# Indian Institute of Information Technology, Allahabad


# Introduction to Machine Learning

## Logistic Regression and Linear Classification

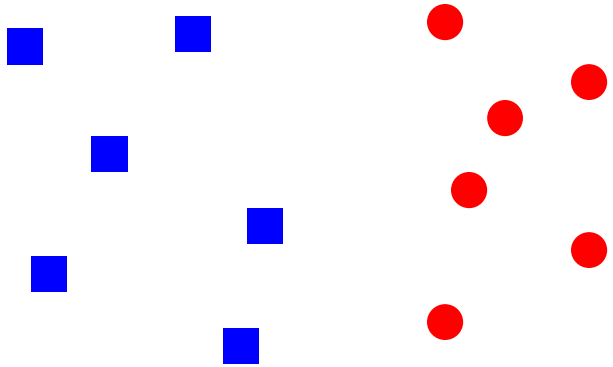
By

**Dr. Shiv Ram Dubey**
Associate Professor
Computer Vision and Biometrics Lab
Department of Information Technology
Indian Institute of Information Technology, Allahabad
Web: https://profile.iiita.ac.in/srdubey/

# DISCLAIMER

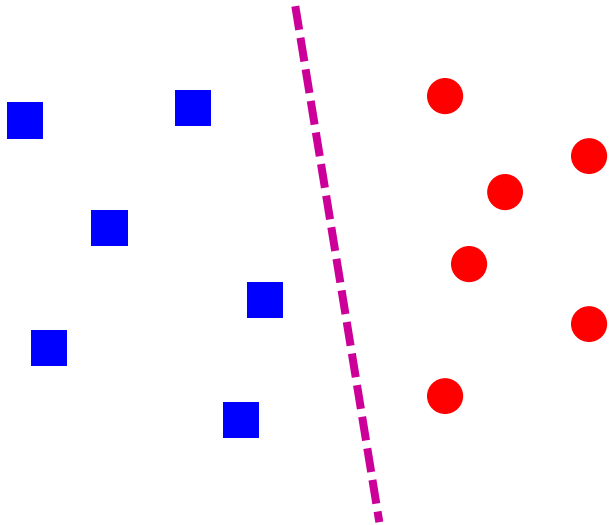
**The content (text, image, and graphics) used in this slide are adopted from many sources for academic purposes. Broadly, the sources have been given due credit appropriately. However, there is a chance of missing out some original primary sources. The authors of this material do not claim any copyright of such material.**

# Linear Classifiers – 2 Class Problem



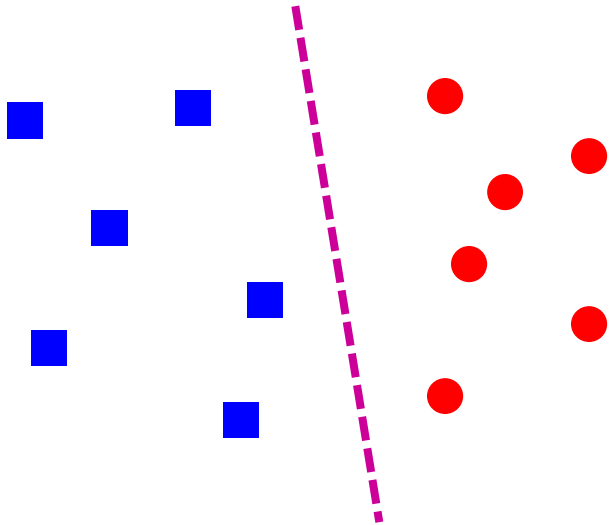
- Find a *linear function* to separate the classes:

# Linear Classifiers – 2 Class Problem



- Find a *linear function* to separate the classes:

# Linear Classifiers – 2 Class Problem



- Find a *linear function* to separate the classes:

$$f(\mathbf{x}) = \text{sign}(\mathbf{w} \cdot \mathbf{x} + b)$$

# Loss Minimization View of ML

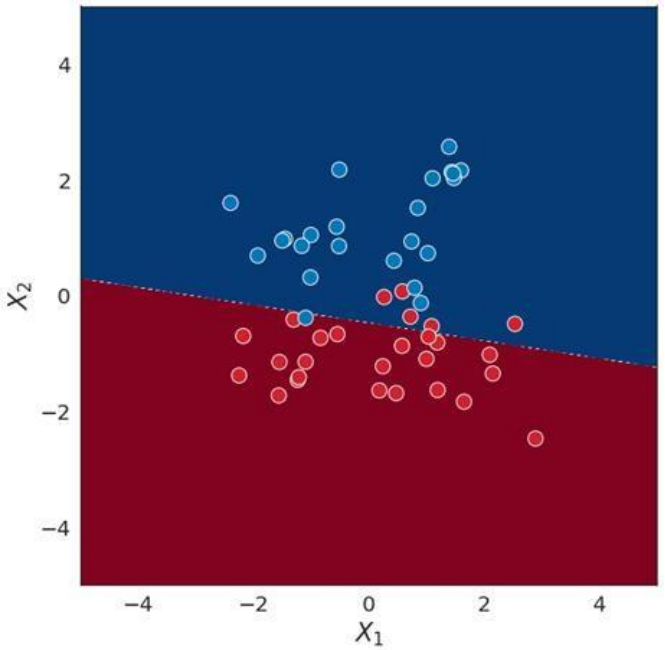
- **Three design decisions**
  - **Model family:** What are the candidate models $f$? (E.g., linear functions)
  - **Loss function:** How to define "approximating"? (E.g., MSE loss)
  - **Optimizer:** How do we optimize the loss? (E.g., gradient descent)

- How do we adapt to classification?

# Linear Functions for (Binary) Classification

- **Input:** Dataset $Z = \{ (x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n) \}$

- **Classification:**
  - Labels $y_i \in \{0, 1\}$
  - Predict $y_i \approx 1(\mathrm{w}^\top x_i \geq 0)$
  - $1(C)$ equals 1 if $C$ is true and 0 if $C$ is false
  - How to learn w? **Need a loss function!**

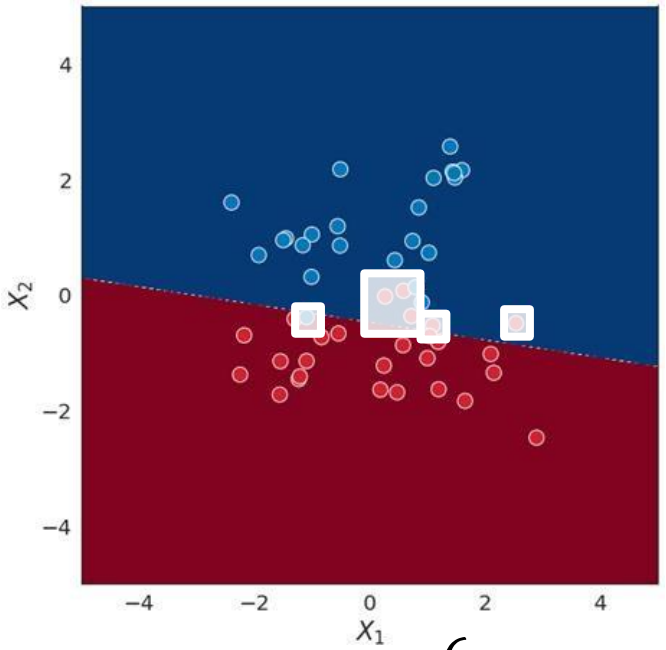


# Loss Functions for Linear Classifiers

- **(In)accuracy:**

$$L(w; Z) = \frac{1}{n}\sum_{i=1}^{n} 1(y_i \neq f_w(x_i))$$

- Computationally intractable
- Often, but not always the "true" loss (e.g., imbalanced data)

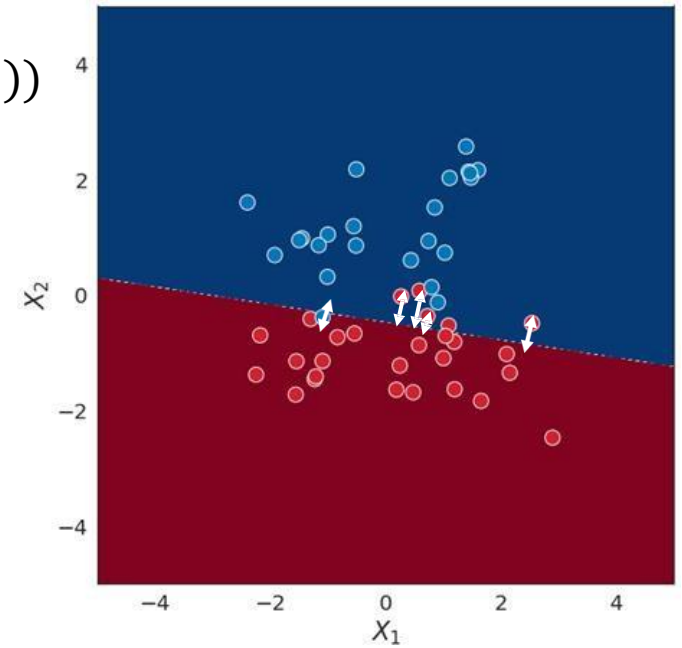



$$L(w; Z) = \frac{6}{50}$$

# Loss Functions for Linear Classifiers

- **Distance:**

$$L(w; Z) = \frac{1}{n}\sum_{i=1}^{n} dist(x_i, f_w).1(y_i \neq f_w(x_i))$$




$$L(\mathrm{w}; Z) = 1.2$$

- If $L(\mathrm{w}; Z) = 0$, then 100% accuracy
- Variant of this loss results in SVM
- We consider a more general strategy

# What is Logistic Regression?

- Logistic regression is a statistical **classification** model that predicts the **probability** of a **binary** outcome (e.g., yes/no) by applying a **logistic (sigmoid) function** to a **linear** combination of input features.

- …a frequent interview question

# Maximum Likelihood View of ML

- **Two design decisions**
  - **Likelihood:** Probability $p_{\mathrm{w}}(\, y \mid x \,)$ of data $(x, y)$ given parameters $\mathrm{w}$
  - **Optimizer:** How do we optimize the Negative Log-Likelihood (NLL)? (E.g., gradient descent)

- **Corresponding Loss Minimization View:**
  - **Model family:** Most likely label $f_{\mathrm{w}}(x) = \arg\max_y p_{\mathrm{w}}(\, y \mid x \,)$
  - **Loss function:** Negative log likelihood (NLL) $\ell(\mathrm{w}; Z) = -\sum_{i=1}^{n} \log p_{\mathrm{w}}(\, y_i \mid x_i \,)$

- Very powerful framework for designing cutting edge ML algorithms
  - Write down the "right" likelihood, form tractable approximation if needed

# Recall: Linear Regression under MLE View

- What about the model family?

$$f_{\mathrm{w}}(x) = \arg\max_{y} p_{\mathrm{w}}(y \mid x)$$

$$= \arg\max_{y} \frac{1}{\sqrt{2\pi}} \cdot e^{-\frac{(\mathrm{w}^{\mathsf{T}} x - y)_2^2}{2}}$$

$$= \mathrm{w}^{\mathsf{T}} x$$

- **Recovers linear functions!**

# What About Classification?

**Compare to linear regression:**

$$p_{\mathrm{w}}(y \mid x_i) \propto e^{-\frac{(\mathrm{w}^{\mathsf{T}} x_i - y)^2}{2}}$$

- Consider the following choice:

$$p_{\mathrm{w}}(Y = 0 \mid x_i) \propto e^{-\frac{\mathrm{w}^{\mathsf{T}} x_i}{2}} \text{ and } p_{\mathrm{w}}(Y = 1 \mid x_i) \propto e^{\frac{\mathrm{w}^{\mathsf{T}} x_i}{2}}$$

- Then, we have

$$p_{\mathrm{w}}(Y = 1 \mid x_i) = \quad ??????$$

# What About Classification?

**Compare to linear regression:**

$$p_{\mathrm{w}}(y \mid x_i) \propto e^{-\frac{(\mathrm{w}^{\mathsf{T}} x_i - y)^2}{2}}$$

- Consider the following choice:

$$p_{\mathrm{w}}(Y = 0 \mid x_i) \propto e^{-\frac{\mathrm{w}^{\mathsf{T}} x_i}{2}} \text{ and } p_{\mathrm{w}}(Y = 1 \mid x_i) \propto e^{\frac{\mathrm{w}^{\mathsf{T}} x_i}{2}}$$

- Then, we have

**Sigmoid function**

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

$$p_{\mathrm{w}}(Y = 1 \mid x_i) = \frac{e^{\frac{\mathrm{w}^{\mathsf{T}} x_i}{2}}}{e^{\frac{\mathrm{w}^{\mathsf{T}} x_i}{2}} + e^{-\frac{\mathrm{w}^{\mathsf{T}} x_i}{2}}} = \frac{1}{1 + e^{-\mathrm{w}^{\mathsf{T}} x_i}}$$

# What About Classification?

**Compare to linear regression:**

$$p_{\mathrm{w}}(y \mid x_i) \propto e^{-\frac{(\mathrm{w}^{\mathsf{T}} x_i - y)^2}{2}}$$

- Consider the following choice:

$$p_{\mathrm{w}}(Y = 0 \mid x_i) \propto e^{-\frac{\mathrm{w}^{\mathsf{T}} x_i}{2}} \text{ and } p_{\mathrm{w}}(Y = 1 \mid x_i) \propto e^{\frac{\mathrm{w}^{\mathsf{T}} x_i}{2}}$$

- Then, we have

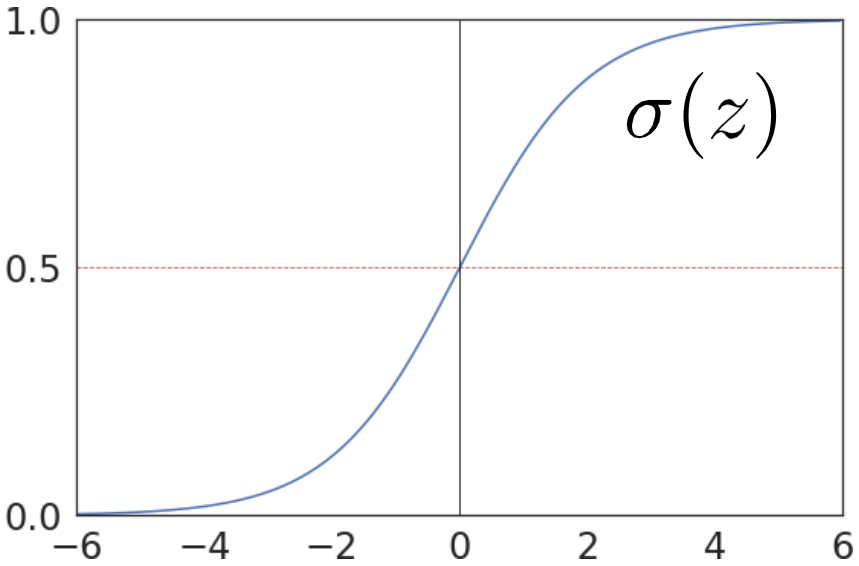
**Sigmoid function**

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

$$p_{\mathrm{w}}(Y = 1 \mid x_i) = \frac{e^{\frac{\mathrm{w}^{\mathsf{T}} x_i}{2}}}{e^{\frac{\mathrm{w}^{\mathsf{T}} x_i}{2}} + e^{-\frac{\mathrm{w}^{\mathsf{T}} x_i}{2}}} = \frac{1}{1 + e^{-\mathrm{w}^{\mathsf{T}} x_i}}$$

- Furthermore, $p_{\mathrm{w}}(Y = 0 \mid x_i) = 1 - \sigma(\mathrm{w}^{\mathsf{T}} x_i)$

