu/~alexmei/cs165b.html

HW3 Overview - Decision Trees

* Please read Piazza before finalizing homework submission to ensure you don't lose points
* **Entropy:** amount of randomness in the data
* **LaPlace Correction:** add one smoothing to each class
* **Cardinality of Hypothesis Space:** number of unique decision trees
* **Stopping Conditions:** scenarios in which the tree does not need to or cannot be further grown
* **Programming Assignment:** should be a concise solution!

Midterm Statistics

* Mean: 76.02 | Median: 77.50 | Standard Deviation: 14.69 | Maximum: 100



Midterm Discussion

* **Live Section:** we will go over the midterm solutions; these cannot be recorded
* **Async Section:** perceptron and support vector machine from a geometric perspective
* Please check your scores on Canvas to see your initial performance
* If you are unhappy with your score, you should talk to the course staff during office hours
* Please do not panic! Usually, people are content with William's grade assignments ^w^
* **Regrades and Exam Viewing:** attend office hours in the next two weeks to discuss more
* To reduce chaos, scan the QR Code to sign up for a slot during office hours
* We will be limiting office hours to 3 students each per 10 minute session
* You must bring your access card or an ID to view your exam

Setting: The Restaurant Rating Problem

* **Input:** features (food taste, service quality, etc.) – $x \in R^d \equiv X$
* **Output:** binary label (good or bad restaurant) – $y \in \{-1, 1\} \equiv Y$
* **Given:** customers reviews – $D : \{(x_1, y_1), \ldots, (x_n, y_n)\}$
* **Goal:** learn relationship that models output for given input – $f : X \rightarrow Y$

If a restaurant is on the Michelin guide, you would expect some standards when it comes to plating. This sight of this plate is downright depressing... in Chinese cuisine either the dish isn't plated at all (traditional) or elevated (fusion), but here we get neither of those elements. On top of that, the dish is served with half raw steamed broccoli, which again is neither traditional nor fusion Chinese cuisine. The baked cod itself is cooked well, but needs to be flavored much more aggressively... I think

Is this drink not the cutest thing ever!? The cream art for this drink is first class and the details of the art is just so meticulous and well executed. Not only is this almost too beautiful to drink, it tastes equally amazing! I opted for a milk tea base which acts as a nice complement to the sweet and slightly bitter matcha cream, and overall I am super impressed by the technical execution of this dish as it is just a phenomenal dish all around!!

A Simple Learning Model

* Good Restaurant if $\sum_{i=1}^{d} w_i x_i > threshold$

* Bad Restaurant if $\sum_{i=1}^{d} w_i x_i < threshold$

> Q: what does a larger versus smaller weight w_i indicate about feature x_i?

A Simple Learning Model

* Good Restaurant if $\sum_{i=1}^{d} w_i x_i > threshold$

* Bad Restaurant if $\sum_{i=1}^{d} w_i x_i < threshold$

> Q: what does a larger versus smaller weight w_i indicate about feature x_i?

* Rewrite as $sign(\sum_{i=0}^{d} w_i x_i) = sign(w^T x);\ x_0 = 1$

> Q: how does the rewritten version integrate the threshold for the decision boundary?

* Notice that this classifier is simply a perceptron!

The Perceptron Learning Algorithm

Idea: start with an arbitrary weight vector and try to improve it

1. $\mathbf{w}(1) = 0$
2. for iteration $t = 1, 2, 3, \ldots$
3. the weight vector is $\mathbf{w}(t)$
4. From $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)$ pick any misclassified sample.
5. Call the misclassified example $(\mathbf{x}_*, y_*)$, i.e.,

$$\text{sign}(\mathbf{w}(t)^T \mathbf{x}_*) \neq y_*$$

6. Update the weight:

$$\mathbf{w}(t+1) = \mathbf{w}(t) + y_* \mathbf{x}_*$$

7. $t \leftarrow t + 1$

Does the Perceptron Update Step Make Sense?

