CS 66: Machine Learning

Prof. Sara Mathieson

Spring 2019

Outline for April 24

- Announcements
- Midterm practice problems (Handout 19)
- Review SVMs
- Open time for questions

Outline for April 24

- Announcements
- Midterm practice problems (Handout 19)
- Review SVMs
- Open time for questions

Midterm 2

- 110 points possible, but will be graded out of 100

- 108 points on the exam itself, and 2 points for attending guest lecture on Friday
  - I will be sitting in the back and taking attendance
  - For 2 points: be on time and give Prof. Zucker your full attention

- Title: "Fun with SVMs and ordinal regression"

Midterm 2

- 8 pages with 4 small parts on each page
- Each part worth 2-4 points (no part worth a lot on its own)
- Keep explanations brief. If an explanation is not required, you may give one anyway to assist with partial credit
- Don’t need to simplify results (unless it’s necessary for answering the question)

CIFAR-10 Competition

- Congratulations to Dylan and Mikey!
  - 85.1%
- Runner up: Ankur and Ford
  - 83.81%
- Others above 70%: Ben and Daniel, Danielle and Nav, Kastan and Matthieu

Outline for April 24

- Announcements
- Midterm practice problems (Handout 19)
- Review SVMs
- Open time for questions

Handout 19, Question 1

- First compute weighted leaf labels

$$P(+ \mid \text{sun}) = \frac{\frac{1}{3}}{\frac{1}{3} + \frac{1}{8} + \frac{1}{8}} = \frac{4}{7} \geq 0.5 \quad \Rightarrow +$$

Handout 19, Question 1

- First compute weighted leaf labels

$$P(+ \mid \text{sun}) = \frac{\frac{1}{3}}{\frac{1}{3} + \frac{1}{8} + \frac{1}{8}} = \frac{4}{7} \geq 0.5 \quad \Rightarrow +$$

$$P(+ \mid \text{rain}) = \frac{\frac{1}{12}}{\frac{1}{12} + \frac{1}{6} + \frac{1}{6}} = \frac{1}{5} < 0.5 \quad \Rightarrow -$$

Handout 19, Question 1

- Based on these labels, we can say which training points are misclassified

$$\epsilon_t = \frac{1}{8} + \frac{1}{8} + \frac{1}{12} = \frac{1}{3}$$

Handout 19, Question 1

- Based on these labels, we can say which training points are misclassified

$$\epsilon_t = \frac{1}{8} + \frac{1}{8} + \frac{1}{12} = \frac{1}{3}$$

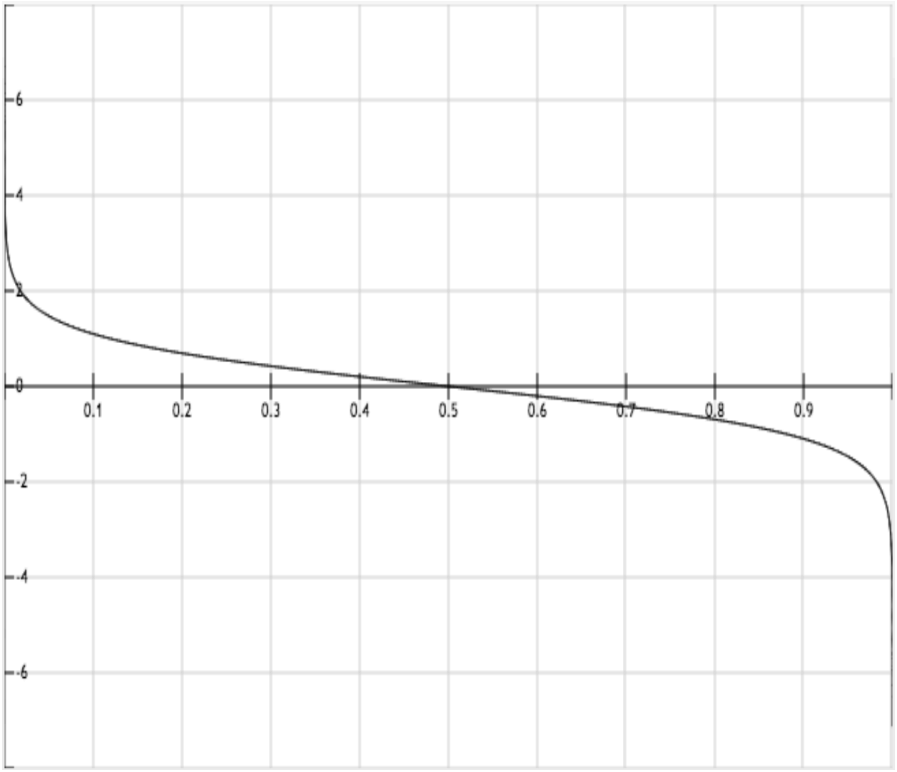
- Note if this was > 0.5, we should have chosen different leaf labels! So this "flipping" step should happen automatically
  - (exception for pathological cases)

Handout 19, Question 1

- Score function: $$\alpha_t = \frac{1}{2}\ln\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$$

- Fraction: accuracy/error

- As error -> 0, score becomes high
- As error -> ½, score goes to 0



Handout 19, Question 2

- r = 1/3, probability of one classifier being wrong
- T = 5, number of classifiers
- R = number of votes for the wrong class
- If R=3,4,5 then we will vote for the wrong class overall

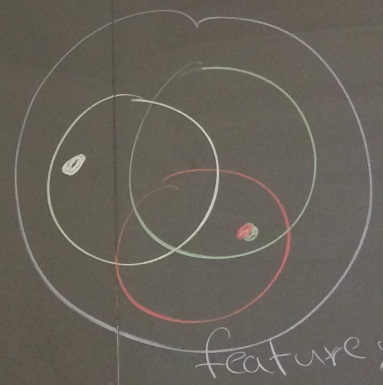
Handout 19

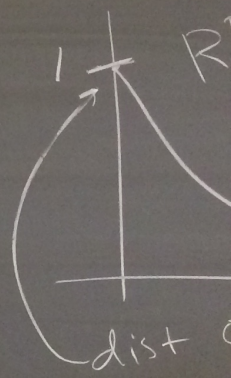
Q2

$$P(R=k) = \binom{T}{k} r^k (1-r)^{T-k}$$

$$+\begin{cases} P(R=5) = \binom{5}{5}\left(\frac{1}{3}\right)^5 \\ P(R=4) = \binom{5}{4}\left(\frac{1}{3}\right)^4\left(\frac{2}{3}\right) \\ P(R=3) = \binom{5}{3}\left(\frac{1}{3}\right)^3\left(\frac{2}{3}\right)^2 \end{cases}$$

$\rightarrow$ Overall Prob of being wrong: $\boxed{\approx 0.21}$





$y_i y_j \alpha_i \alpha_j (x_i x_j)$

$(x))$

$\vec{w} = \sum \alpha_i$

$0 = \sum \alpha_i y_i$

Handout 19, Question 2

- This analysis assumed classifiers were independent!

- What if they are not? How did Random Forests help us decorrelate classifiers?

Handout 19, Question 2

- This analysis assumed classifiers were independent!

- What if they are not? How did Random Forests help us decorrelate classifiers?