# Logistic/Sigmoid Function



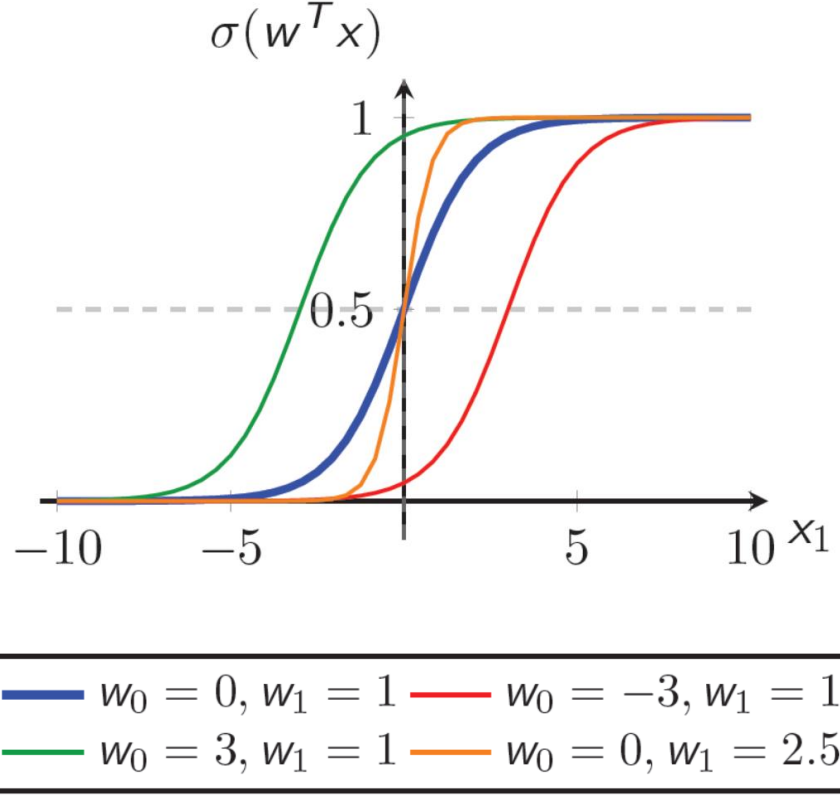

$$p_{\mathrm{w}}(Y = 1 \mid x_i) = \sigma(\mathrm{w}^{\mathsf{T}} x_i)$$

# Logistic/Sigmoid Function

Simple example:

$w = [w_0, w_1]$ and $x = [1, x_1]$

- $w_0$ controls the offset
- $w_1$ controls the steepness



# Logistic Regression Model Family

$$f_{\mathrm{w}}(x) = \arg\max_{y} p_{\mathrm{w}}(y \mid x)$$

$$= \arg\max_{y} \begin{cases} \sigma(\mathrm{w}^{\mathsf{T}} x) & \text{if } y = 1 \\ 1 - \sigma(\mathrm{w}^{\mathsf{T}} x) & \text{if } y = 0 \end{cases}$$

$$= \begin{cases} 1 & \text{if } \sigma(\mathrm{w}^{\mathsf{T}} x) \geq \frac{1}{2} \\ 0 & \text{otherwise} \end{cases}$$

# Logistic Regression Model Family

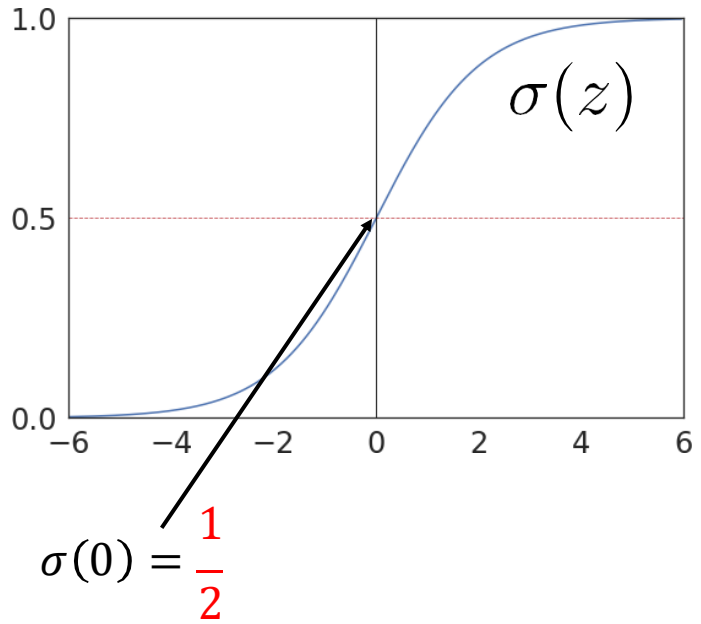
$$f_{\mathrm{w}}(x) = \arg\max_{y} p_{\mathrm{w}}(y \mid x)$$

$$= \arg\max_{y} \begin{cases} \sigma(\mathrm{w}^{\mathsf{T}} x) & \text{if } y = 1 \\ 1 - \sigma(\mathrm{w}^{\mathsf{T}} x) & \text{if } y = 0 \end{cases}$$

$$= \begin{cases} 1 & \text{if } \sigma(\mathrm{w}^{\mathsf{T}} x) \geq \frac{1}{2} \\ 0 & \text{otherwise} \end{cases}$$

$$= \begin{cases} 1 & \text{if } \mathrm{w}^{\mathsf{T}} x \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

$$= 1(\mathrm{w}^{\mathsf{T}} x \geq 0)$$




- **Recovers linear classifiers!**

# Logistic Regression Algorithm

- Then, we have the following Negative Log-Likelihood (NLL) loss:

$$\begin{aligned}\ell(\mathrm{w}; Z) &= -\sum_{i=1}^{n} \log p_{\mathrm{w}}(y_i \mid x_i) \\ &= -\sum_{i=1}^{n} 1(y_i = 1) \cdot \log\big(\sigma(\mathrm{w}^{\mathsf{T}} x_i)\big) + 1(y_i = 0) \cdot \log\big(1 - \sigma(\mathrm{w}^{\mathsf{T}} x_i)\big) \\ &= -\sum_{i=1}^{n} y_i \cdot \log\big(\sigma(\mathrm{w}^{\mathsf{T}} x_i)\big) + (1 - y_i) \cdot \log\big(1 - \sigma(\mathrm{w}^{\mathsf{T}} x_i)\big)\end{aligned}$$

Binary Cross-Entropy (BCE) Loss

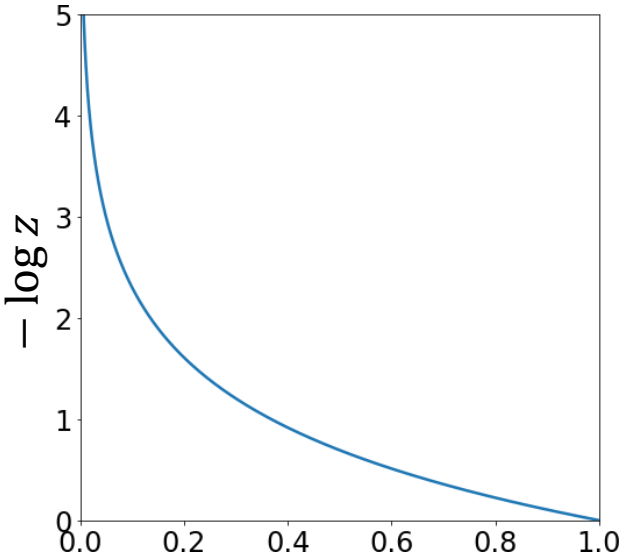
- Logistic regression minimizes this loss:

$$\widehat{w}(Z) = \arg\min_{\mathrm{w}} \ell(\mathrm{w}; Z)$$

# Intuition on the Objective

- Loss for example $i$ is

$$\begin{cases} -\log\left(\sigma(\mathrm{w}^{\mathsf{T}} x_i)\right) & \text{if } y_i = 1 \\ -\log\left(1 - \sigma(\mathrm{w}^{\mathsf{T}} x_i)\right) & \text{if } y_i = 0 \end{cases}$$

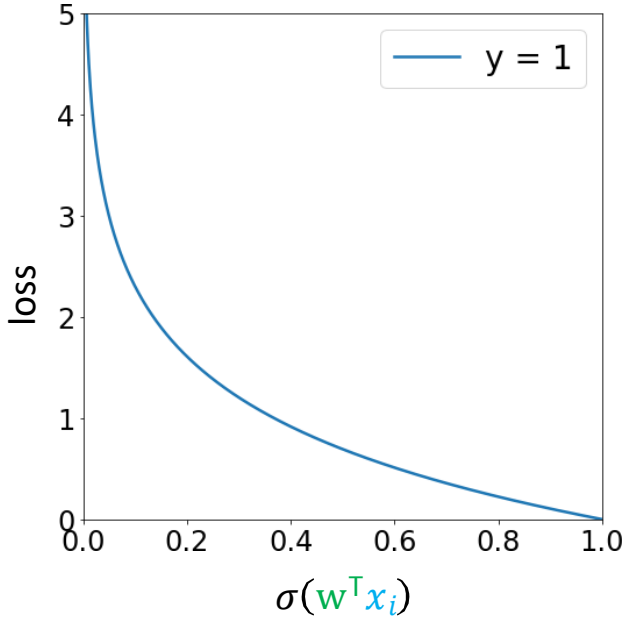


# Intuition on the Objective

- Loss for example $i$ is

$$\begin{cases} -\log\left(\sigma(\mathrm{w}^\top x_i)\right) & \text{if } y_i = 1 \\ -\log\left(1-\sigma(\mathrm{w}^\top x_i)\right) & \text{if } y_i = 0 \end{cases}$$

- If $y_i = 1$:
  - If $\sigma(\mathrm{w}^\top x_i) = 1$, then loss $= 0$
  - As $\sigma(\mathrm{w}^\top x_i) \to 0$, loss $\to \infty$

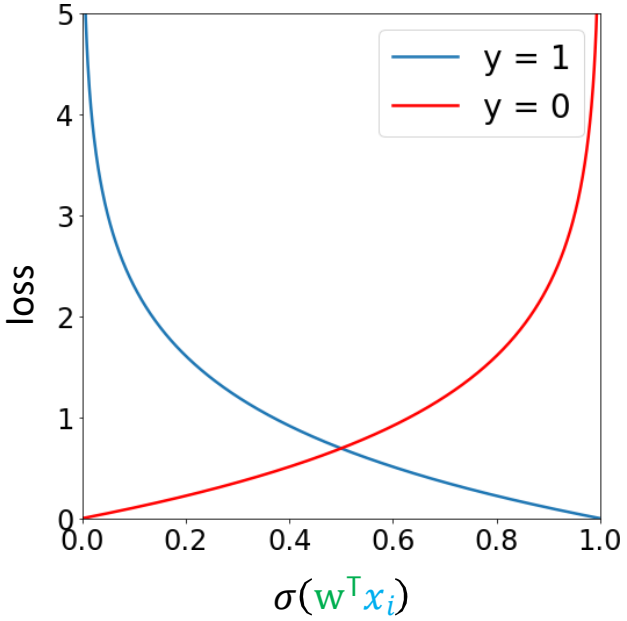



$$-y_i \cdot \log\left(\sigma(\mathrm{w}^\top x_i)\right) - (1-y_i) \cdot \log\left(1-\sigma(\mathrm{w}^\top x_i)\right)$$

# Intuition on the Objective

- Loss for example $i$ is

$$\begin{cases} -\log\left(\sigma(\mathrm{w}^\top x_i)\right) & \text{if } y_i = 1 \\ -\log\left(1-\sigma(\mathrm{w}^\top x_i)\right) & \text{if } y_i = 0 \end{cases}$$




- If $y_i = 1$:
  - If $\sigma(\mathrm{w}^\top x_i) = 1$, then loss $= 0$
  - As $\sigma(\mathrm{w}^\top x_i) \to 0$, loss $\to \infty$
- If $y_i = 0$
  - If $\sigma(\mathrm{w}^\top x_i) = 0$, then loss $= 0$
  - As $\sigma(\mathrm{w}^\top x_i) \to 1$, loss $\to \infty$

$$-y_i \cdot \log\left(\sigma(\mathrm{w}^\top x_i)\right) - (1-y_i) \cdot \log\left(1-\sigma(\mathrm{w}^\top x_i)\right)$$

# Optimization for Logistic Regression

- To optimize the loss, we need its gradient:

$$\nabla_{\mathrm{w}}\ell(\mathrm{w};Z) = -\sum_{i=1}^{n} y_i \cdot \nabla_{\mathrm{w}} \log\big(\sigma(\mathrm{w}^{\mathrm{T}}x_i)\big) + (1-y_i)\cdot \nabla_{\mathrm{w}} \log\big(1-\sigma(\mathrm{w}^{\mathrm{T}}x_i)\big)$$

$$= -\sum_{i=1}^{n} y_i \cdot \frac{\nabla_{\mathrm{w}}\sigma(\mathrm{w}^{\mathrm{T}}x_i)}{\sigma(\mathrm{w}^{\mathrm{T}}x_i)} - (1-y_i)\cdot \frac{\nabla_{\mathrm{w}}\sigma(\mathrm{w}^{\mathrm{T}}x_i)}{1-\sigma(\mathrm{w}^{\mathrm{T}}x_i)}$$

$\sigma'(z) = \sigma(z)\big(1-\sigma(z)\big)$

$$= -\sum_{i=1}^{n} y_i \cdot \frac{\sigma(\mathrm{w}^{\mathrm{T}}x_i)\big(1-\sigma(\mathrm{w}^{\mathrm{T}}x_i)\big)\cdot x_i}{\sigma(\mathrm{w}^{\mathrm{T}}x_i)} - (1-y_i)\cdot \frac{\sigma(\mathrm{w}^{\mathrm{T}}x_i)\big(1-\sigma(\mathrm{w}^{\mathrm{T}}x_i)\big)\cdot x_i}{1-\sigma(\mathrm{w}^{\mathrm{T}}x_i)}$$

$$= -\sum_{i=1}^{n} y_i \cdot \big(1-\sigma(\mathrm{w}^{\mathrm{T}}x_i)\big)\cdot x_i - (1-y_i)\cdot \sigma(\mathrm{w}^{\mathrm{T}}x_i)\cdot x_i$$

$$= -\sum_{i=1}^{n} \big(y_i - \sigma(\mathrm{w}^{\mathrm{T}}x_i)\big)\cdot x_i$$

# Optimization for Logistic Regression

- Gradient of loss:

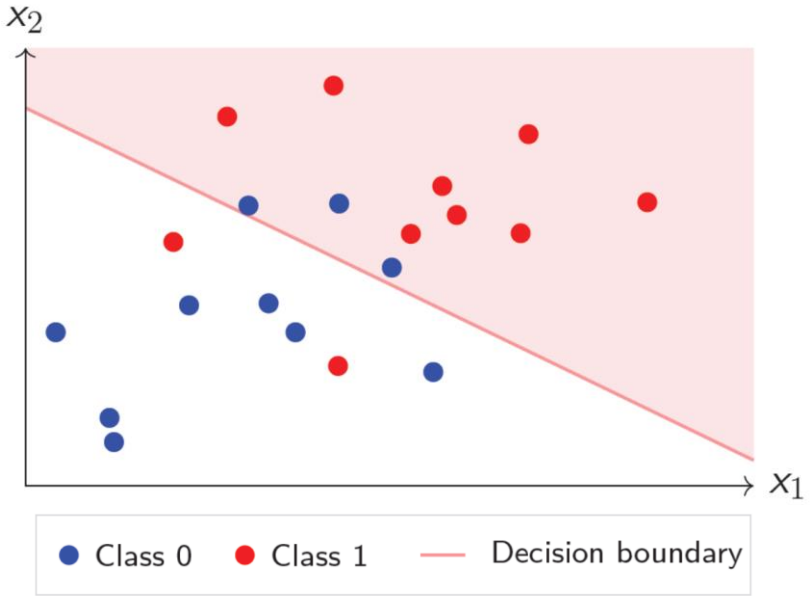
$$\nabla_{\mathrm{w}} \ell\,(\mathrm{w}; Z) = -\sum_{i=1}^{n} (\sigma\,(\mathrm{w}^{\mathsf{T}} x_i) - y_i\,) \cdot x_i$$

- Surprisingly similar to the gradient for linear regression!
  - Only difference is the $\sigma$

- Gradient descent works as before
  - No closed-form solution for $\widehat{w}(Z)$

# Logistic Regression is a Linear Classifier

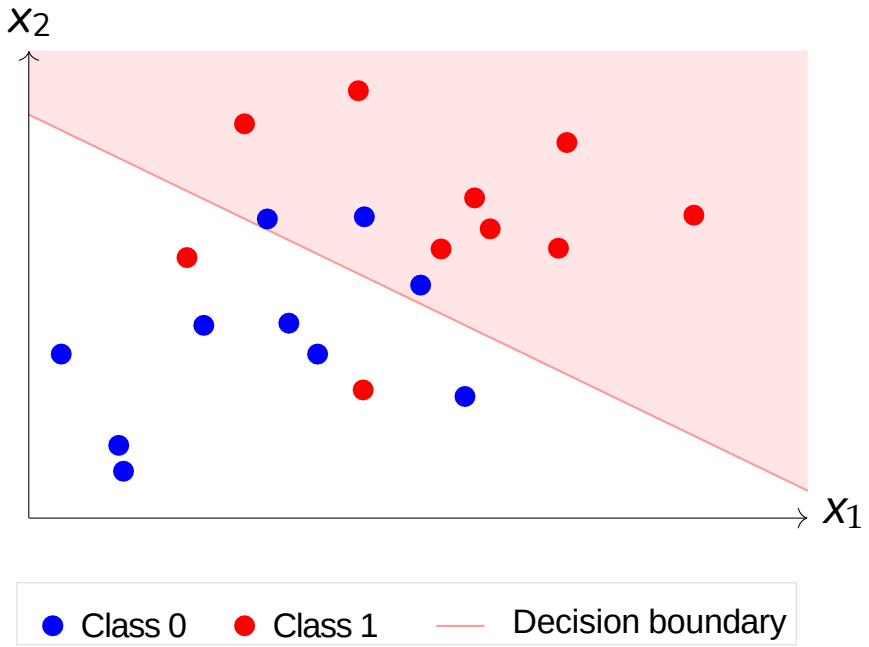
- A note on naming: it's called logistic *regression* because it's regressing the continuous value $P(Y|X) \in [0,1]$.
- The *classification* part only happens when we apply the threshold, i.e. predict Class 1 if $P_w(Y|X) \geq 0.5$.