* Identify a misclassified example (x_*, y_*) and update $w(t+1) = w(t) + y_* x_*$

> Q: does the dimensions of the update step align?

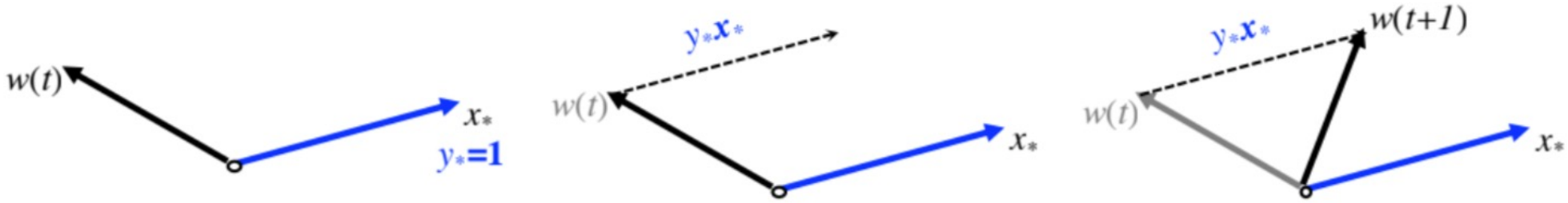
Does the Perceptron Update Step Make Sense?

* Identify a misclassified example (x_*, y_*) and update $w(t+1) = w(t) + y_* x_*$

> Q: does the dimensions of the update step align?

* Recall that $\frac{w^T x}{||w|| ||x||} = \cos(w, x)$ – positive when angle is less than 90 degrees

* Let's look at the geometric interpretation for a misclassified positive example



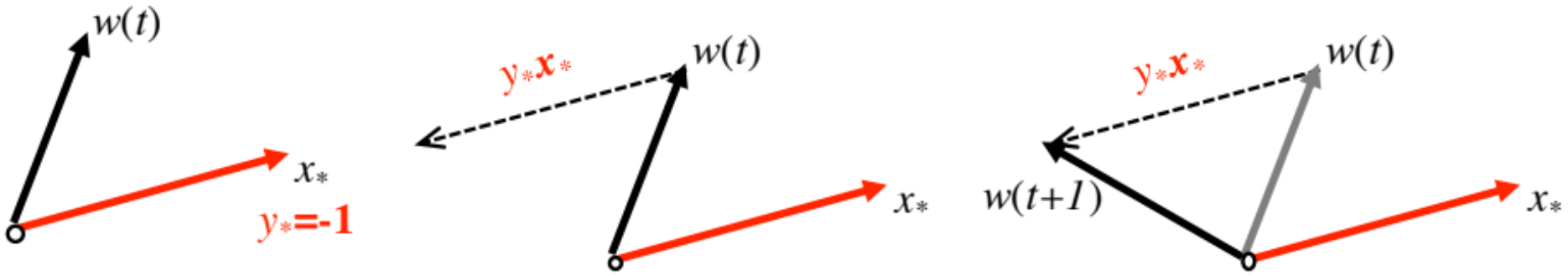
Does the Perceptron Update Step Make Sense?

* Identify a misclassified example (x_*, y_*) and update $w(t+1) = w(t) + y_* x_*$

> Q: does the dimensions of the update step align?

* Recall that $\frac{w^T x}{||w|| ||x||} = \cos(w, x)$ – positive when angle is less than 90 degrees

* Let's look at the geometric interpretation for a misclassified negative example



Convergence of a Perceptron

* **Theorem:** if the input data is linearly separable, then PLA will find a satisfying solution in a finite number of steps.

> **At Home Exercise:** prove this theorem using the Cauchy-Schwartz inequality

Programming Exercise – Perceptron Convergence

Let's explore the convergence of a perceptron empirically.

(1) Implement the Perceptron Learning Algorithm in Google Colab.