- Note about Bagging: choosing *n* with resampling actually does produce a very different dataset
  - As *n* increases, roughly 0.37 not chosen each time

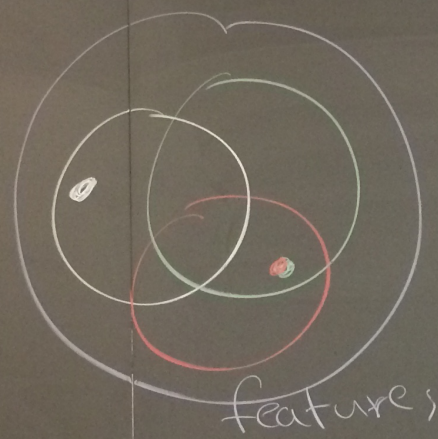
$$\binom{\ }{\ } r^k (1-r)^{T-k}$$

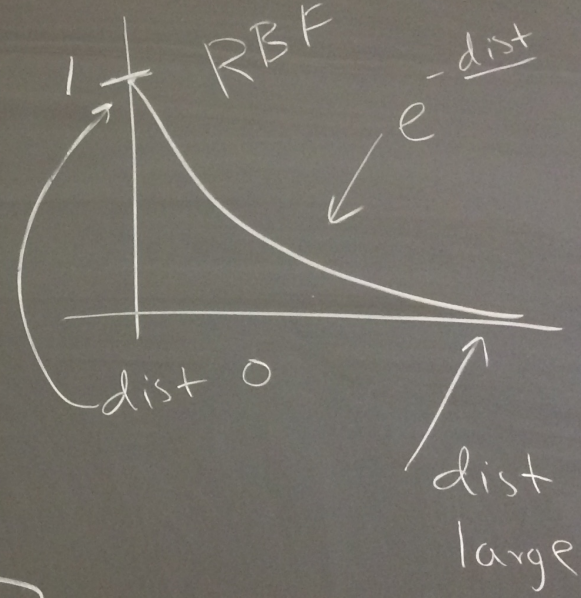
$$\binom{5}{5}\left(\frac{1}{3}\right)^5$$

$$\binom{5}{4}\left(\frac{1}{3}\right)^4\left(\frac{2}{3}\right)$$

$$\binom{5}{3}\left(\frac{1}{3}\right)^3\left(\frac{2}{3}\right)^2$$

Prob of being wrong: $\boxed{\approx 0.21}$





Handout 19, Question 3

- Generative vs. Discriminative
  - Generative models allow us to create simulated ("generated") new datasets in the style of our training data
  - Discriminative models create boundaries or distinctions between classes

Handout 19, Question 3

- Generative vs. Discriminative
  - Generative models allow us to create simulated ("generated") new datasets in the style of our training data
  - Discriminative models create boundaries or distinctions between classes

- Generative: only Naïve Bayes and GMM, although Neural Networks can be used in a generative framework

5

$\vec{x}_1$ & $\vec{x}_2$

3 datasets

$$\begin{bmatrix} 1 & 1 \\ 1 & 2 \\ 2 & 2 \end{bmatrix}$$

x_1	x_2	x_3
111	112	113
222	122	133
333	233	
123	223	

10

$$\binom{n+k-1}{k}$$

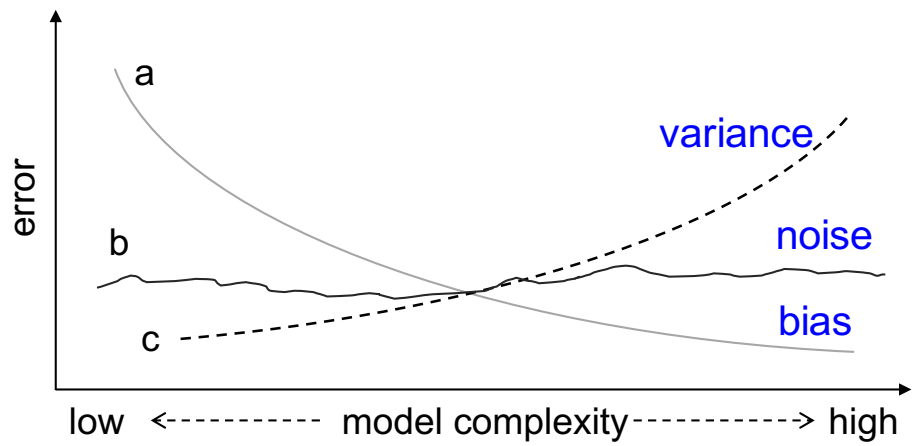
Outline for April 24

- Announcements
- Midterm practice problems (Handout 19)
- Review SVMs
- Open time for questions

Notecards (quick questions)

- How do we initialize bias? A: usually all zeros
- Bias/variance tradeoff
- Three sources of error:

$$\mathrm{E}\left[\left(y-\hat{f}(x)\right)^2\right] = \left(\mathrm{Bias}\left[\hat{f}(x)\right]\right)^2 + \mathrm{Var}\left[\hat{f}(x)\right] + \sigma^2$$



Equations and more info:
https://en.wikipedia.org/wiki/Bias%E2%80%93variance_tradeoff

Notecards (quick questions)

- Expectations are with respect to different choices of training data

Equations and more info:
https://en.wikipedia.org/wiki/Bias%E2%80%93variance_tradeoff

Notecards (quick questions)

- Expectations are with respect to different choices of training data
- High bias comes from choosing a model that cannot capture a real-world phenomenon

$$\mathrm{Bias}\left[\hat{f}(x)\right] = \mathrm{E}\left[\hat{f}(x)\right] - f(x)$$

Equations and more info:
https://en.wikipedia.org/wiki/Bias%E2%80%93variance_tradeoff

Notecards (quick questions)

- Expectations are with respect to different choices of training data
- High bias comes from choosing a model that cannot capture a real-world phenomenon

$$\mathrm{Bias}\left[\hat{f}\left(x\right)\right] = \mathrm{E}\left[\hat{f}\left(x\right)\right] - f(x)$$

- High variance comes from small changes in the training data having a large impact on the model

$$\mathrm{Var}\left[\hat{f}\left(x\right)\right] = \mathrm{E}[\hat{f}\left(x\right)^2] - \mathrm{E}[\hat{f}\left(x\right)]^2$$

Equations and more info:
https://en.wikipedia.org/wiki/Bias%E2%80%93variance_tradeoff