# Linear Features: Create Simple Decision Boundary

- Simple regression models
  →smooth predictors
- Simple classifier models
  →smooth decision boundaries

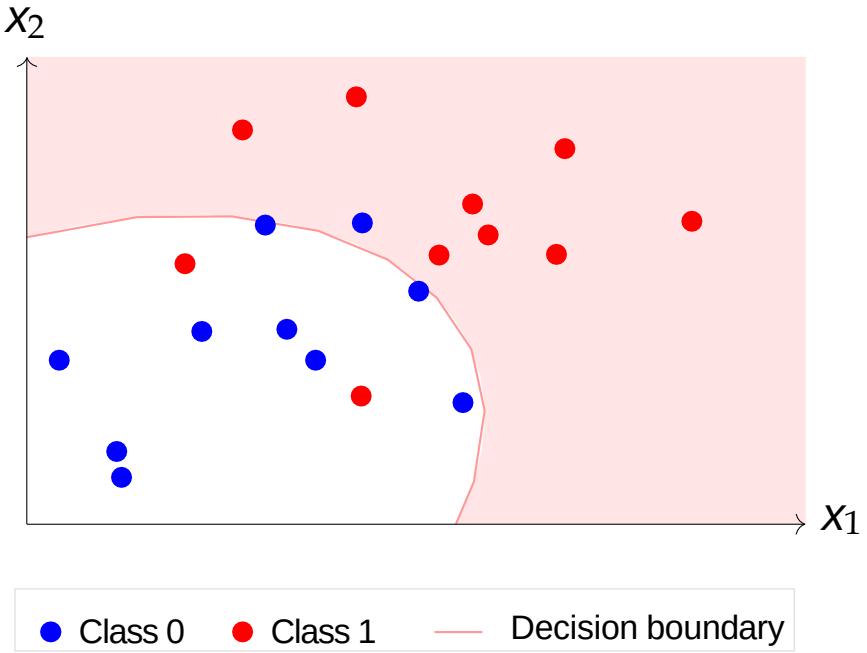
| Feature | Value | Coeff $w$ |
|---|---|---|
| $h_0(x)$ | 1 | 0.23 |
| $h_1(x)$ | $x_1$ | 1.12 |
| $h_2(x)$ | $x_2$ | -1.07 |



# Quadratic Features: Creates More Complex Decision Boundary

- Adding more features gives more complex models
- Decision boundary becomes more complex

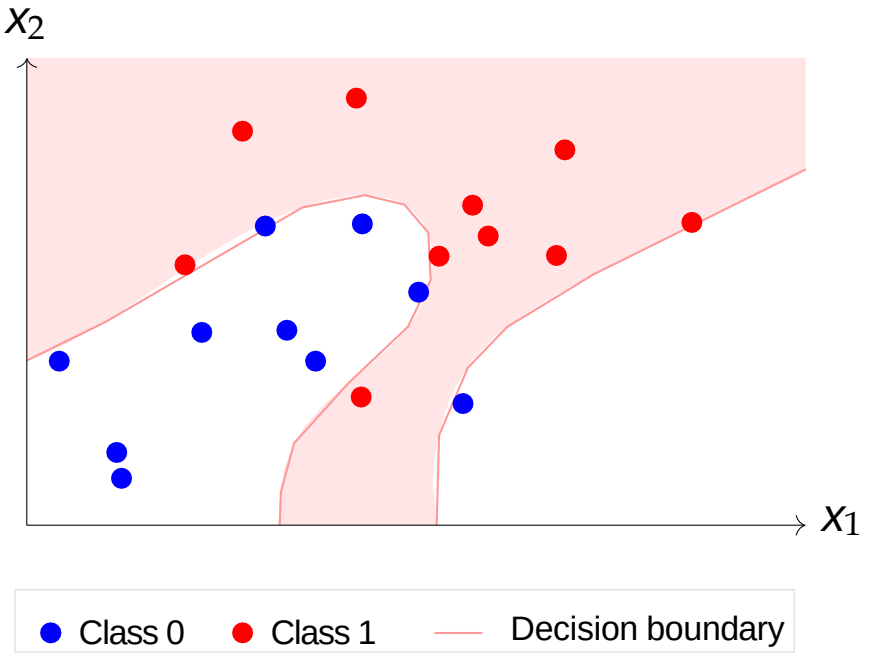
| Feature | Value | Coeff |
|---|---|---|
| $h_0(x)$ | 1 | 1.68 |
| $h_1(x)$ | $x_1$ | 1.39 |
| $h_2(x)$ | $x_2$ | -0.59 |
| $h_3(x)$ | $x_1^2$ | -0.17 |
| $h_4(x)$ | $x_2^2$ | -0.96 |
| $h_5(x)$ | $x_1x_2$ | 0.46 |



# Higher-degree Polynomial Features: Can Lead to Overfitting

- Too much model complexity can lead to overfitting

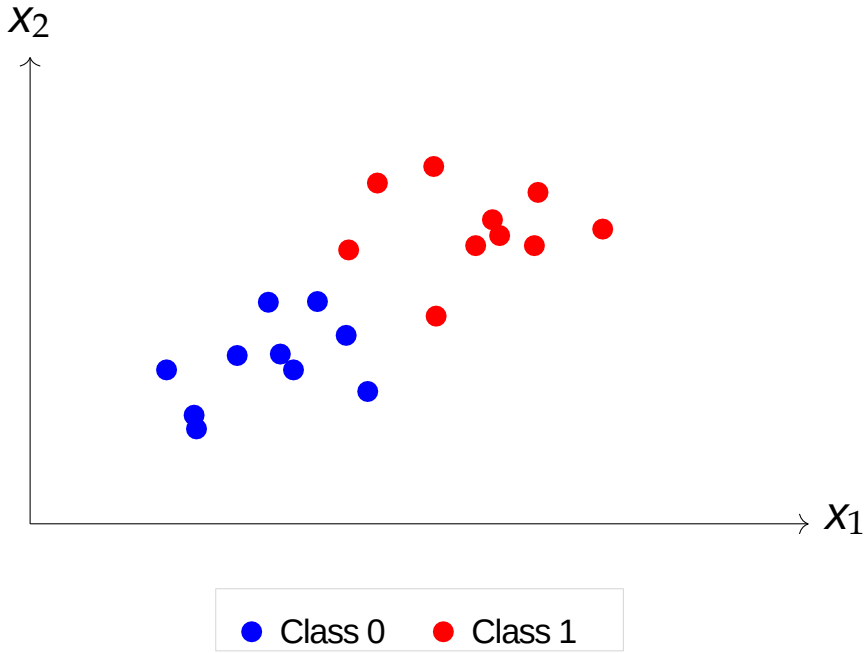
| Feature | Value | Coeff |
|---|---|---|
| $h_0(x)$ | 1 | 21.6 |
| $h_1(x)$ | $x_1$ | 5.3 |
| $h_2(x)$ | $x_2$ | -42.7 |
| $h_3(x)$ | $x_1^2$ | -15.9 |
| $h_4(x)$ | $x_2^2$ | -48.6 |
| ⋮ | ⋮ | ⋮ |
| $h_{k-1}(x)$ | $x_1^6$ | 0.8 |
| $h_k(x)$ | $x_2^6$ | -8.6 |



# Even Linear Features Can Overfit

When would this badly overfit?

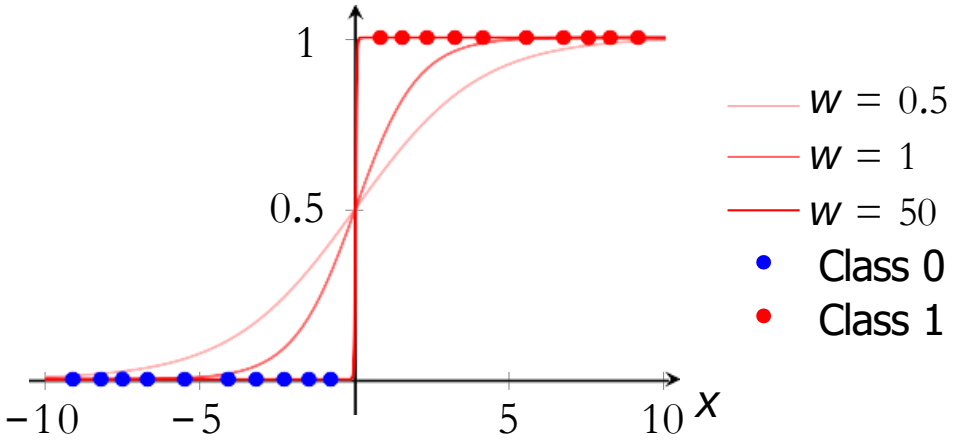
- When is the loss small?
- More likely with non-linear features. Why?



# Linearly Separable Data will Overfit

When data is linearly separable, $\|w\| \rightarrow \infty$

$$P(Y = 1) = \sigma(wx)$$



# Regularized Logistic Regression

We can add $L_1$ or $L_2$ regularization to the loss, e.g.:

$$\ell(\mathrm{w}; Z) = -\sum_{i=1}^{n} y_i \cdot \log\left(\sigma(\mathrm{w}^\top x_i)\right) + (1 - y_i) \cdot \log\left(1 - \sigma(\mathrm{w}^\top x_i)\right) + \lambda \cdot \|\mathrm{w}\|_2^2$$

- Regularization ensures that w does not become too large
  - Prevents overconfidence

- Regularization can also be **necessary**
  - Without regularization (i.e., $\lambda = 0$) and data is linearly separable, then gradient descent diverges (i.e., $\mathrm{w} \to \pm\infty$)

# Bias Term

- Most of the time, we want to add a bias term. Otherwise, the decision boundary is forced to pass through the origin (not true in general)

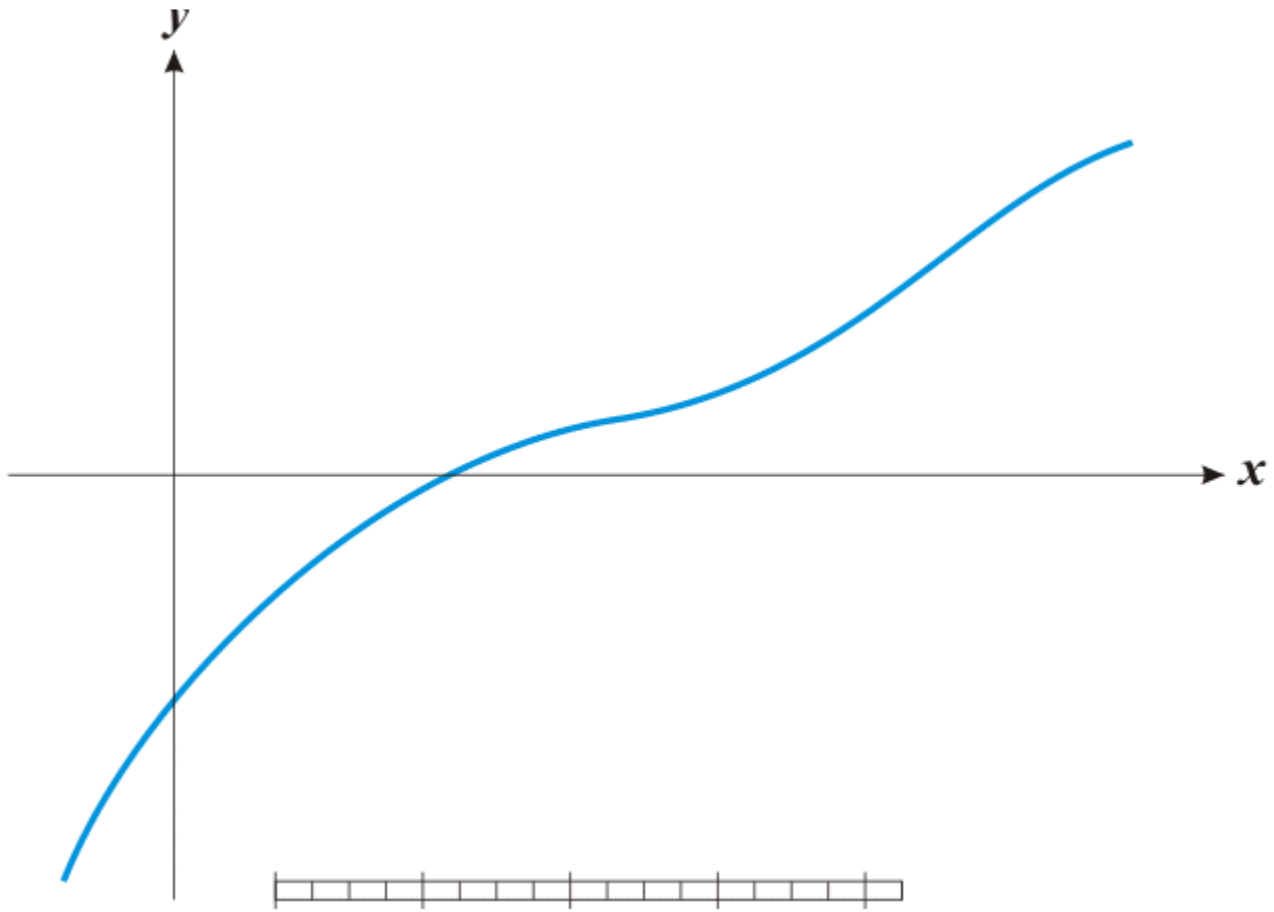
$$\ell(\mathrm{w}, \mathrm{b}; Z) = -\sum_{i=1}^{n} y_i \cdot \log(\sigma(\mathrm{w}^\top x_i + \mathrm{b})) + (1 - y_i) \cdot \log(1 - \sigma(\mathrm{w}^\top x_i + \mathrm{b})) + \lambda \cdot \|\mathrm{w}\|_2^2$$