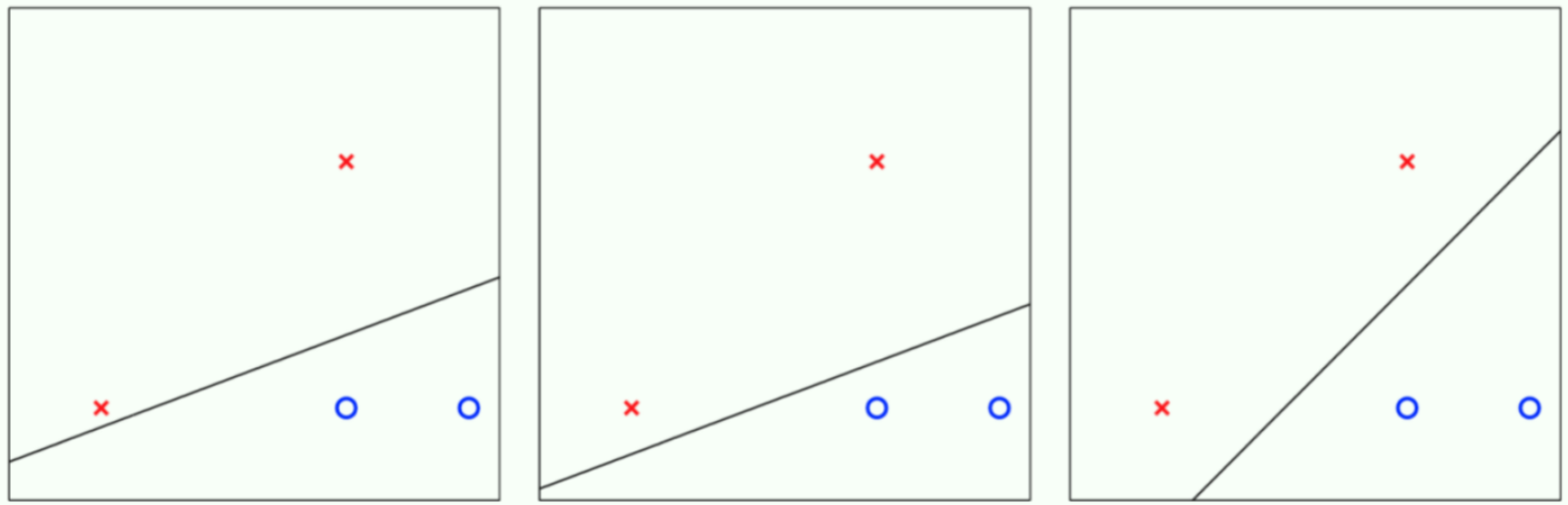
(2) Generate a linearly separable data set of size 20. Plot the examples as well as the target function f on a plane. Be sure to mark the examples from different classes differently and add labels to the axes of the plot.

(3) Run PLA on the dataset above. Report the number of updates needed for convergence. Add the final hypothesis g to the figure from (2) and comment on whether f is close to g.

(4) Repeat everything from (3) with new randomly generated datasets of size 20, 100, and 1000 respectively and compare your results.