Perceptron Recap

$$\alpha = 0.2$$

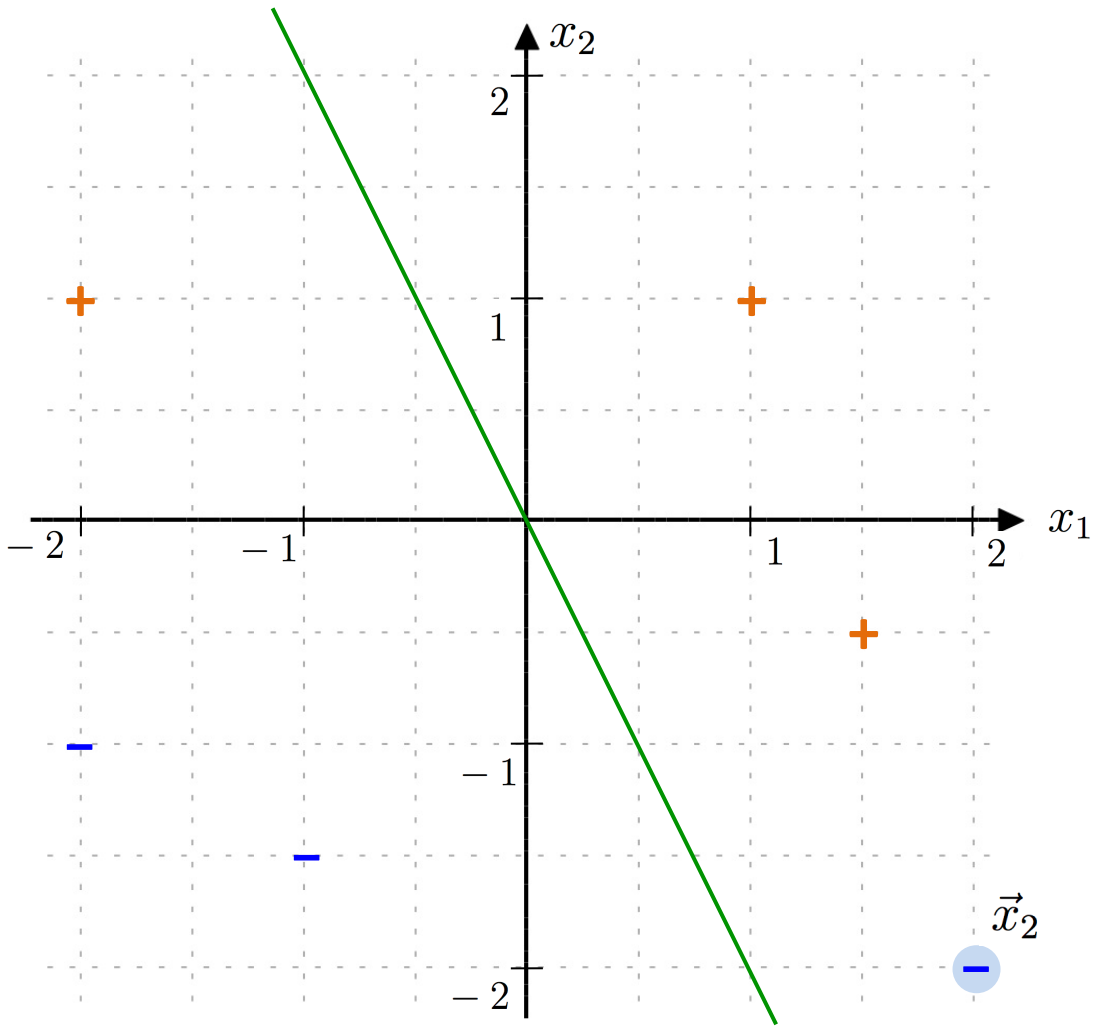
$$\vec{w} = \begin{bmatrix} 0 \\ 1 \\ 0.5 \end{bmatrix}$$

Round 2:

$$\vec{x}_2 = \begin{bmatrix} 1 \\ 2 \\ -2 \end{bmatrix}$$

$$\vec{w} \cdot \vec{x}_2 > 0$$

Incorrect classification



Example modified from Achim J. Lilienthal & Thorsteinn Rögnvaldsson

Perceptron Recap

$$\alpha = 0.2$$

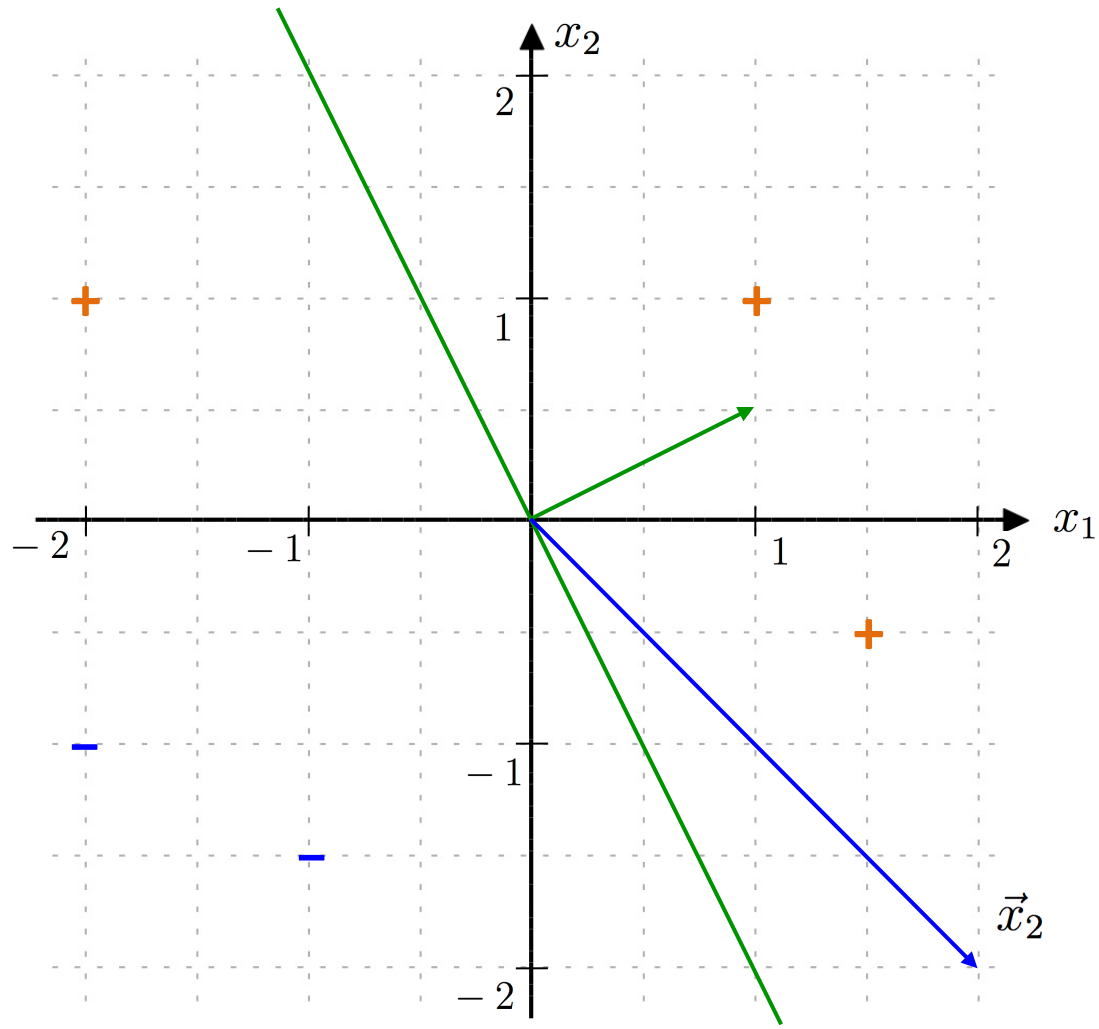
$$\vec{w} = \begin{bmatrix} 0 \\ 1 \\ 0.5 \end{bmatrix}$$

Round 2:

$$\vec{x}_2 = \begin{bmatrix} 1 \\ 2 \\ -2 \end{bmatrix}$$

$$\vec{w} \cdot \vec{x}_2 > 0$$

Incorrect classification



Example modified from Achim J. Lilienthal & Thorsteinn Rögnvaldsson

Perceptron Recap

$$\alpha = 0.2$$

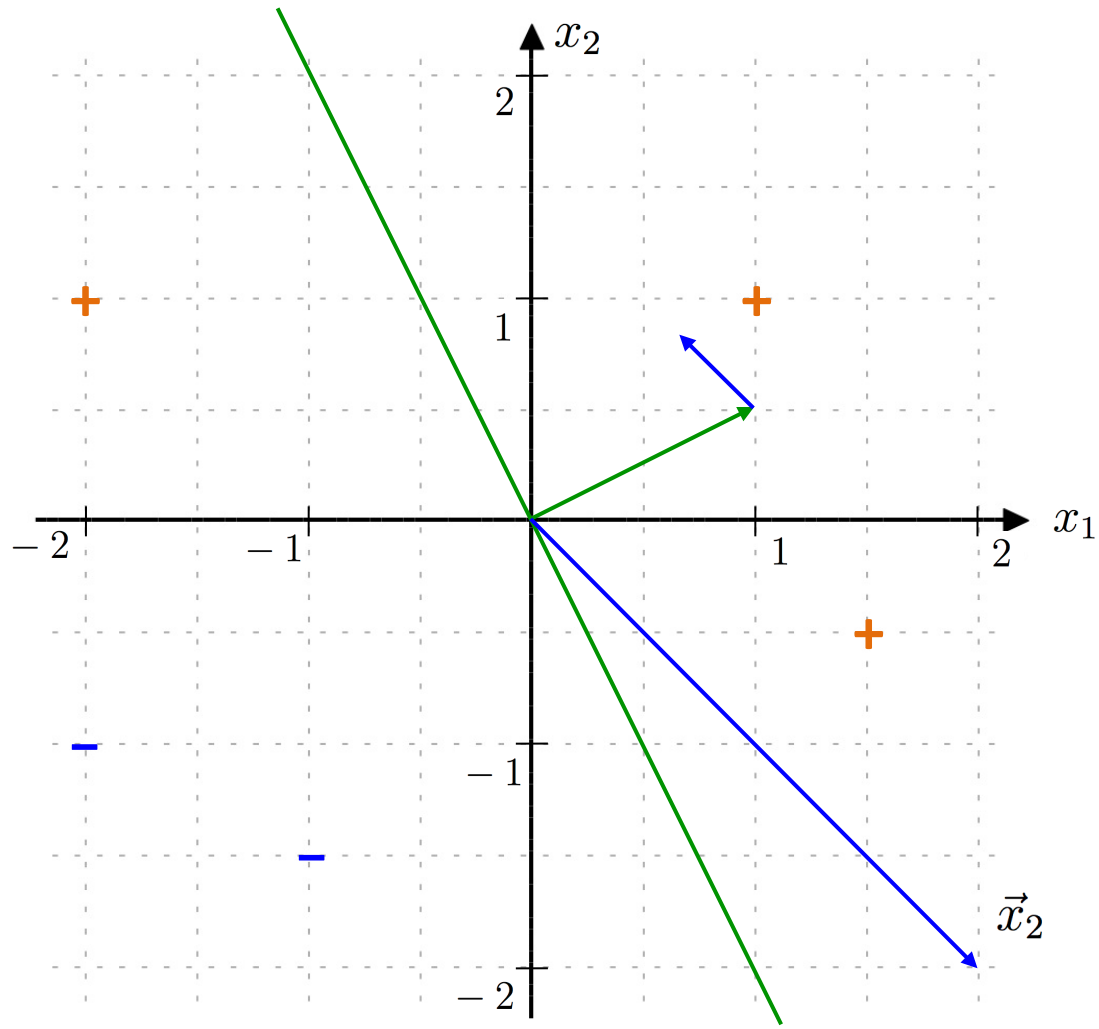
$$\vec{w} = \begin{bmatrix} 0 \\ 1 \\ 0.5 \end{bmatrix}$$

Round 2:

$$\vec{x}_2 = \begin{bmatrix} 1 \\ 2 \\ -2 \end{bmatrix}$$

$$\vec{w} \cdot \vec{x}_2 > 0$$

Incorrect classification
"Push" ***w*** away from negative point



Example modified from Achim J. Lilienthal & Thorsteinn Rögnvaldsson

Perceptron Recap

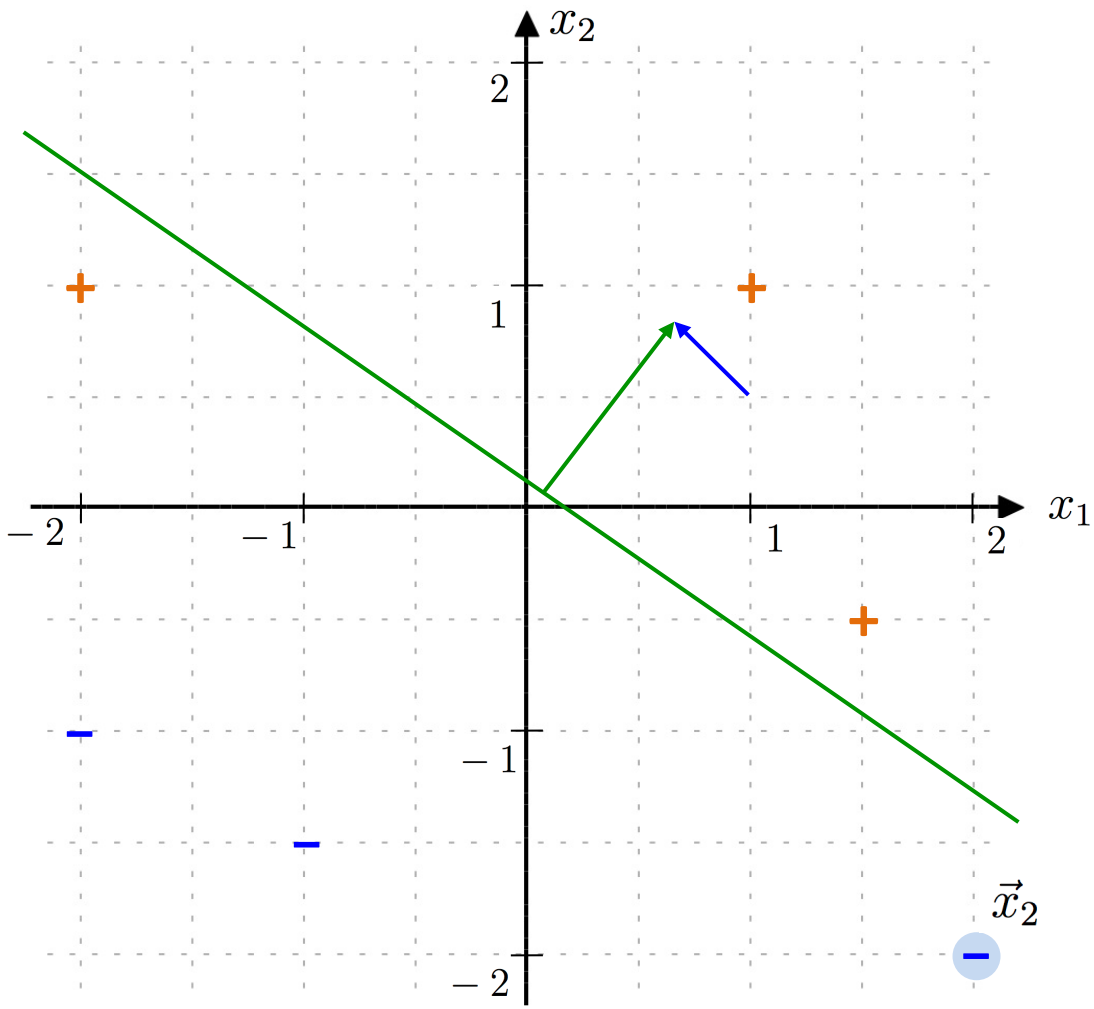
$$\alpha = 0.2$$

$$\vec{w} = \begin{bmatrix} 0 \\ 1 \\ 0.5 \end{bmatrix}$$

Round 2:

$$\vec{x}_2 = \begin{bmatrix} 1 \\ 2 \\ -2 \end{bmatrix}$$

$$\vec{w} \cdot \vec{x}_2 > 0$$



Incorrect classification
"Push" ***w*** away from negative point

Example modified from Achim J. Lilienthal & Thorsteinn Rögnvaldsson

Perceptron Recap

$$\alpha = 0.2$$

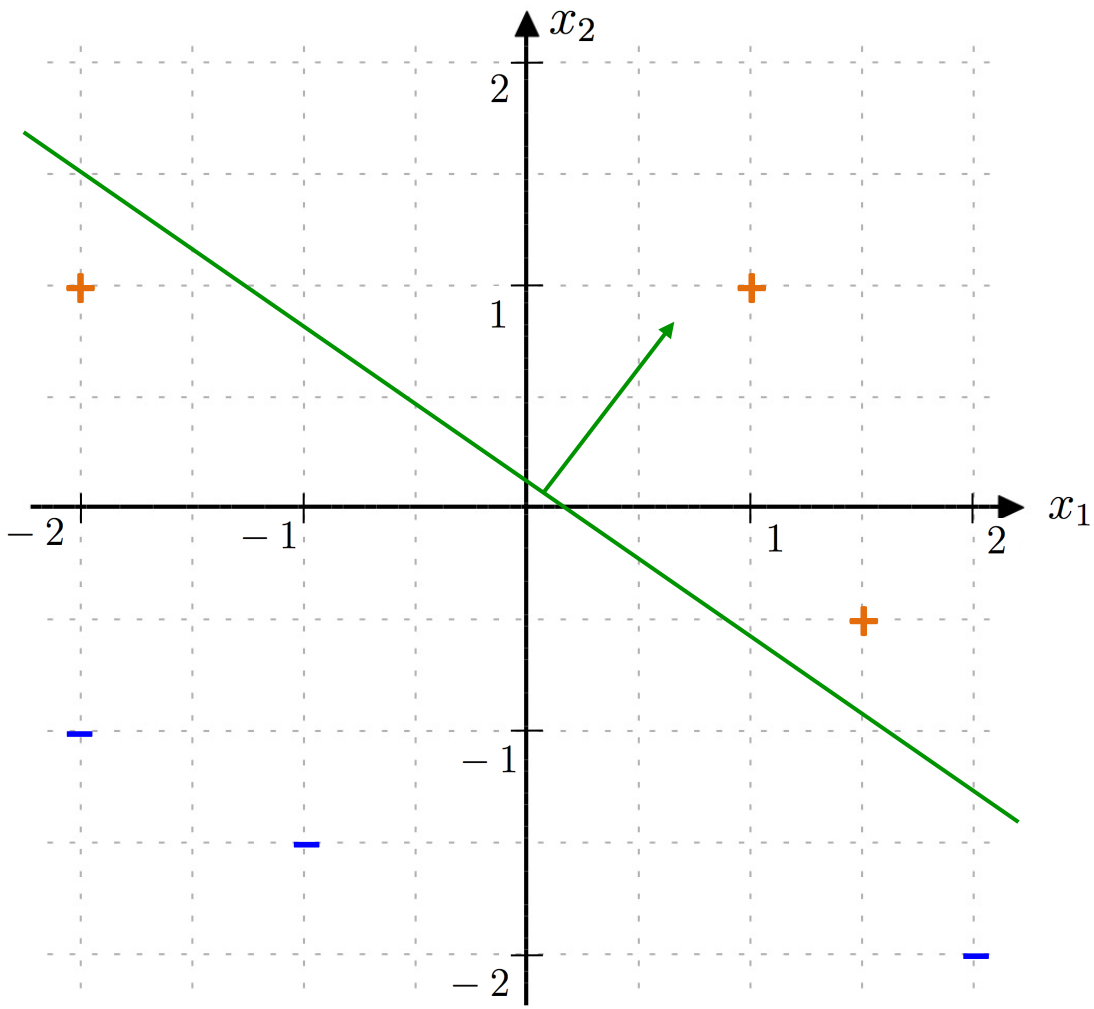
$$\vec{w} = \begin{bmatrix} 0 \\ 1 \\ 0.5 \end{bmatrix}$$

Round 2:

$$\vec{x}_2 = \begin{bmatrix} 1 \\ 2 \\ -2 \end{bmatrix}$$

$$\vec{w} \cdot \vec{x}_2 > 0$$

What is the new weight vector?



Example modified from Achim J. Lilienthal & Thorsteinn Rögnvaldsson

SVM flowchart

Goal: maximize the separation between positive and negative examples

SVM flowchart

Goal: maximize the separation between positive and negative examples

Create optimization problem: “maximizing the margin”

SVM flowchart

Goal: maximize the separation between positive and negative examples

→

Create optimization problem: "maximizing the margin"

→

Convert this "primal" optimization problem into the "dual"

SVM flowchart

Goal: maximize the separation between positive and negative examples

Create optimization problem: "maximizing the margin"

Convert this "primal" optimization problem into the "dual"

Training: solve dual optimization problem to find alpha value for each training example

SVM flowchart

Goal: maximize the separation between positive and negative examples

Create optimization problem: "maximizing the margin"

Convert this "primal" optimization problem into the "dual"

Training: solve dual optimization problem to find alpha value for each training example

Testing: use alpha values of support vectors to classify new points (don't explicitly calculate w)

SVM flowchart

Goal: maximize the separation between positive and negative examples

Create optimization problem: "maximizing the margin"

- Use relationship between functional & geo margin
- Put arbitrary constraint on functional margin
- Take multiplicative inverse to make max into min

Convert this "primal" optimization problem into the "dual"

Training: solve dual optimization problem to find alpha value for each training example

Testing: use alpha values of support vectors to classify new points (don't explicitly calculate w)

SVM flowchart

Goal: maximize the separation between positive and negative examples

Create optimization problem: "maximizing the margin"

- Use relationship between functional & geo margin
- Put arbitrary constraint on functional margin
- Take multiplicative inverse to make max into min

Convert this "primal" optimization problem into the "dual"

- Use Lagrange multipliers to incorporate constraints
- Use "min max" = "max min" (special case)
- Minimize with respect to w and b

Training: solve dual optimization problem to find alpha value for each training example

Testing: use alpha values of support vectors to classify new points (don't explicitly calculate w)

Functional and Geometric Margins

SVM classifier:
(same as perceptron)

$$h(\vec{x}) = \text{sign}\left(\vec{w} \cdot \vec{x} + b\right)$$

Functional and Geometric Margins

SVM classifier:
(same as perceptron)

$$h(\vec{x}) = \text{sign}\left(\vec{w} \cdot \vec{x} + b\right)$$

Functional Margin:

$$\hat{\gamma}_i = y_i(\vec{w} \cdot \vec{x}_i + b)$$

Functional and Geometric Margins

SVM classifier:
(same as perceptron)

$$h(\vec{x}) = \text{sign}\left(\vec{w} \cdot \vec{x} + b\right)$$

Functional Margin:

$$\hat{\gamma}_i = y_i(\vec{w} \cdot \vec{x}_i + b)$$

Geometric Margin:
(distance between example and hyperplane)

$$\gamma_i = y_i \left(\frac{\vec{w}}{\|\vec{w}\|} \cdot \vec{x}_i + \frac{b}{\|\vec{w}\|} \right)$$

Functional and Geometric Margins

SVM classifier:
(same as perceptron)

$$h(\vec{x}) = \text{sign}\left(\vec{w} \cdot \vec{x} + b\right)$$

Functional Margin:

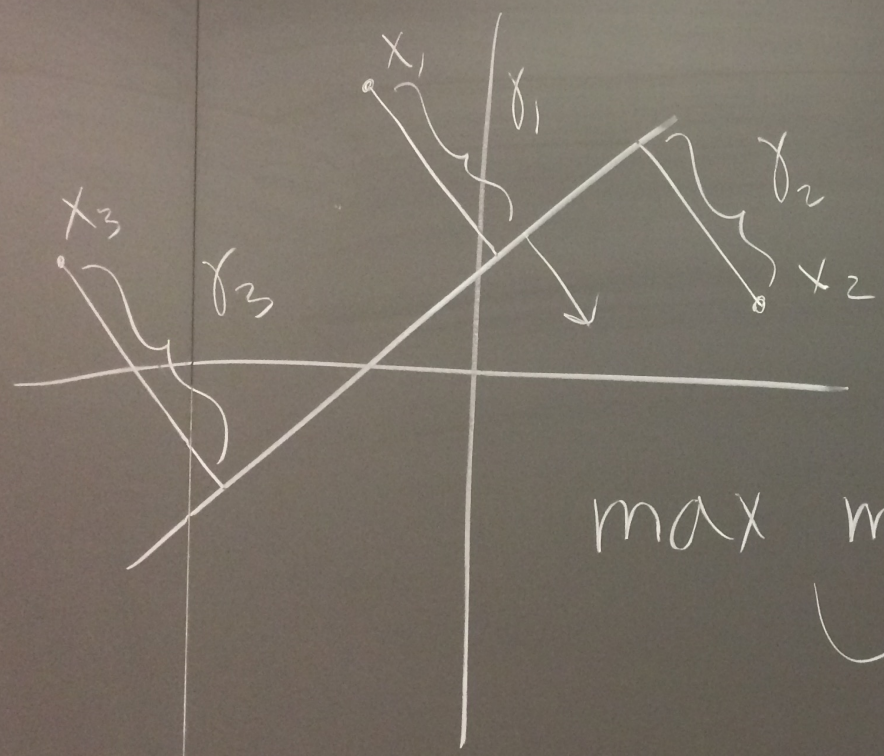
$$\hat{\gamma}_i = y_i(\vec{w} \cdot \vec{x}_i + b)$$

Geometric Margin:
(distance between example and hyperplane)

$$\gamma_i = y_i \left(\frac{\vec{w}}{\|\vec{w}\|} \cdot \vec{x}_i + \frac{b}{\|\vec{w}\|} \right)$$

Note:

$$\boxed{\gamma_i = \frac{\hat{\gamma}_i}{\|\vec{w}\|}}$$



$$\max \underbrace{\min_i \gamma_i}_{\gamma}$$

Optimization Problem: try 1

Goal: maximize the minimum distance between example and hyperplane

$$\gamma = \min_{i=1,\cdots,n} \gamma_i$$

Optimization Problem: try 1

Goal: maximize the minimum distance between example and hyperplane

$$\gamma = \min_{i=1,\cdots,n} \gamma_i$$

Formulation: optimize a function with respect to a constraint

$$\begin{aligned} \max_{\gamma,\vec{w},b} \quad & \gamma \\ \text{s.t.} \quad & y_i(\vec{w}\cdot\vec{x}_i + b) \geq \gamma, \quad i = 1,\cdots,n \\ \text{and} \quad & \|\vec{w}\| = 1 \end{aligned}$$

(force functional and geometric margin to be equal)

Optimization Problem: try 2

Idea: substitute functional margin
divided by magnitude of weight vector

$$\max_{\hat{\gamma},\vec{w},b} \quad \frac{\hat{\gamma}}{\|\vec{w}\|}$$
$$\text{s.t.} \quad y_i(\vec{w}\cdot\vec{x}_i + b) \geq \hat{\gamma}, \quad i = 1, \cdots, n$$

(gets rid of non-convex constraint)

Optimization Problem: try 3

Idea: put arbitrary constraint on functional margin

$$\boxed{\hat{\gamma} = 1}$$

$$\min_{\vec{w},b} \quad \frac{1}{2}\|\vec{w}\|^2$$

$$\text{s.t.} \quad y_i(\vec{w} \cdot \vec{x}_i + b) \geq 1, \quad i = 1, \cdots, n$$

Optimization Problem: try 3

Idea: put arbitrary constraint on functional margin

$$\boxed{\hat{\gamma} = 1}$$

$$\min_{\vec{w},b} \quad \frac{1}{2}\|\vec{w}\|^2$$
$$\text{s.t.} \quad y_i(\vec{w}\cdot\vec{x}_i + b) \geq 1, \quad i = 1, \cdots, n$$

$$\boxed{\begin{aligned} &\min_{\vec{w},b} \quad \frac{1}{2}\|\vec{w}\|^2 \\ &\text{s.t.} \quad -y_i(\vec{w}\cdot\vec{x}_i + b) + 1 \leq 0, \quad i = 1, \cdots, n \end{aligned}}$$

Lagrangian

- The alpha values are our Lagrange multipliers
- We don't care about our constraint if it is not *active*

$$\mathcal{L}(\vec{w}, b, \vec{\alpha}) = \frac{1}{2}\|\vec{w}\|^2 - \sum_{i=1}^{n} \alpha_i \left[y_i(\vec{w} \cdot \vec{x}_i + b) - 1 \right]$$

Lagrangian

- The alpha values are our Lagrange multipliers
- We don't care about our constraint if it is not *active*

$$\mathcal{L}(\vec{w}, b, \vec{\alpha}) = \frac{1}{2}\|\vec{w}\|^2 - \sum_{i=1}^{n} \alpha_i \left[y_i(\vec{w} \cdot \vec{x}_i + b) - 1 \right]$$

- First minimize with respect to w & b, becomes W(alpha)

$$\mathcal{L}(\vec{w}, b, \vec{\alpha}) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} y_i y_j \alpha_i \alpha_j \vec{x}_i \cdot \vec{x}_j$$

Lagrangian

- The alpha values are our Lagrange multipliers
- We don't care about our constraint if it is not *active*

$$\mathcal{L}(\vec{w}, b, \vec{\alpha}) = \frac{1}{2}\|\vec{w}\|^2 - \sum_{i=1}^{n} \alpha_i \left[y_i(\vec{w} \cdot \vec{x}_i + b) - 1 \right]$$

- First minimize with respect to w & b, becomes W(alpha)

$$\mathcal{L}(\vec{w}, b, \vec{\alpha}) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} y_i y_j \alpha_i \alpha_j \boxed{\vec{x}_i \cdot \vec{x}_j}$$

Kernel Trick

- Now we can replace dot products with any kernel!

$$W(\vec{\alpha}) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} y_i y_j \alpha_i \alpha_j K(\vec{x}_i, \vec{x}_j)$$

Final Goal: classification

- After using Kernel Trick with dual optimization problem, we have:

$$\vec{w}^* = \sum_{i=1}^{n} \alpha_i y_i \vec{x}_i$$

- To classify, we use:

$$h(\vec{x}) = \text{sign}\left(\vec{w}^* \cdot \vec{x} + b^*\right)$$

Final Goal: classification

- After using Kernel Trick with dual optimization problem, we have:

$$\vec{w}^* = \sum_{i=1}^{n} \alpha_i y_i \vec{x}_i$$

- To classify, we use:

$$h(\vec{x}) = \text{sign}\left(\vec{w}^* \cdot \vec{x} + b^*\right)$$

$$h(\vec{x}) = \text{sign}\left(\sum_{i=1}^{n} \alpha_i y_i \vec{x}_i \cdot \vec{x} + b^*\right)$$

Final Goal: classification

- After using Kernel Trick with dual optimization problem, we have:

$$\vec{w}^* = \sum_{i=1}^{n} \alpha_i y_i \vec{x}_i$$

- To classify, we use:

$$h(\vec{x}) = \text{sign}\left(\vec{w}^* \cdot \vec{x} + b^*\right)$$

$$h(\vec{x}) = \text{sign}\left(\sum_{i=1}^{n} \alpha_i y_i \vec{x}_i \cdot \vec{x} + b^*\right)$$

$$\boxed{h(\vec{x}) = \text{sign}\left(\sum_{i=1}^{n} \alpha_i y_i K(\vec{x}_i, \vec{x}) + b^*\right)}$$

If the surface normal does not match our normal, we could move in some direction to improve our position