# Newton's Method

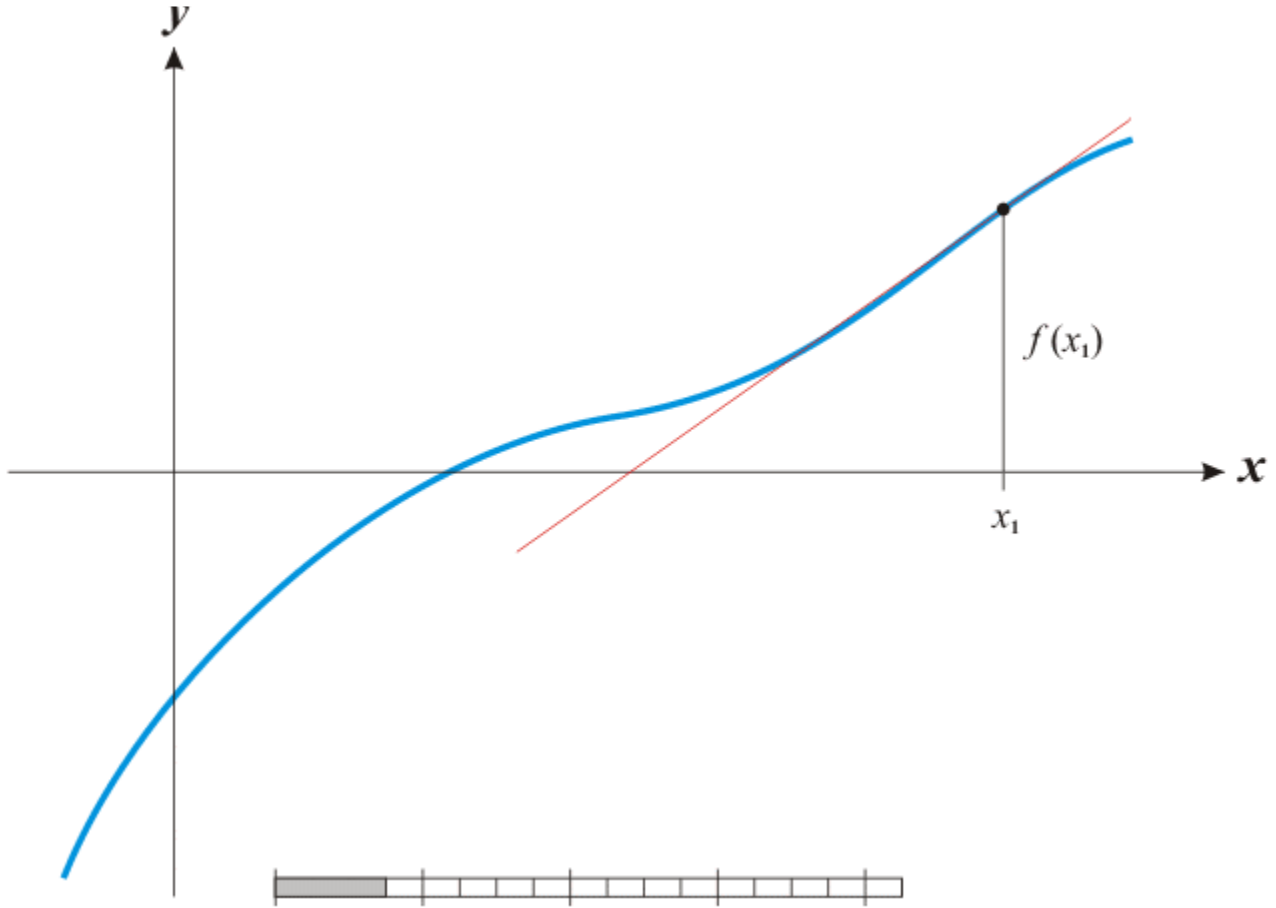
- **Newton's Method** is an iterative equation solver.
- It is an algorithm to find the roots of a polynomial function.
- In the simple, one-variable case, Newton's Method is implemented as follows:

1. *Find the tangent line to* $f(x)$ *at point* $(x_n, y_n)$
   - $y = f'(x_n)(x - x_n) + f(x_n)$
2. *Find the x-intercept of the tangent line,* $x_{n+1}$
   - $0 = f'(x_n)(x_{n+1} - x_n) + f(x_n)$
   - $-f(x_n) = f'(x_n)(x_{n+1} - x_n)$
   - $x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$
3. *Find the* $y$ *value at the x-intercept.*
   - $y_{n+1} = f(x_{n+1})$
4. *If* $y_{n+1} - y_n \approx 0$*:*
   - **return** $y_{n+1}$ because we've converged!
5. *Else update point* $(x_n, y_n)$*, and iterate*
   - $x = x_{n+1}, y = y_{n+1}$, **goto (1)**.