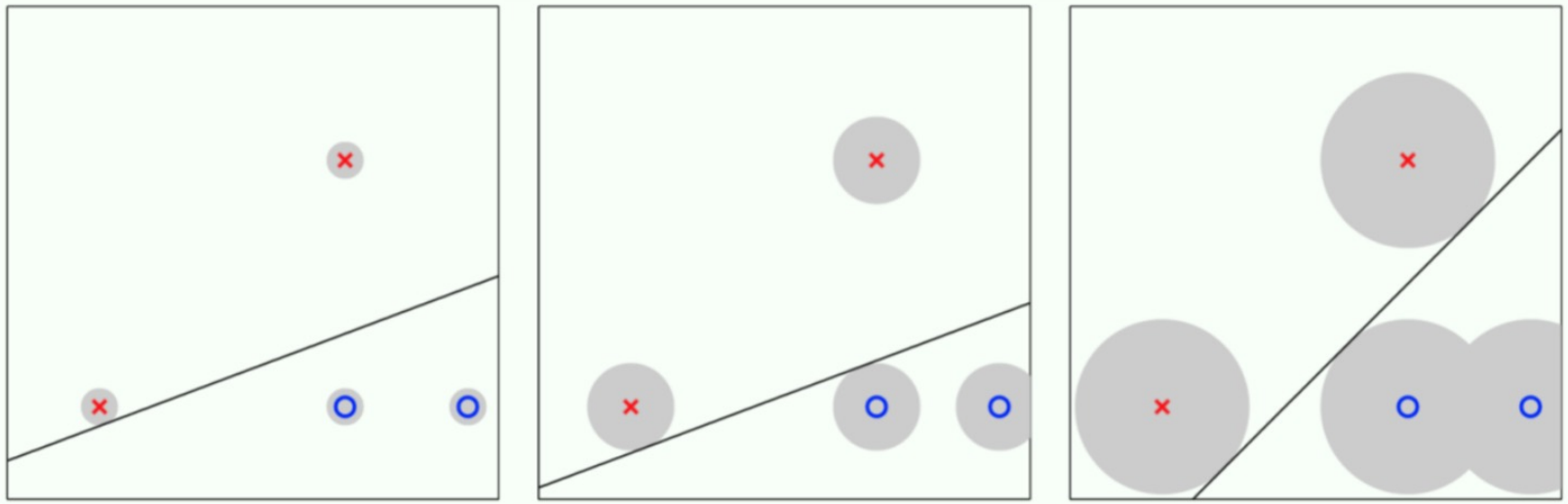
From Perceptron to SVM

> Q: given the following linear separators, which one is optimal?



From Perceptron to SVM

> Q: given the following linear separators, which one is optimal?



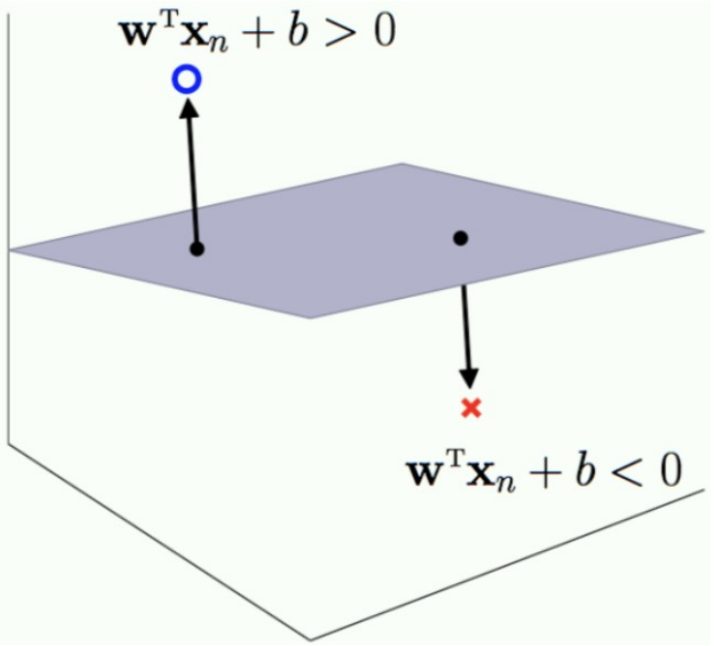
* A: pick the model most robust to noise