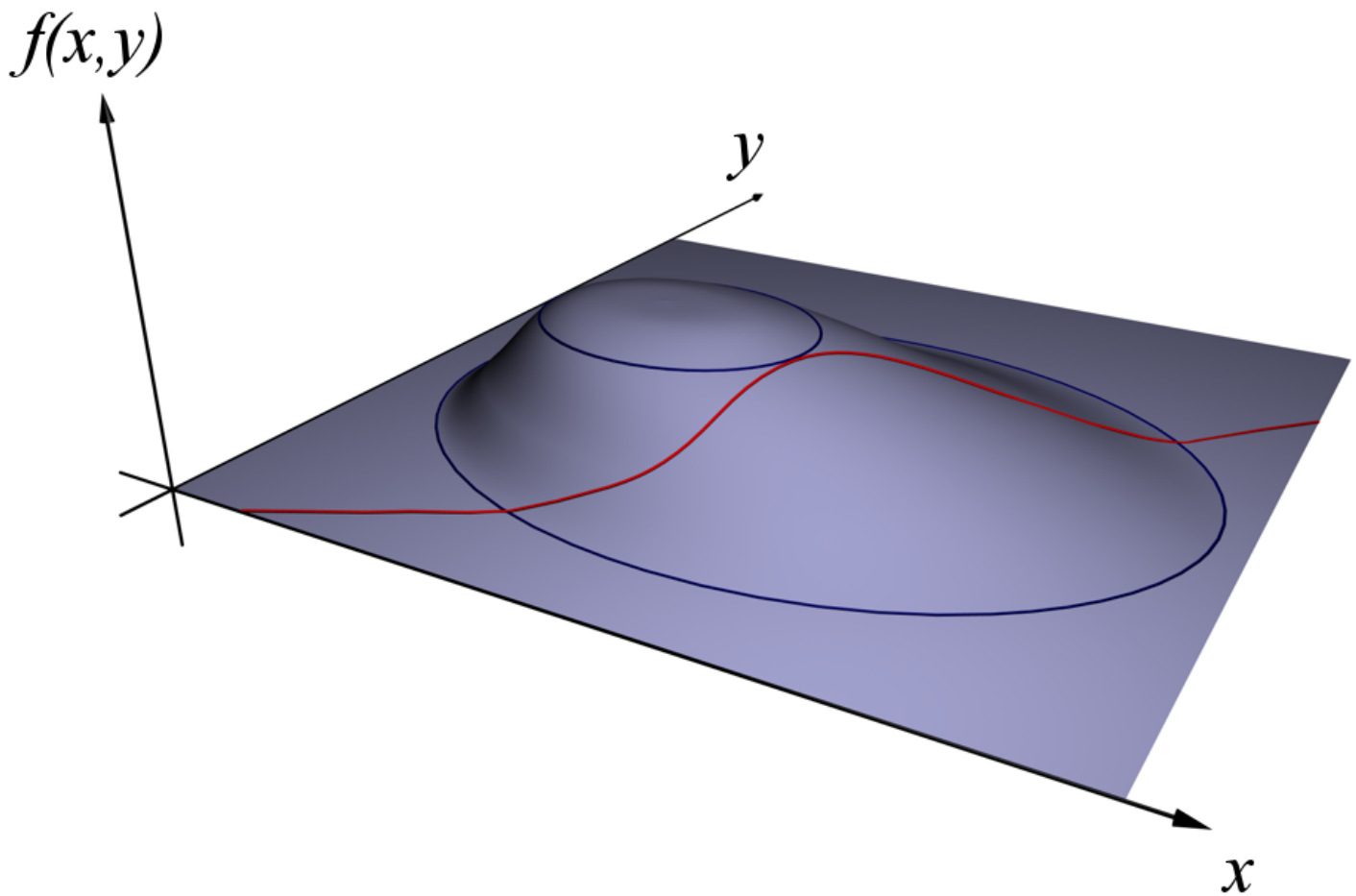
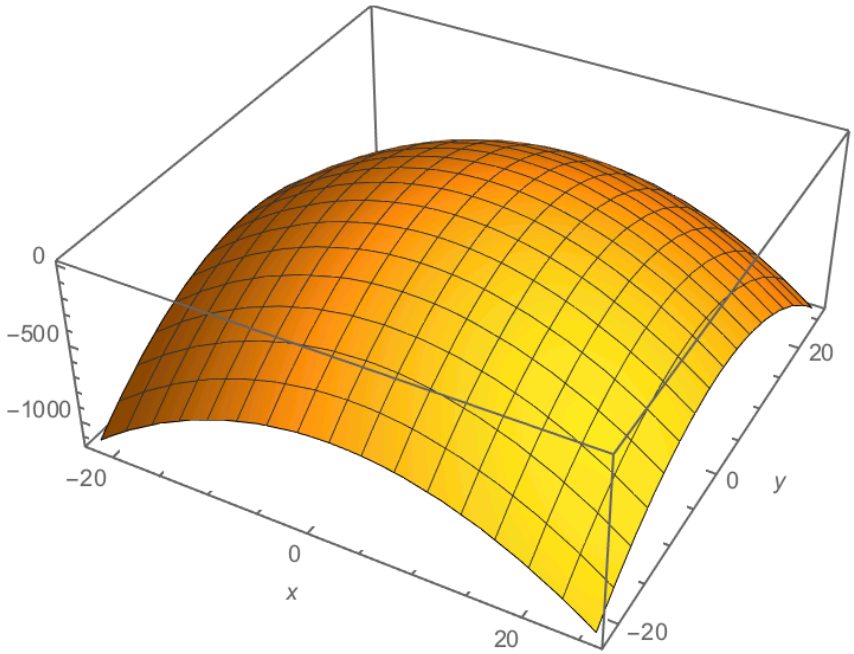
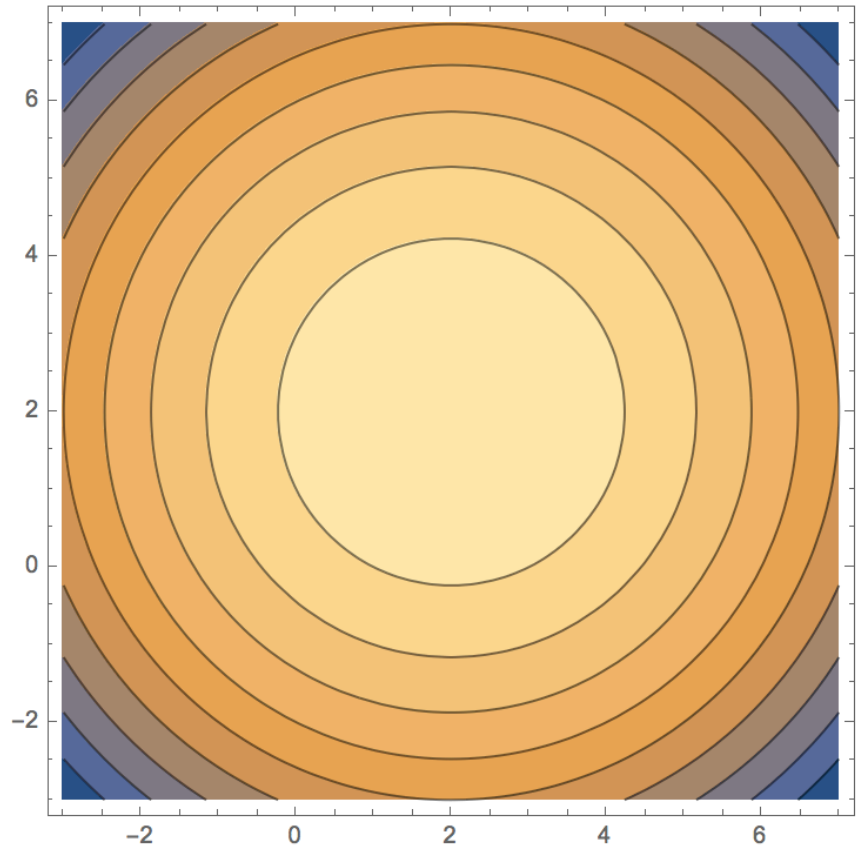