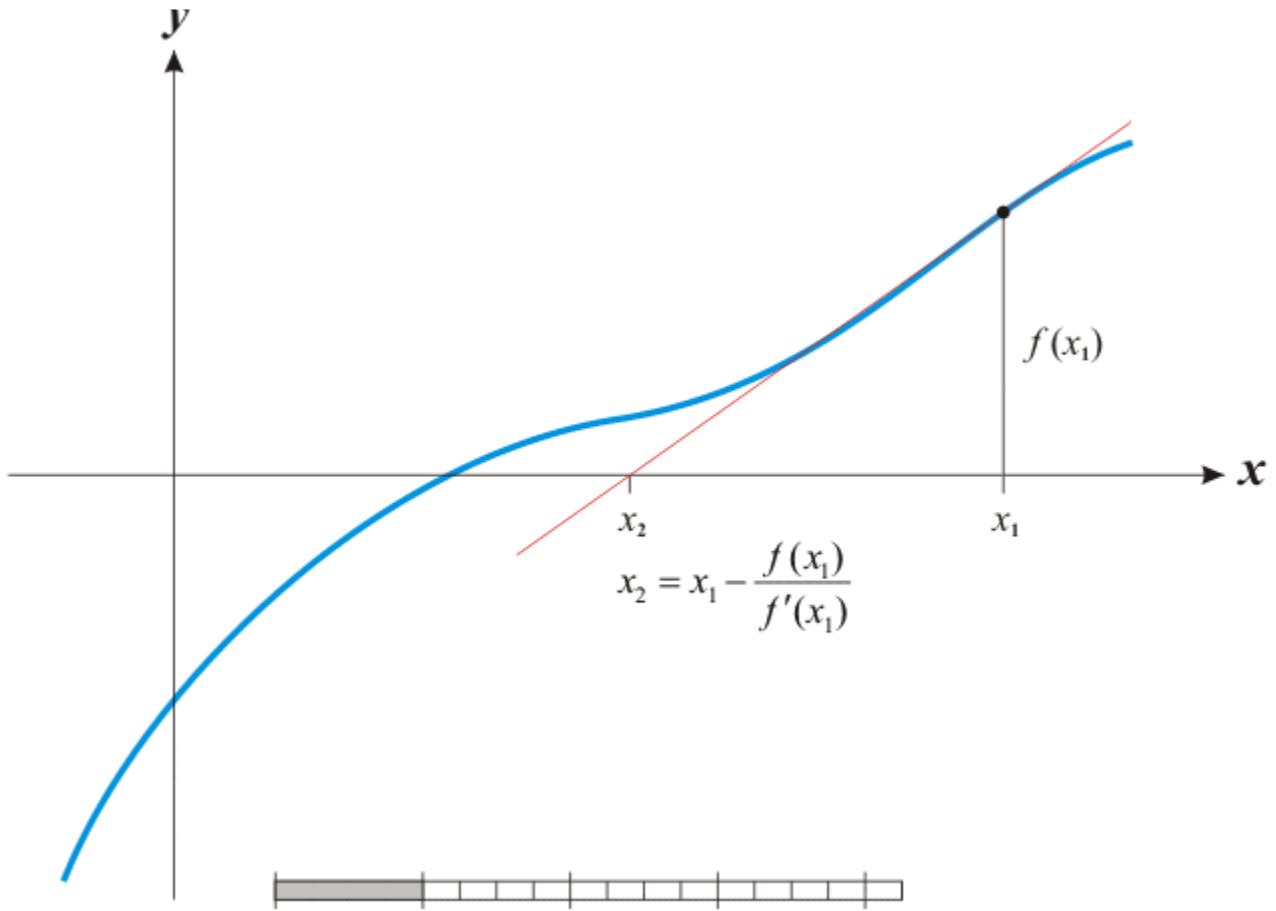
# Newton's Method



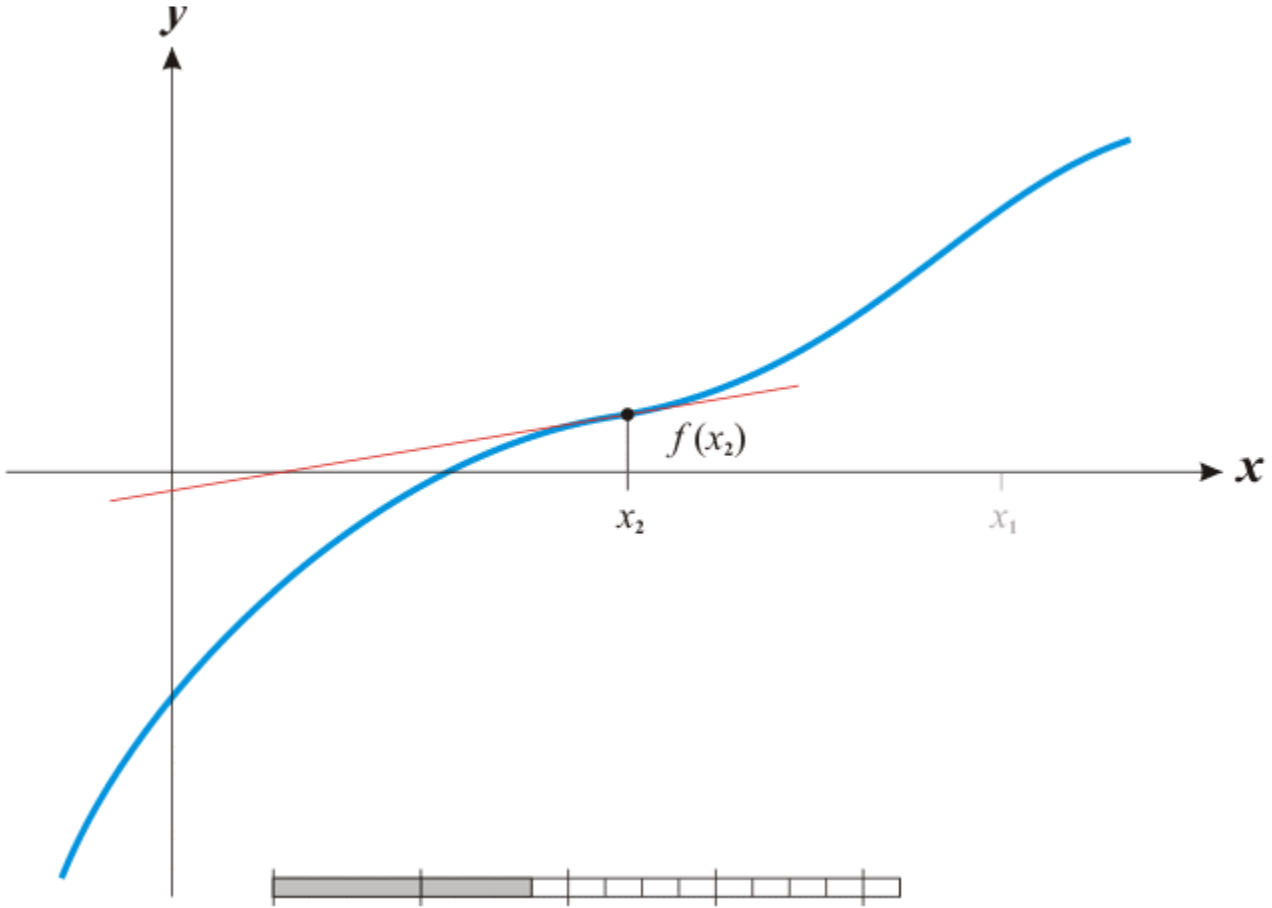
# Newton's Method



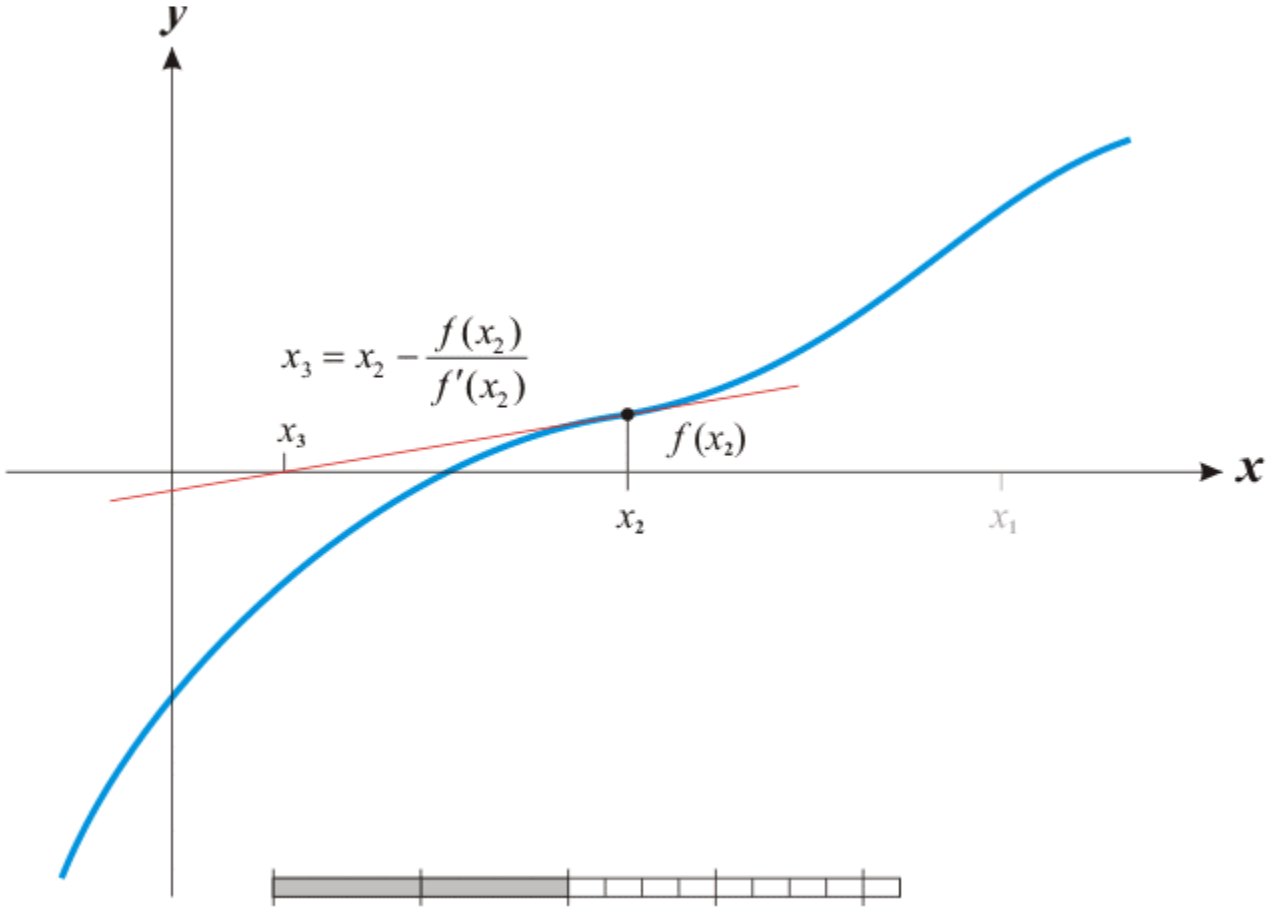
# Newton's Method



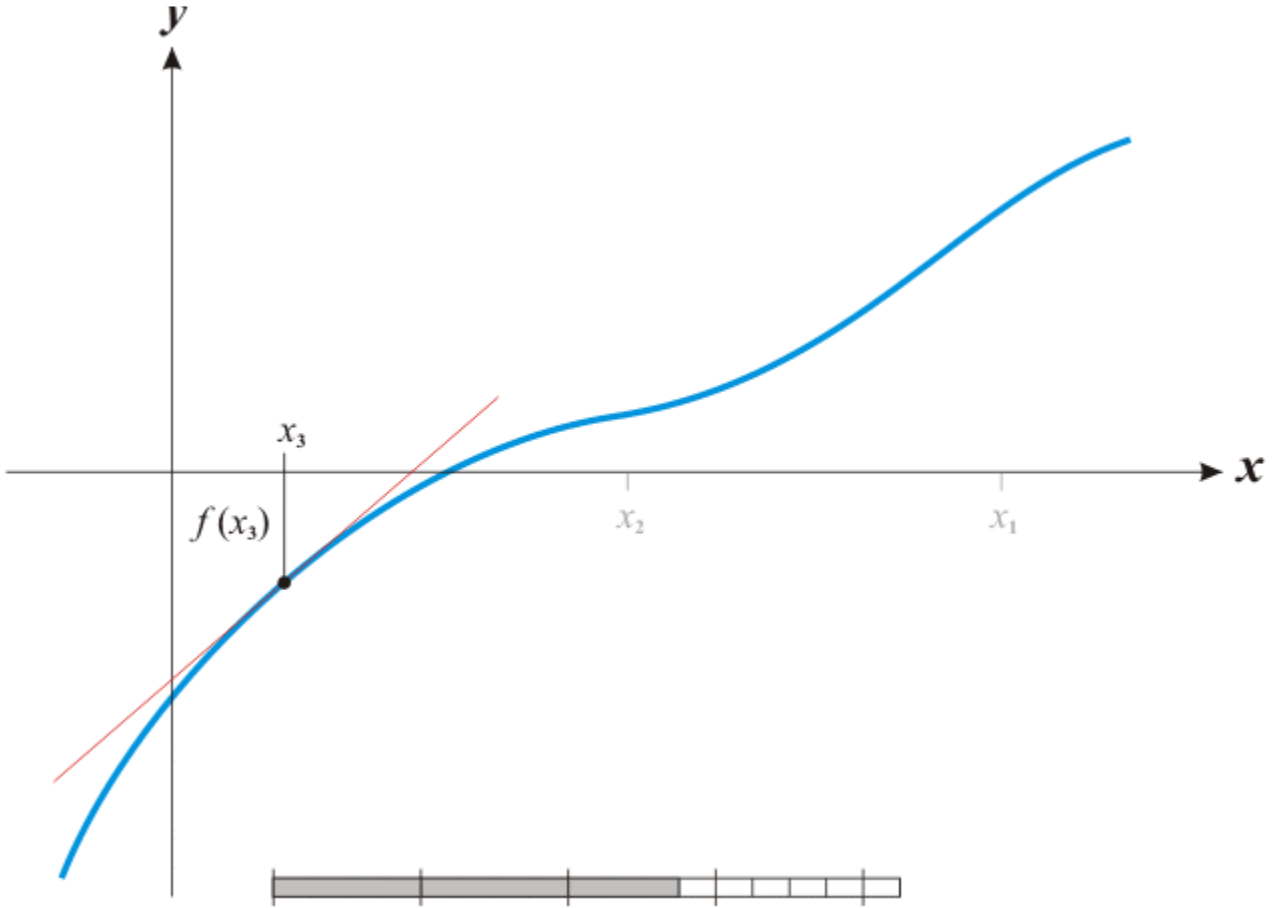
# Newton's Method



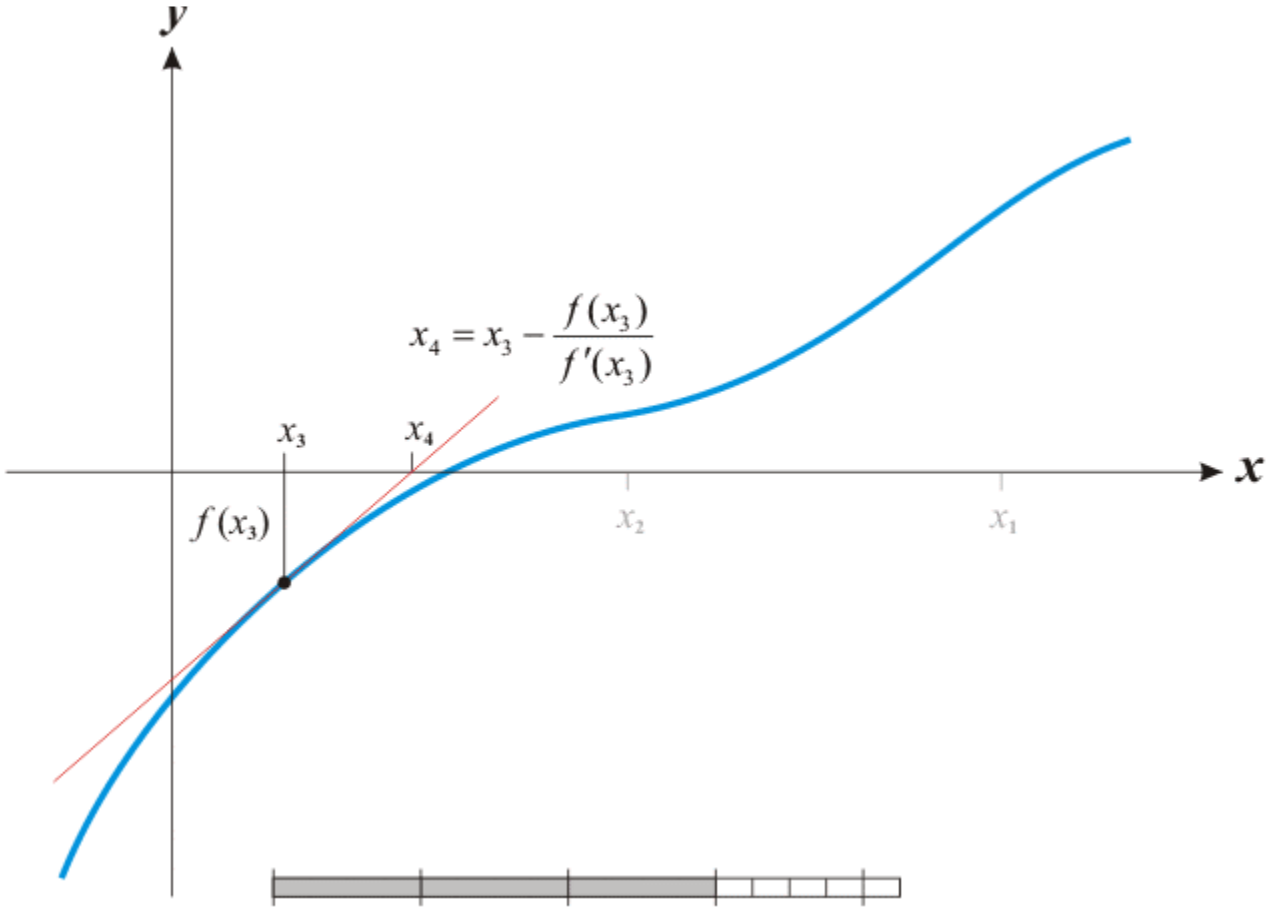
# Newton's Method



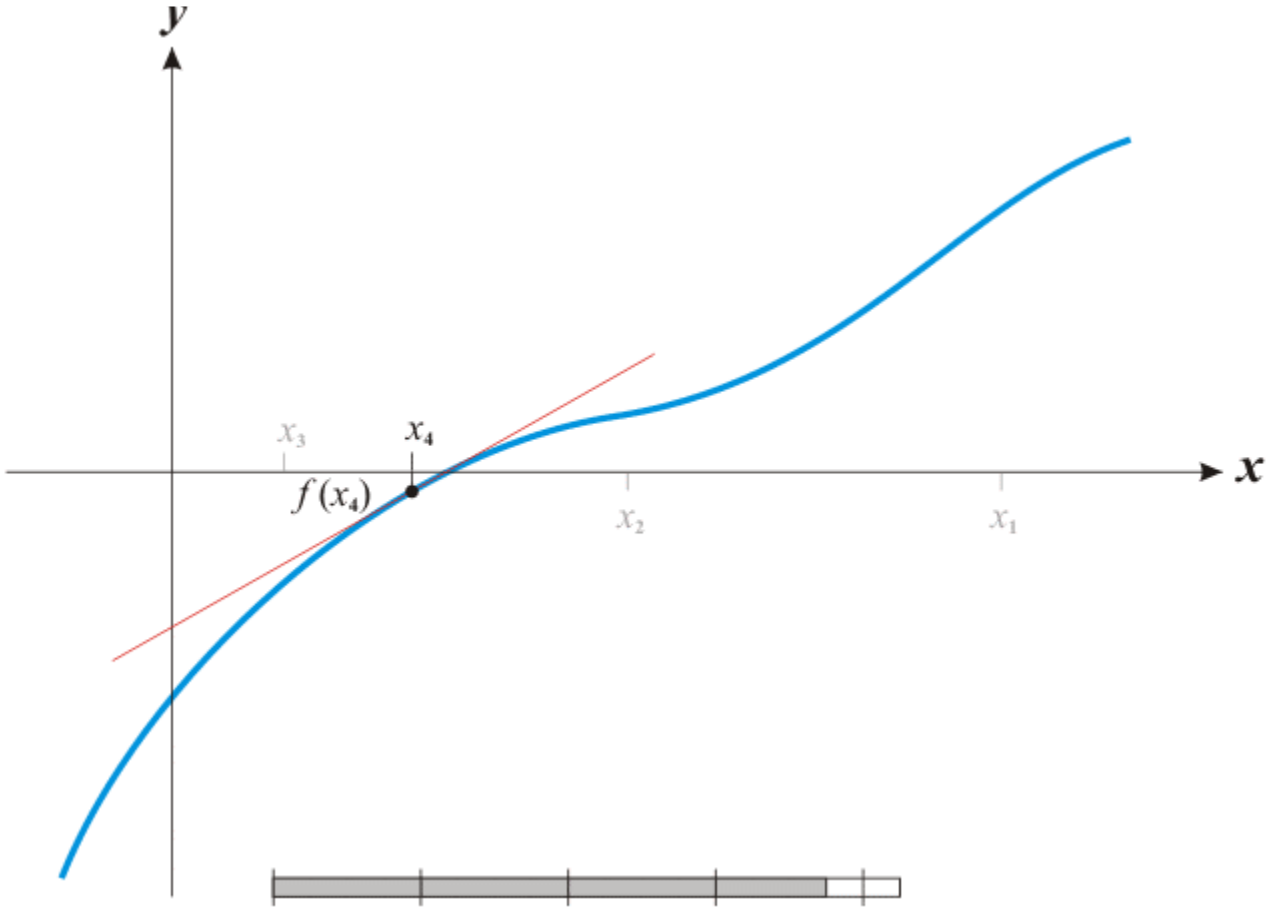
# Newton's Method



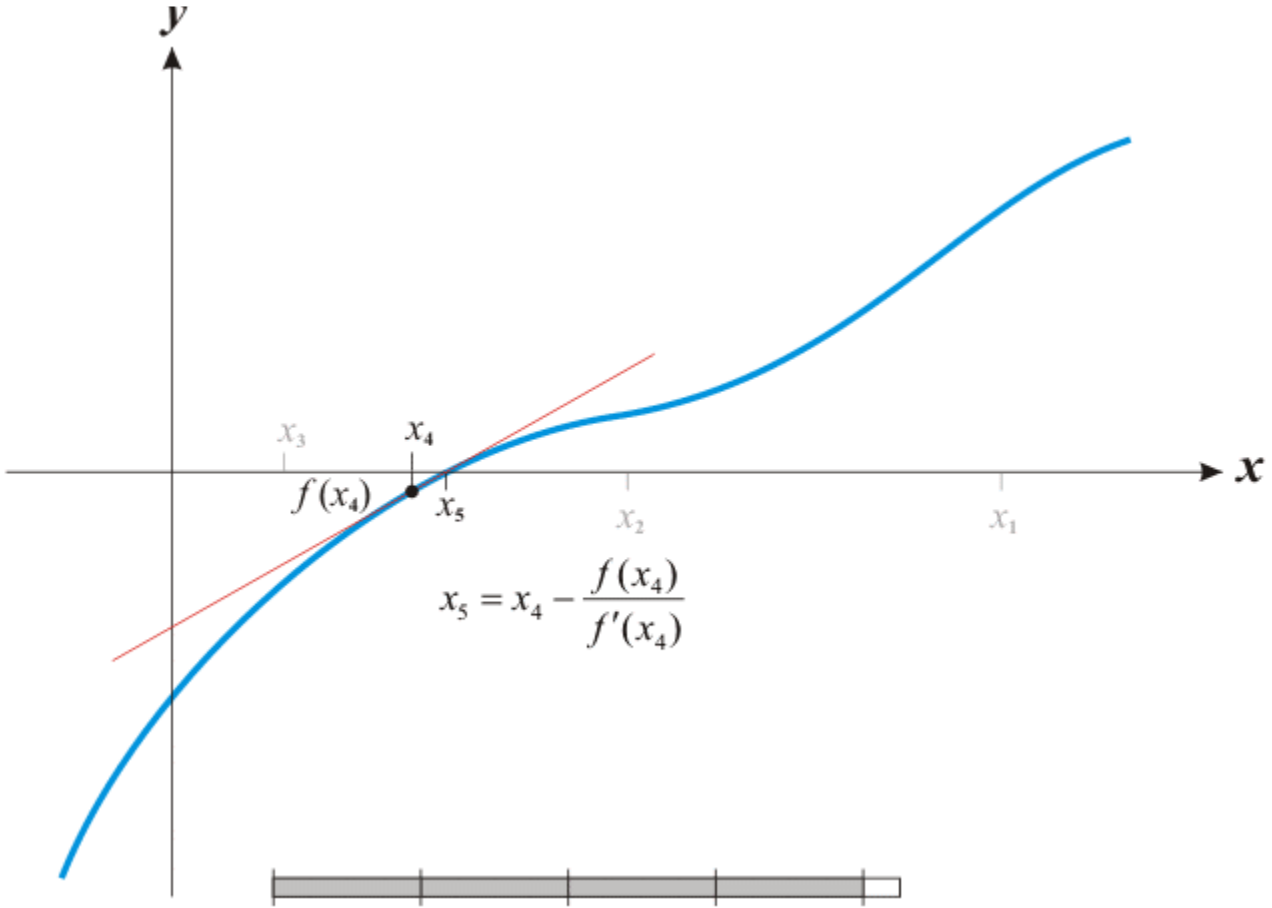
# Newton's Method



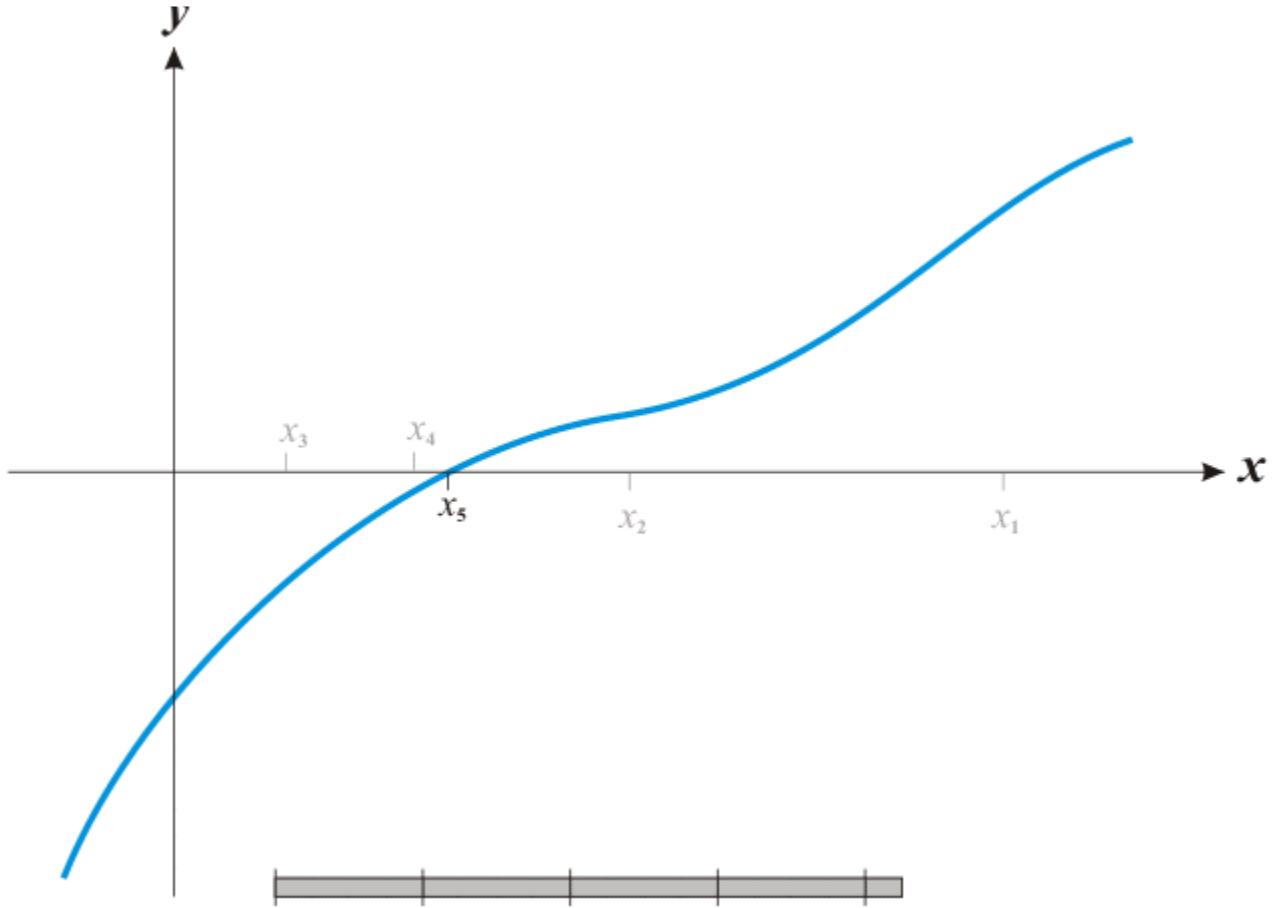
# Newton's Method



# Newton's Method



# Newton's Method



# Newton's Method in N-dimensions

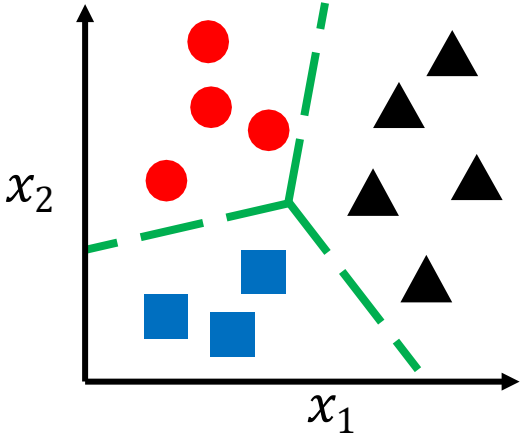
- Recall that in n-dimensions, we replace single-variable derivatives with a vector of partial derivatives called the **gradient**.