Defining Support Vectors

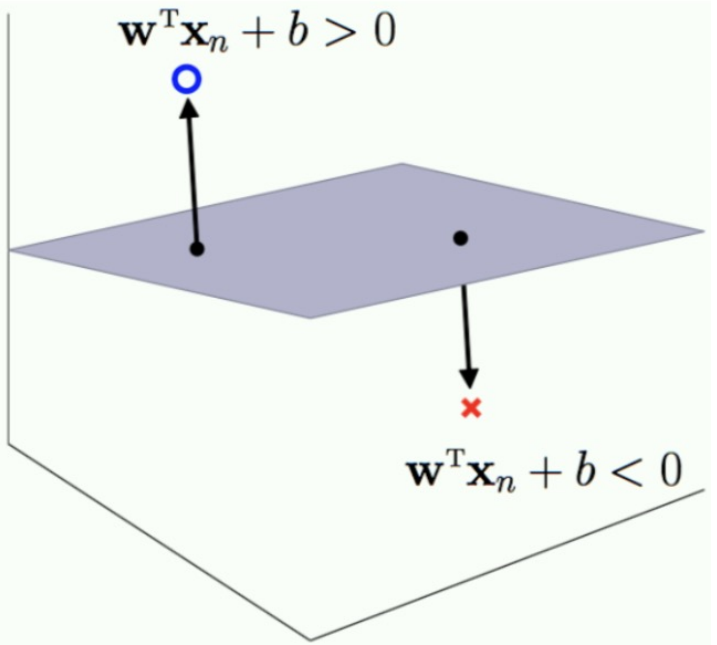
* Hyperplane $h = (b, w)$ separates the data means

$$y_i(w^T x_i + b) > 0$$

> Q: is the distance of a point to a hyperplane is affected by scale?



Defining Support Vectors



* Hyperplane $h = (b, w)$ separates the data means:

$$y_i(w^T x_i + b) > 0$$

> Q: is the distance of a point to a hyperplane is affected by scale?

* Add additional constraint on the scale of the hyperplane:

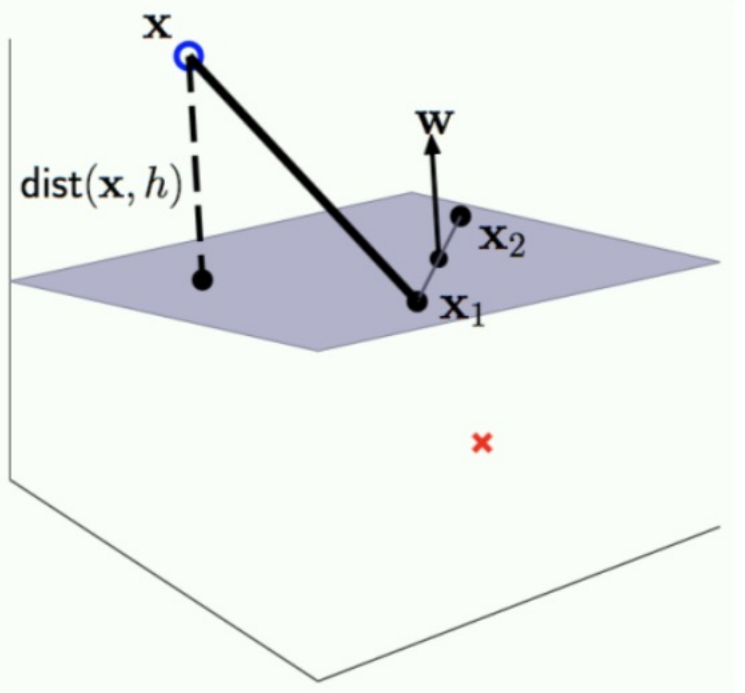
$$\min_i y_i(w^T x_i + b) = 1$$

* Support Vectors: minima points that uniquely define the hyperplane

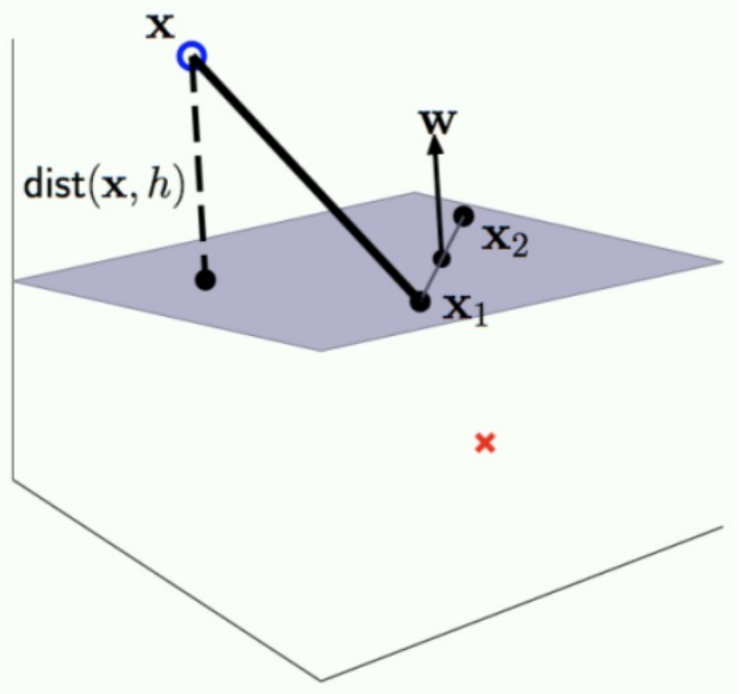
Distance to a Hyperplane

* w is normal to the hyperplane because…

$$\boldsymbol{w^T(x_2 - x_1) = w^T x_2 - w^T x_1 = -b + b = 0}$$



Distance to a Hyperplane



* w is normal to the hyperplane because…

$$\boldsymbol{w^T(x_2 - x_1) = w^T x_2 - w^T x_1 = -b + b = 0}$$

* $\boldsymbol{x'}$ is the orthogonal projection of $\boldsymbol{x}$ to the hyperplane

* Distance to the hyperplane is given by $\boldsymbol{x - x'}$ projected onto $\boldsymbol{w}$

* Recall a projection is defined by $proj_b(a) = \frac{a^T b}{||b||} = ||a||\cos(a, b)$

$$\boldsymbol{dist(x, h) = \frac{1}{||w||}\,|w^T x - w^T x'| = \frac{1}{||w||}\,|w^T x + b|}$$

Maximum Margin Classifier

* **Margin** (of a support vector): $\gamma(h) = 1/||w||$

> **Q:** What does a larger margin imply about model robustness?

Maximum Margin Classifier

* Margin (of a support vector): $\gamma(h) = 1/||w||$

> Q: What does a larger margin imply about model robustness?

* Maximum Margin Objective: $\min_{b,w} \frac{1}{2} w^T w$

subject to $\min_i y_i(w^T x_i + b) = 1$

* Equivalent Objective: $\min_{b,w} \frac{1}{2} w^T w$

subject to $y_i(w^T x_i + b) \geq 1$

Closed Form SVM Solution with Quadratic Programming

* Quadratic Solver: solves $\min_{u \in R^q} \frac{1}{2} u^T Q u + p^T u$ subject to $Au \geq c$

* $u^* = [\, b^* \,, w^*]^T = QP(Q, p, A, c)$

* For the primal, linear case of the hard margin SVM, use:

$$u = \begin{bmatrix} b \\ w \end{bmatrix}, Q = \begin{bmatrix} 0 & 0_d^T \\ 0_d & I_d \end{bmatrix}, p = 0_{d+1}, A = \begin{bmatrix} y_1 & y_1 x_1^T \\ \dots & \dots \\ y_n & y_n x_n^T \end{bmatrix}, c = 1_d$$

* SVM Inference: $g(x) = sign(w^T x + b)$

> At Home Exercise: reduce the hard margin objective to the quadratic programming problem

Programming Exercise – Hard Margin SVM

Let's explore the implementation of a primal, linear hard-margin SVM.

(1) Implement the Hard Margin SVM in Google Colab. Use the CVXOPT as the QP solver. [Note: the notation used in the tutorial is different from the previous slide. The (Q, p, A, c) notation corresponds to (P, q, G, h) in the tutorial.]

(2) Use random datasets (X, y) as input to your algorithm and vary the sample size and feature dimensionality. Investigate how time costs grow as you increase the sample size (while the dimensionality is fixed) and as you increase the dimensionality (while the sample size is fixed). Is your algorithm efficient?

Acknowledgement

Many slides are adapted from Professor Shiyu Chang at UCSB:

https://code-terminator.github.io/