Image: Khan Academy

Lagrange multipliers example 1

$$f(x,y) = 5 - (x-2)^2 - (y-2)^2$$





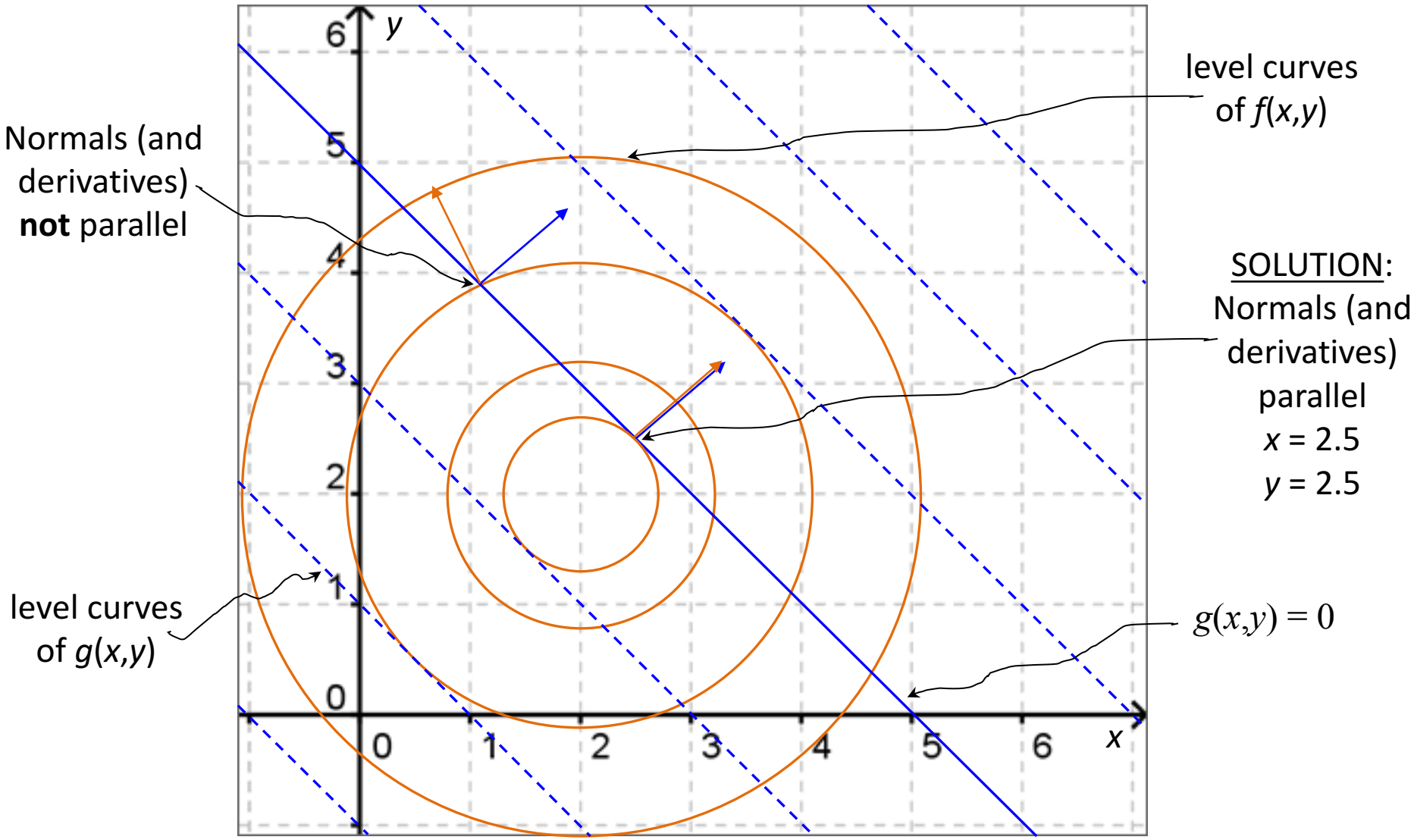
Contour plot of *f*(*x*,*y*)

$$\text{maximize}_{x,y} \quad f(x,y)$$

$$s.t. \quad g(x,y) = 0$$

$$g(x,y) = -5 + x + y$$

Lagrange multipliers example 1



Coordinate ascent/descent vs. Gradient ascent/descent

- In gradient descent we update all our model parameters at once

Coordinate ascent/descent vs. Gradient ascent/descent

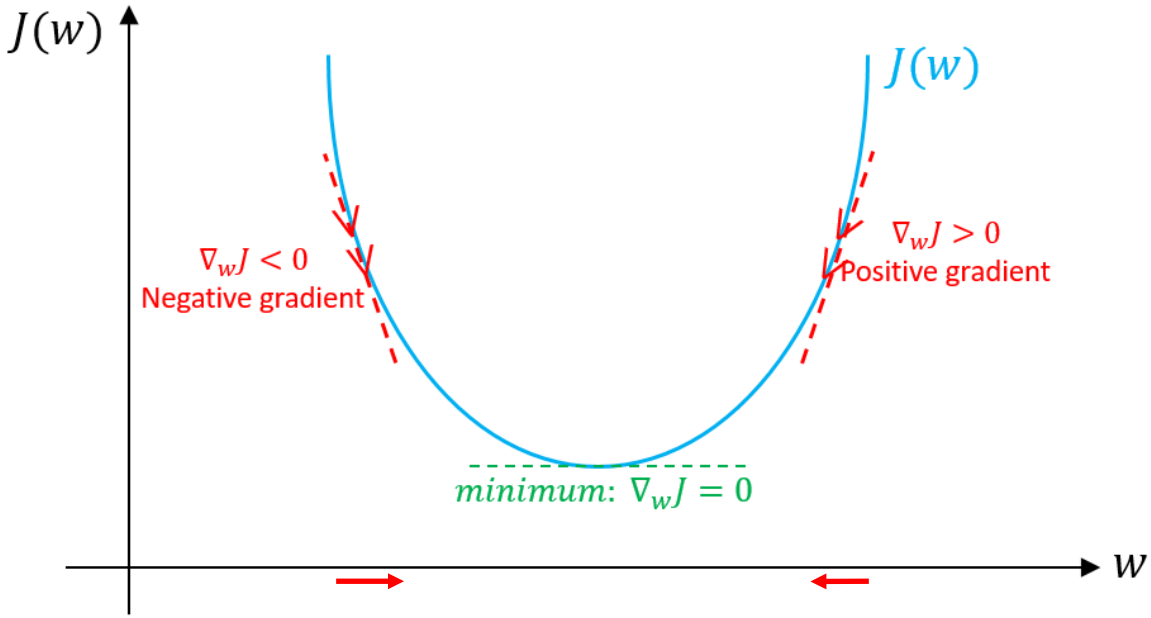
- In gradient descent we update all our model parameters at once
  - Use one data point (“stochastic”)
  - Use a few data points (“mini-batch”)
  - Use all data points (“batch”)

Coordinate ascent/descent vs. Gradient ascent/descent

- In gradient descent we update all our model parameters at once
  - Use one data point ("stochastic")
  - Use a few data points ("mini-batch")
  - Use all data points ("batch")

- In coordinate descent we fix all model parameters but 1, then optimize with respect to this parameter, using all our data

Optimization goal: minimize or maximize a function



- Minimize:
  - new_value = old_value – learning_rate * gradient
- Maximize:
  - new_value = old_value + learning_rate * gradient

Image: "Towards Data Science"

Outline for April 24

- Announcements
- Midterm practice problems (Handout 19)
- Review SVMs
- Open time for questions