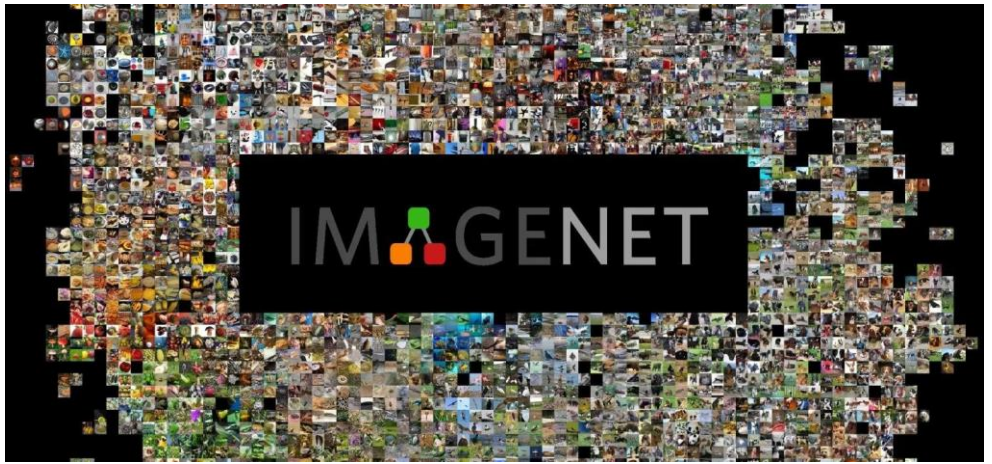
- Thus, our update rule, in its multivariate form, becomes our vector of parameters $\hat{x}$, minus the scalar $f(\hat{x})$ multiplied by the inverse of the gradient vector:

$$\hat{x}_{n+1} = \hat{x}_n - f(\hat{x}_n) * \nabla f(\hat{x}_n)^{-1}$$

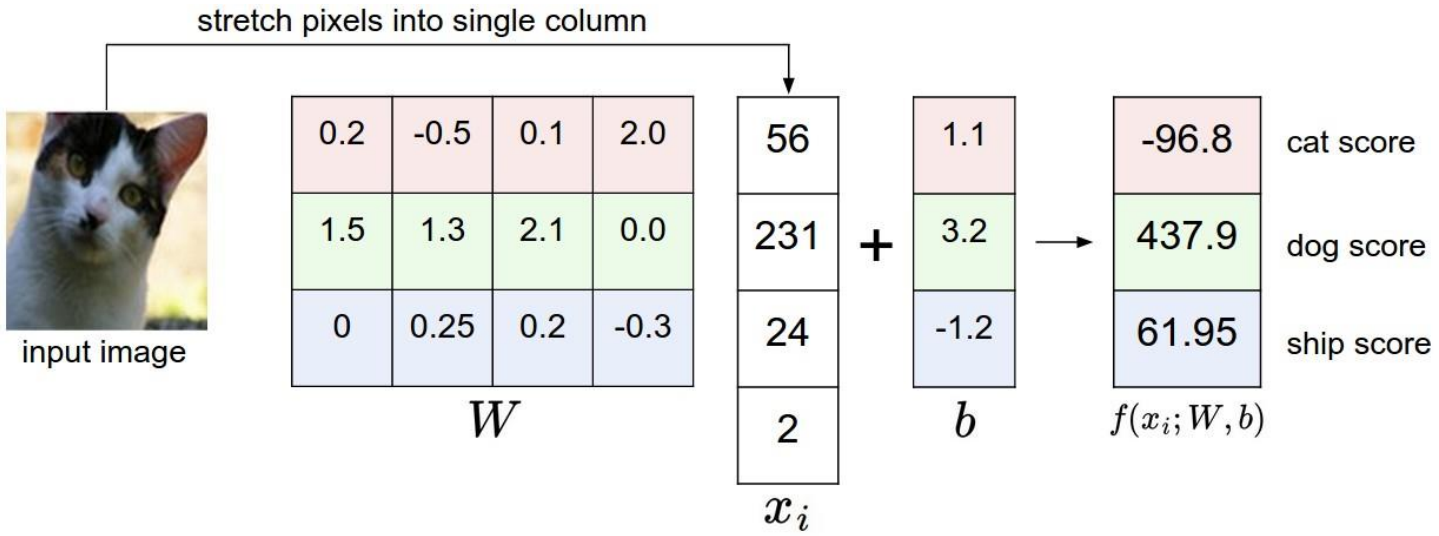
# Multi-Class Classification

- What about more than two classes?
  - **Disease diagnosis:** healthy, cold, flu, pneumonia
  - **Object classification:** desk, chair, monitor, bookcase
  - In general, consider a finite space of labels $\mathcal{Y}$



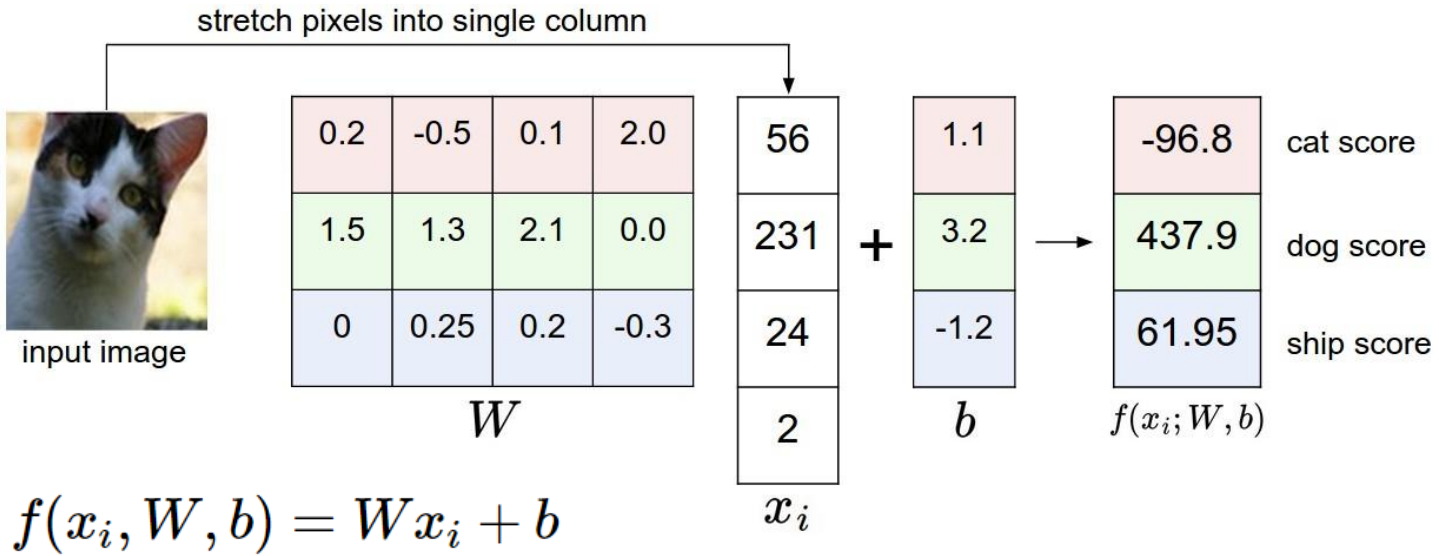



# Linear Classifiers – More than 2 Class




Source: cs231n, http://cs231n.github.io/linear-classify/

# Linear Classifiers – More than 2 Class




Source: cs231n, http://cs231n.github.io/linear-classify/

# Linear Classifiers – More than 2 Class

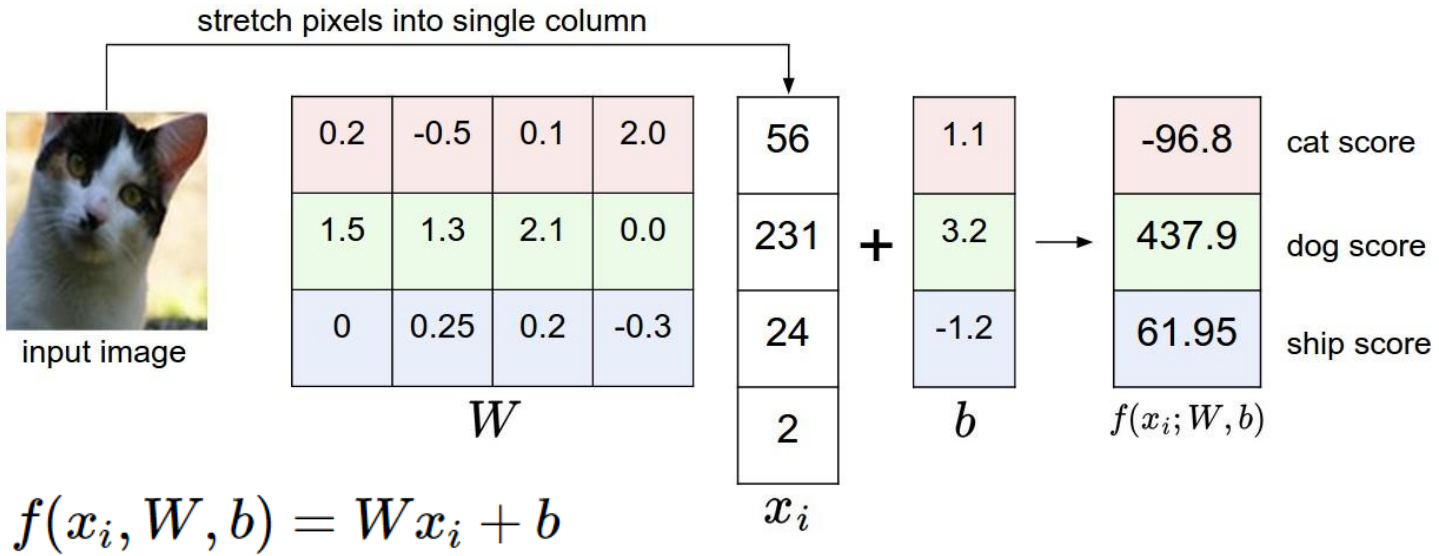


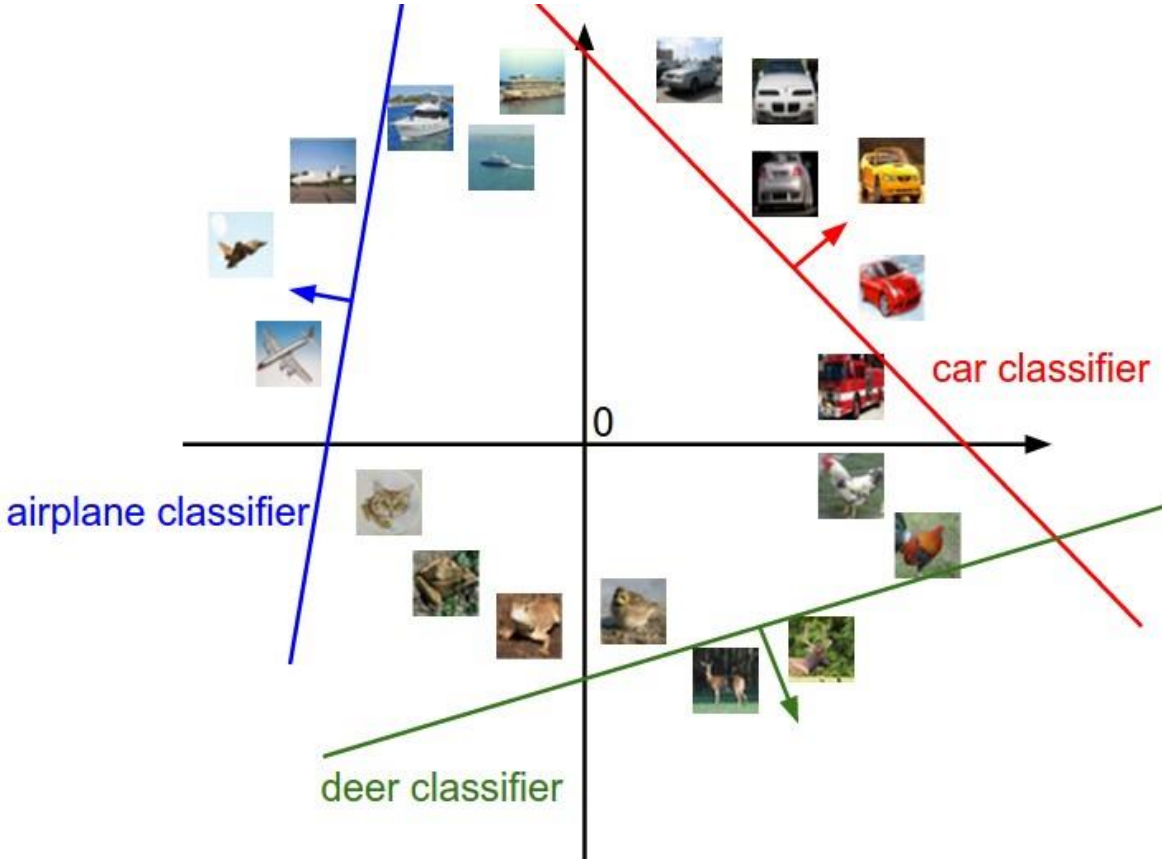

Image $x_i$ has all of its pixels flattened out to a single column vector of shape [D x 1].

Matrix **W** (of size [K x D]), and vector **b** (of size [K x 1]) are the **parameters**.

**K** is the number of classes.

Source: cs231n, http://cs231n.github.io/linear-classify/

# Analogy of Images as High-dimensional Points




Source: cs231n, http://cs231n.github.io/linear-classify/

# Interpretation: Linear Classifiers as Template Matching


Example learned weights at the end of learning for CIFAR-10. Note that, for example, the ship template contains a lot of blue pixels as expected. This template will therefore give a high score once it is matched against images of ships on the ocean with an inner product.

Source: cs231n, http://cs231n.github.io/linear-classify/

# Recall: Logistic Regression

- Consider the following choice:

$$p_{\mathrm{w}}(Y=0 \mid x_i) \propto e^{-\frac{\mathrm{w}^{\mathsf{T}} x_i}{2}} \text{ and } p_{\mathrm{w}}(Y=1 \mid x_i) \propto e^{\frac{\mathrm{w}^{\mathsf{T}} x_i}{2}}$$

- Then, we have

**Sigmoid function**

$$\sigma(z) = \frac{1}{1+e^{-z}}$$

$$p_{\mathrm{w}}(Y=1 \mid x_i) = \frac{e^{\frac{\mathrm{w}^{\mathsf{T}} x_i}{2}}}{e^{\frac{\mathrm{w}^{\mathsf{T}} x_i}{2}} + e^{-\frac{\mathrm{w}^{\mathsf{T}} x_i}{2}}} = \frac{1}{1+e^{-\mathrm{w}^{\mathsf{T}} x_i}}$$

- Furthermore, $p_{\mathrm{w}}(Y=0 \mid x_i) = 1 - \sigma(\mathrm{w}^{\mathsf{T}} x_i)$

# Multi-Class Logistic Regression

- **Strategy:** Include separate $w_y$ for each label $y \in \mathcal{Y} = \{1, \ldots, k\}$
- Let $p_w(y \mid x) \propto e^{w_y^T x}$, i.e.

$$p_w(y \mid x) = \frac{e^{w_y^T x}}{\sum_{y' \in \mathcal{Y}} e^{w_{y'}^T x}}$$

- We define $\text{softmax}(z_1, \ldots, z_k) = \left[ \frac{e^{z_1}}{\sum_{i=1}^k e^{z_i}} \quad \cdots \quad \frac{e^{z_k}}{\sum_{i=1}^k e^{z_i}} \right]$

- Thus, sometimes called **softmax regression**

# Multi-Class Logistic Regression

- Objective function: Negative Log Likelihood

$$\ell(\mathrm{w}; Z) = -\sum_{i=1}^{n} \log \frac{e^{\mathrm{w}_{y_i}^{\mathsf{T}} x_i}}{\sum_{y' \in y} e^{\mathrm{w}_{y'}^{\mathsf{T}} x_i}} = -\sum_{i=1}^{n} \left[ \mathrm{w}_{y_i}^{\mathsf{T}} x_i - \log \sum_{y' \in y} e^{\mathrm{w}_{y'}^{\mathsf{T}} x_i} \right]$$

# Multi-Class Logistic Regression

- **Define One-hot vector:** $\hat{\boldsymbol{y}}_{\boldsymbol{ik}} \in \{\mathbf{0}, \mathbf{1}\}, \sum_{\boldsymbol{k}} \hat{\boldsymbol{y}}_{\boldsymbol{ik}} = \mathbf{1}, \hat{\boldsymbol{y}}_{\boldsymbol{iy}_i} = \mathbf{1}$
- Then loss will become:

$$\ell(w;\, Z) = -\sum_i \sum_k \hat{\boldsymbol{y}}_{\boldsymbol{ik}} \log p_{ik}$$

where: $p_{ik} = \dfrac{e^{w_k^T x_i}}{\sum_{k' \in y} e^{w_{k'}^T x_i}}$

# Multi-Class Logistic Regression

- **Define One-hot vector:** $\hat{\boldsymbol{y}}_{\boldsymbol{ik}} \in \{\mathbf{0}, \mathbf{1}\}, \sum_{\boldsymbol{k}} \hat{\boldsymbol{y}}_{\boldsymbol{ik}} = \mathbf{1}, \hat{\boldsymbol{y}}_{\boldsymbol{iy}_i} = \mathbf{1}$
- Then loss will become:

$$\ell(w;\, Z) = -\sum_i \sum_k \hat{\boldsymbol{y}}_{\boldsymbol{ik}} \log p_{ik}$$

where: $p_{ik} = \dfrac{e^{\mathrm{w}_k^{\mathrm{T}} x_i}}{\sum_{k' \in y} e^{\mathrm{w}_{k'}^{\mathrm{T}} x_i}}$

*Let K=2, then it will fall back to the form of BCE!*

- Then loss will become:

$$\ell\,(\mathrm{w}; Z) = -\sum_i \; y_i \log p_i + (1 - y_i) \log(1 - p_i)$$

where:

$$p_i = \frac{e^{\mathrm{w}_0^{\mathrm{T}} x_i}}{e^{\mathrm{w}_0^{\mathrm{T}} x_i} + e^{\mathrm{w}_1^{\mathrm{T}} x_i}} = \sigma((\mathrm{w}_0^{\mathrm{T}} - \mathrm{w}_1^{\mathrm{T}}) x_i)$$

# Multi-Class Logistic Regression

$$p_{ik} = \frac{e^{w_k^T x_i}}{\sum_{k' \in y} e^{w_{k'}^T x_i}}$$

- Gradient Derivation:

$$\ell(w; Z) = -\sum_i \sum_k \hat{\boldsymbol{y}}_{\boldsymbol{ik}} \log p_{ik} = -\sum_{i=1}^{n} \left[ \sum_{k \in y} \left[ \hat{y}_{ik} w_k^T x_i - \log \sum_{k' \in y} w_{k'}^T x_i \right] \right]$$

- We can derive gradients for both terms:

$$\nabla_{w_k} \ell(w; Z) = -\sum_{i=1}^{n} \hat{y}_{ik} x_i - \frac{e^{w_k^T x_i}}{\sum_{k' \in y} e^{w_{k'}^T x_i}} x_i = -\sum_{i=1}^{n} (p_{ik} - \hat{y}_{ik}).\ x_i$$

# Multi-Class Logistic Regression

$$p_{ik} = \frac{e^{w_k^T x_i}}{\sum_{k' \in y} e^{w_{k'}^T x_i}}$$

- Gradient Derivation:

$$\ell(w; Z) = -\sum_i \sum_k \hat{\boldsymbol{y}}_{\boldsymbol{ik}} \log p_{ik} = -\sum_{i=1}^{n} \left[ \sum_{k \in y} \left[ \hat{y}_{ik} w_k^T x_i - \log \sum_{k' \in y} w_{k'}^T x_i \right] \right]$$

- We can derive gradients for both terms:

Probability Label Input

$$\nabla_{w_k} \ell(w; Z) = -\sum_{i=1}^{n} \hat{y}_{ik} x_i - \frac{e^{w_k^T x_i}}{\sum_{k' \in y} e^{w_{k'}^T x_i}} x_i = -\sum_{i=1}^{n} (p_{ik} - \hat{y}_{ik}). \, x_i$$

# Multi-Class Logistic Regression

$$p_{ik} = \frac{e^{w_k^T x_i}}{\sum_{k' \in y} e^{w_{k'}^T x_i}}$$

- Gradient Derivation:

$$\ell(w; Z) = -\sum_i \sum_k \hat{\boldsymbol{y}}_{\boldsymbol{ik}} \log p_{ik} = -\sum_{i=1}^{n} \left[ \sum_{k \in \mathcal{Y}} \left[ \hat{y}_{ik} w_k^T x_i - \log \sum_{k' \in \mathcal{Y}} w_{k'}^T x_i \right] \right]$$

- We can derive gradients for both terms:

Probability Label Input

$$\nabla_{w_k} \ell(w; Z) = -\sum_{i=1}^{n} \hat{y}_{ik} x_i - \frac{e^{w_k^T x_i}}{\sum_{k' \in \mathcal{Y}} e^{w_{k'}^T x_i}} x_i = -\sum_{i=1}^{n} (p_{ik} - \hat{y}_{ik}). \, x_i$$

- It has a very similar form as in binary case:

$$\nabla_w \ell(w; Z) = -\sum_{i=1}^{n} (\sigma(w^T x_i) - y_i) \cdot x_i$$

# Multi-Class Logistic Regression

- **Aside:** This loss is also known as cross-entropy:

$$\ell(w;\, Z) = -\sum_i \sum_k q_{ik} \log p_{ik} = \sum_i H(qi, pi)$$

- **Label smoothing:** $\hat{y}_{ik}$ can be any simplex vector ($q$), not necessarily one-hot
  - $[0, 0, \ldots, 1, \ldots, 0] \rightarrow [\epsilon/K, \epsilon/K, \ldots, 1 - \epsilon + \epsilon/K, \ldots \epsilon/K]$

$$\nabla_{w_k} \ell\,(w;\, Z) = -\sum_{i=1}^{n} (pik - q_{ik}) \,.\, x_i$$

# Multi-Class Logistic Regression

- Model family

$$f_{\mathrm{w}}(x) = \arg\max_{y} p_{\mathrm{w}}(y \mid x) = \arg\max_{y} \frac{e^{\mathrm{w}_y^{\mathsf{T}} x}}{\sum_{k' \in y} e^{\mathrm{w}_{k'}^{\mathsf{T}} x}} = \arg\max_{y} \mathrm{w}_y^{\mathsf{T}} x$$

- Optimization
  - Gradient descent on Negative Log Likelihood

$$\nabla_{\mathrm{w}_k} \ell(\mathrm{w}; Z) = -\sum_{i=1}^{n} (p_{ik} - \hat{y}_{ik}).\ x_i$$

  - Simultaneously update all parameters $\{\mathrm{w}_y\}_{y \in y}$

# Linear Classifiers

“Parametric” classifier: model defined by a small number of parameters (w, b)

**Pros:**

- Very fast at test time

**Cons:**

- Slow at training time: need to estimate the parameters
- Data may not be linearly separable

# Classification Metrics

- While we minimize the Negative Log Likelihood, we often evaluate using **accuracy**

- However, even accuracy isn't necessarily the "right" metric
  - If 99% of labels are negative (i.e., $y_i = 0$), accuracy of $f_{\mathrm{w}}(x) = 0$ is 99%!
  - For instance, very few patients test positive for most diseases
  - "Imbalanced data"

- What are alternative metrics for these settings?

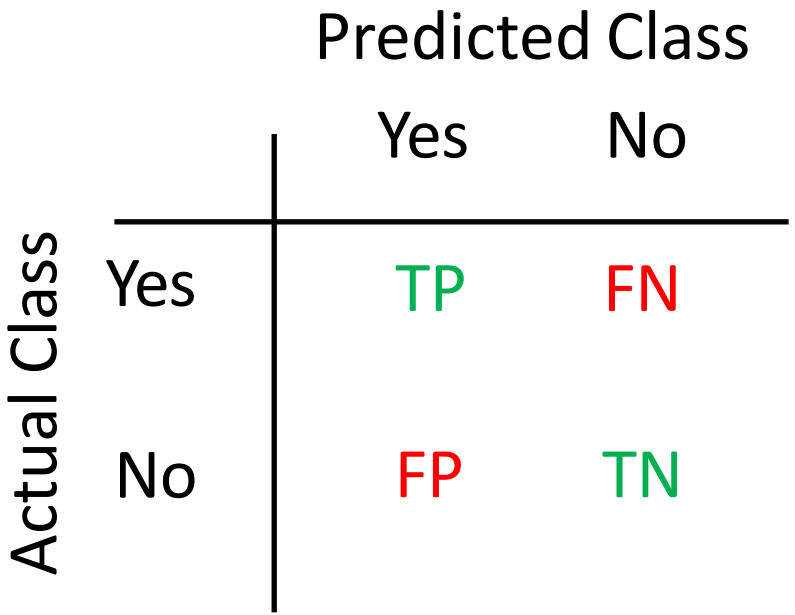
# Classification Metrics

- **Classify test examples as follows:**
  - **True Positive (TP):** Actually positive, predicted positive
  - **False Negative (FN):** Actually positive, predicted negative
  - **True Negative (TN):** Actually negative, predicted negative
  - **False Positive (FP):** Actually negative, predicted positive

- Many metrics expressed in terms of these; for example:

$$\text{accuracy} = \frac{TP + TN}{n} \qquad \text{error} = 1 - \text{accuracy} = \frac{FP + FN}{n}$$

# Confusion Matrix



# Confusion Matrix

| | | Predicted Class: Yes | Predicted Class: No |
|---|---|---|---|
| Actual Class | Yes | 3 TP | 4 FN |
| | No | 6 FP | 37 TN |

Accuracy = 0.8

# Classification Metrics

- For imbalanced metrics, we roughly want to disentangle:
  - Accuracy on “positive examples”
  - Accuracy on “negative examples”

- Different definitions are possible (and lead to different meanings)!

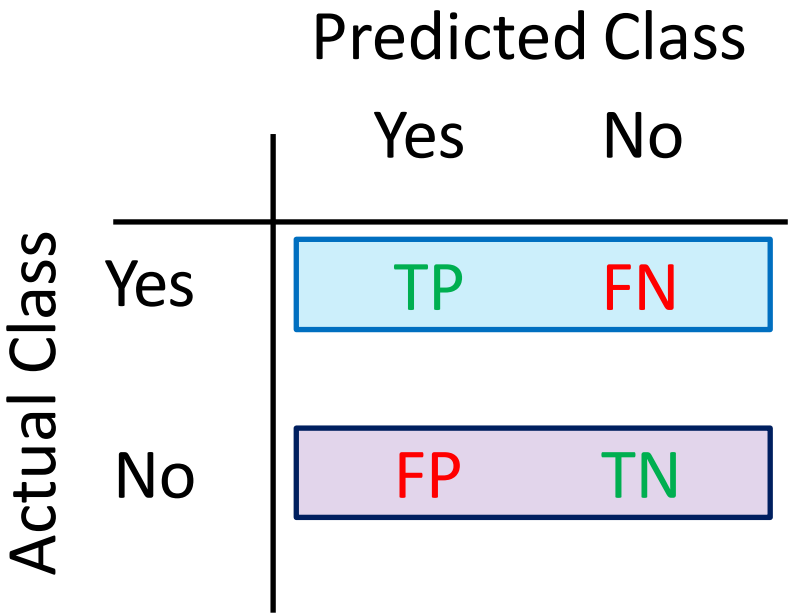
# Sensitivity & Specificity

- **Sensitivity:** What fraction of **actual positives** are **predicted positive**?
  - **Good sensitivity:** If you have the disease, the test correctly detects it
  - Also called **true positive rate**

- **Specificity:** What fraction of **actual negatives** are **predicted negative**?
  - **Good specificity:** If you do not have the disease, the test says so
  - Also called **true negative rate**

- Commonly used in medicine

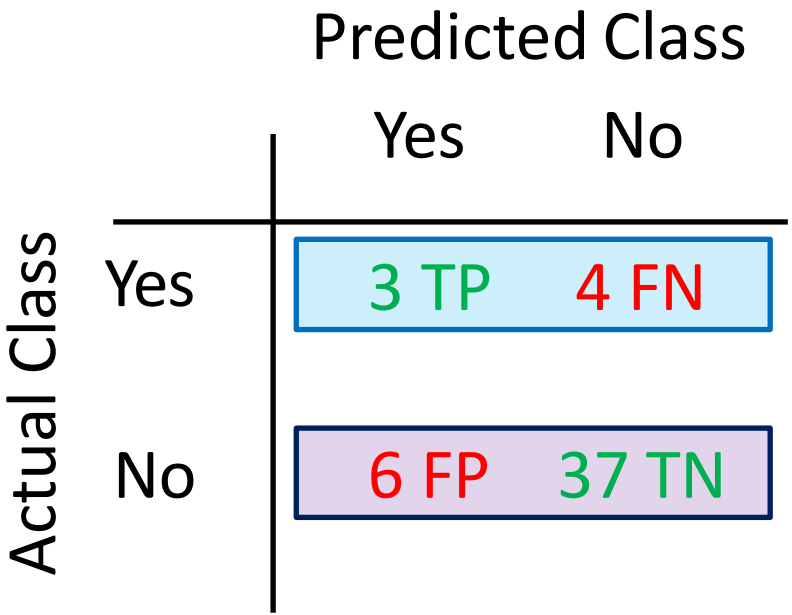
# Sensitivity & Specificity




$$\text{sensitivity} = \frac{TP}{TP + FN}$$

$$\text{specificity} = \frac{TN}{TN + FP}$$

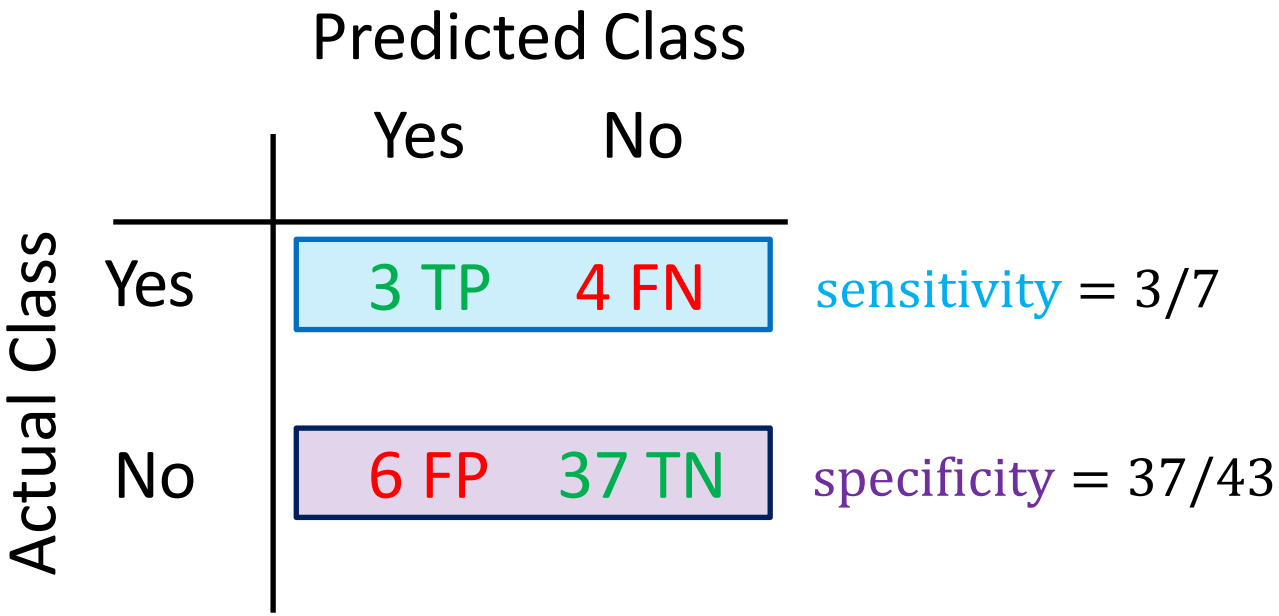
# Sensitivity & Specificity




$$\text{sensitivity} = \frac{TP}{TP + FN}$$

$$\text{specificity} = \frac{TN}{TN + FP}$$

# Sensitivity & Specificity



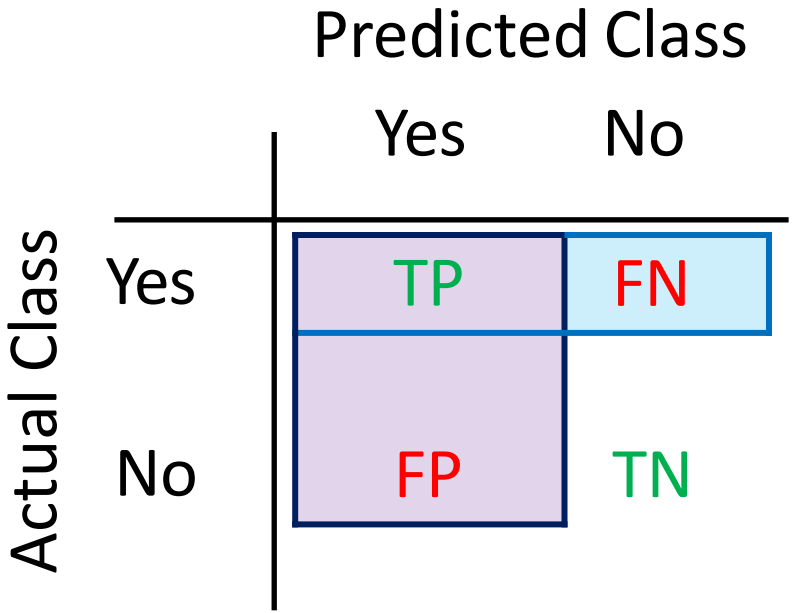
# Precision & Recall

- **Recall:** What fraction of **actual positives** are **predicted positive**?
  - **Good recall:** If you have the disease, the test correctly detects it
  - Also called the **true positive rate** (and sensitivity)

- **Precision:** What fraction of **predicted positives** are **actual positives**?
  - **Good precision:** If the test says you have the disease, then you have it
  - Also called **positive predictive value**

- Used in information retrieval, NLP

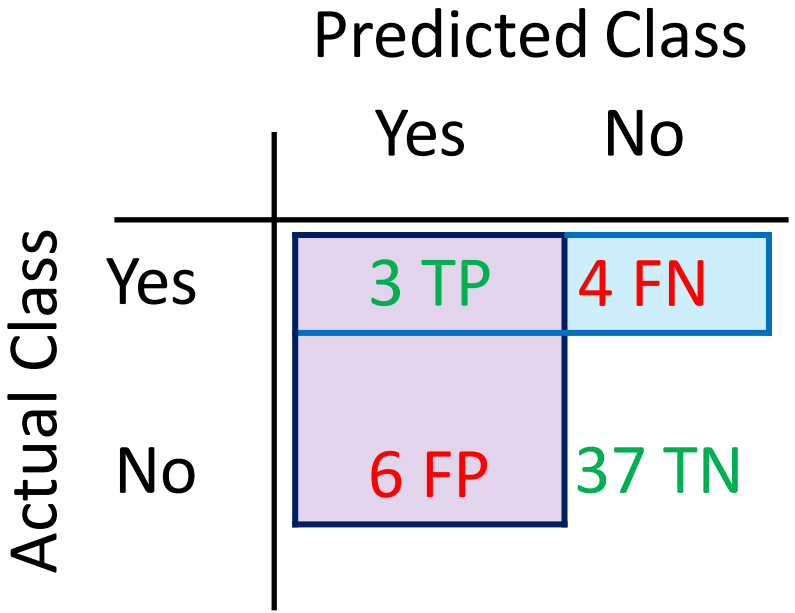
# Precision & Recall




$$\text{recall} = \frac{TP}{TP + FN}$$

$$\text{precision} = \frac{TP}{TP + FP}$$

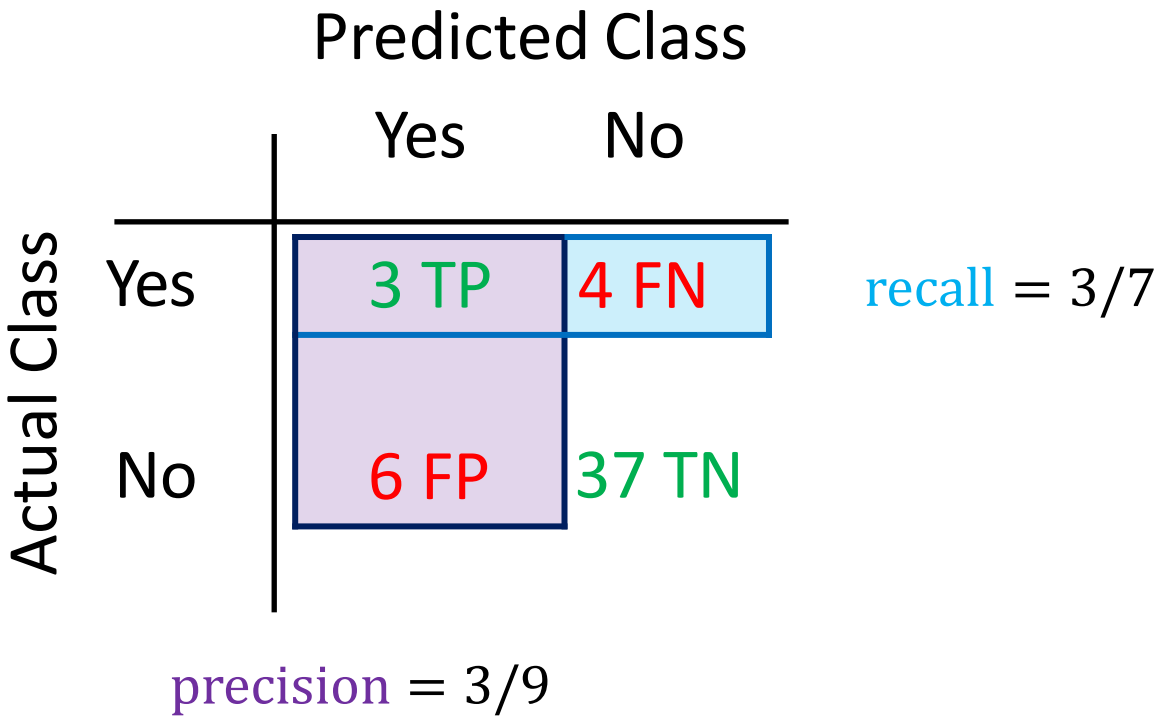
# Precision & Recall




$$\text{recall} = \frac{TP}{TP + FN}$$

$$\text{precision} = \frac{TP}{TP + FP}$$

# Precision & Recall



# Classification Metrics

- **How to obtain a single metric?**
  - Combination, e.g., $F_1\ \text{Score} = \dfrac{2.\,\text{precision}.\,\text{recall}}{\text{precision} + \text{recall}}$ is the harmonic mean

- **How to choose the "right" metric?**
  - No generally correct answer
  - Depends on the goals for the specific problem/domain

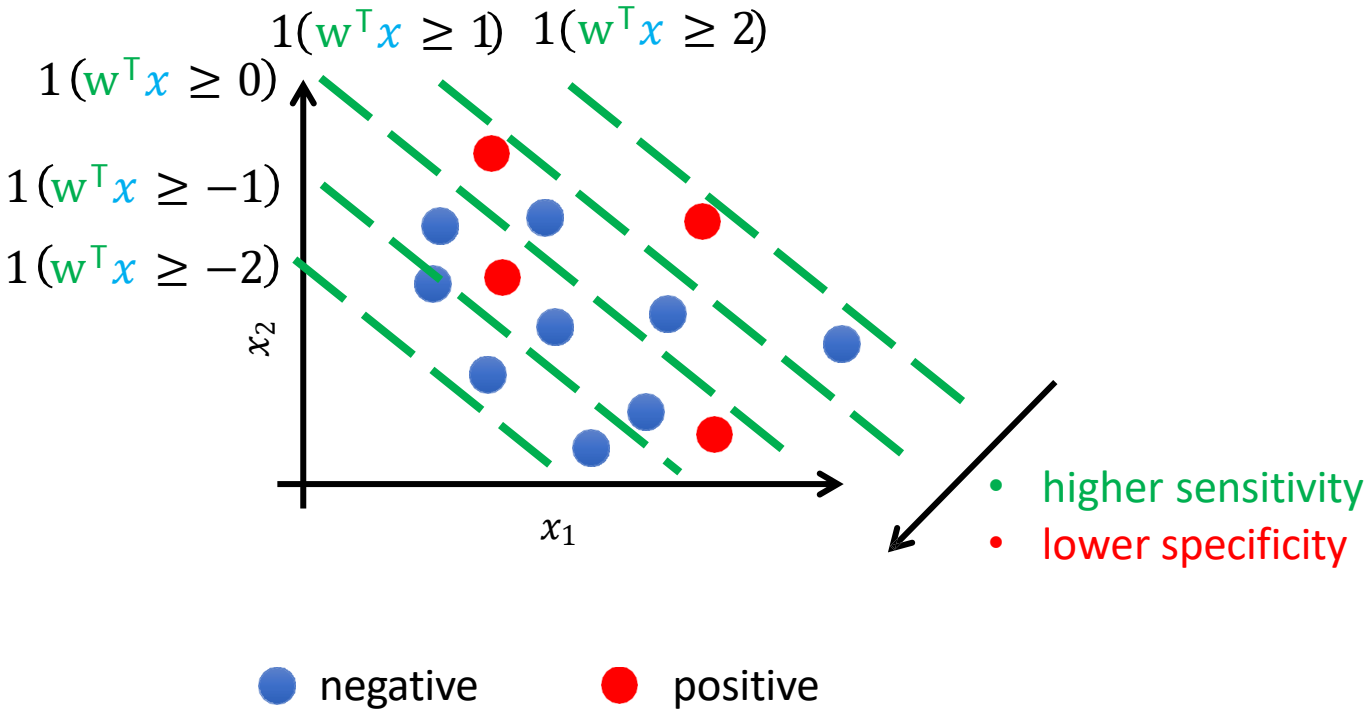
# Optimizing Prediction Threshold

- Consider hyperparameter $\tau$ for the threshold:

$$f_w(x) = 1(w^T x \geq 0)$$

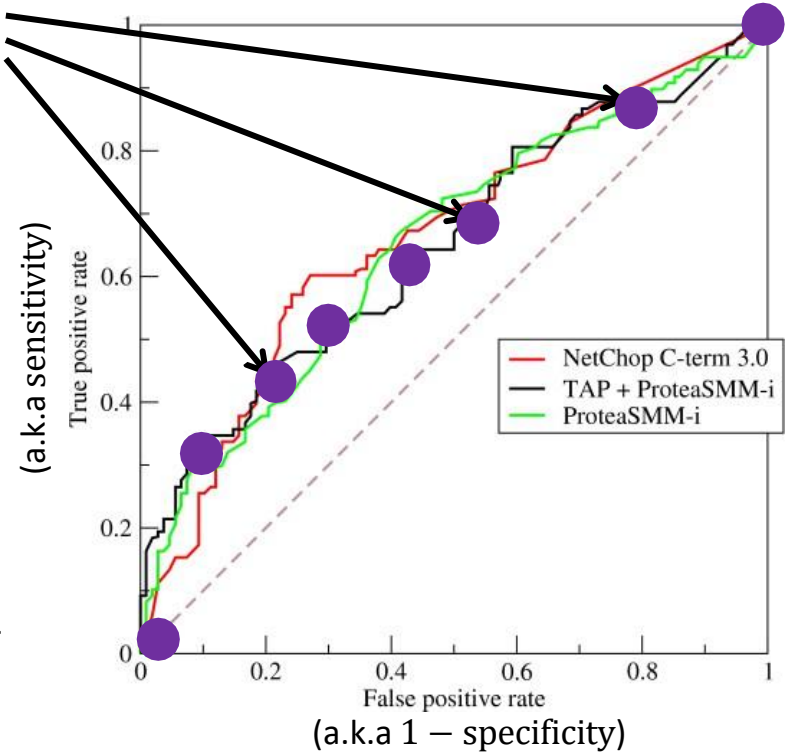
$$f_w(x) = 1(w^T x \geq \tau)$$

# Optimizing Prediction Threshold



# Visualization: ROC Curve

Each point on this curve corresponds to a choice of $\tau$

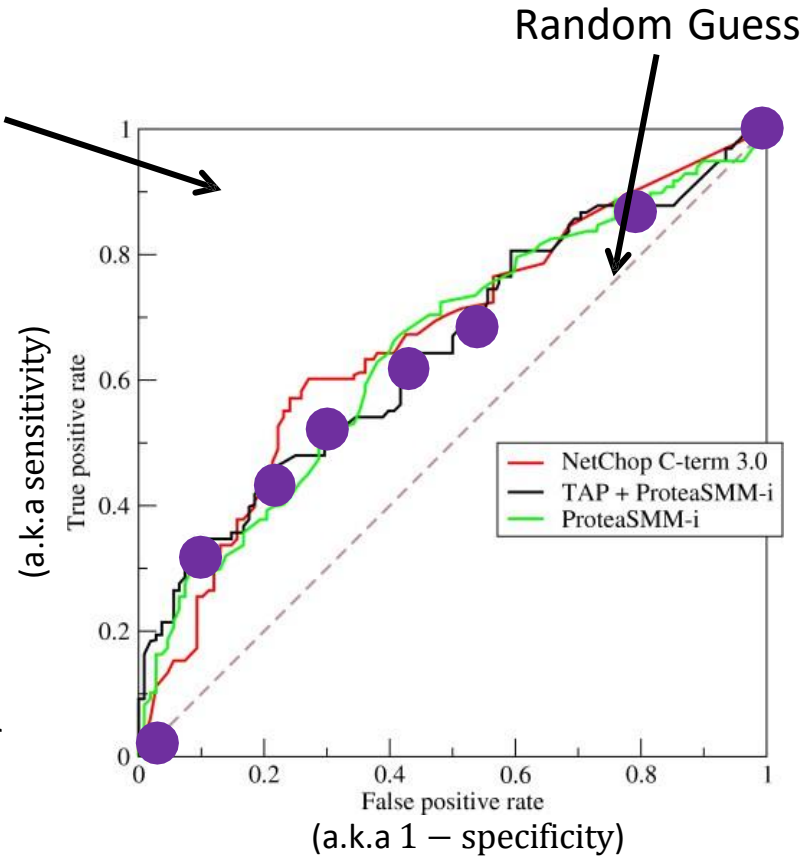



ROC = Receiver Operating Characteristic

**AUC-ROC:** Area under ROC curve is another metric people consider when evaluating $\hat{w}(z)$

# Visualization: ROC Curve

Better Area (High TPR, low FPR)




ROC = Receiver Operating Characteristic

**AUC-ROC:** Area under ROC curve is another metric people consider when evaluating $\hat{w}(z)$

# Acknowledgement

Most of the slides are based on the ML course of:

- CS231n, Stanford University
- Matt Barnes and Matt Golub, University of Washington
- Mingmin Zhao and Jiatao Gu, University of Pennsylvania

# THANK YOU