CSC311: Linear Algebra + Midterm Review

Anastasia Razdaibiedina

October 10, 2019

Based on slides from Sargur Srihari, James Lucas and others

Linear Algebra basics

Symmetric matrix: $A^T = A$

Diagonal matrix: all elements are 0, except for diagonal elements. If diagonal elements are ones, matrix is called ***identity matrix***.

Inverse matrix: $AA^{-1} = A^{-1}A = I$

L2 norm: $||x||_2 = \sqrt{\sum_i x_i^2}$

Diagonal matrix

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 8 & 0 \\ 0 & 0 & 4 \end{bmatrix}$$

Symmetric matrix

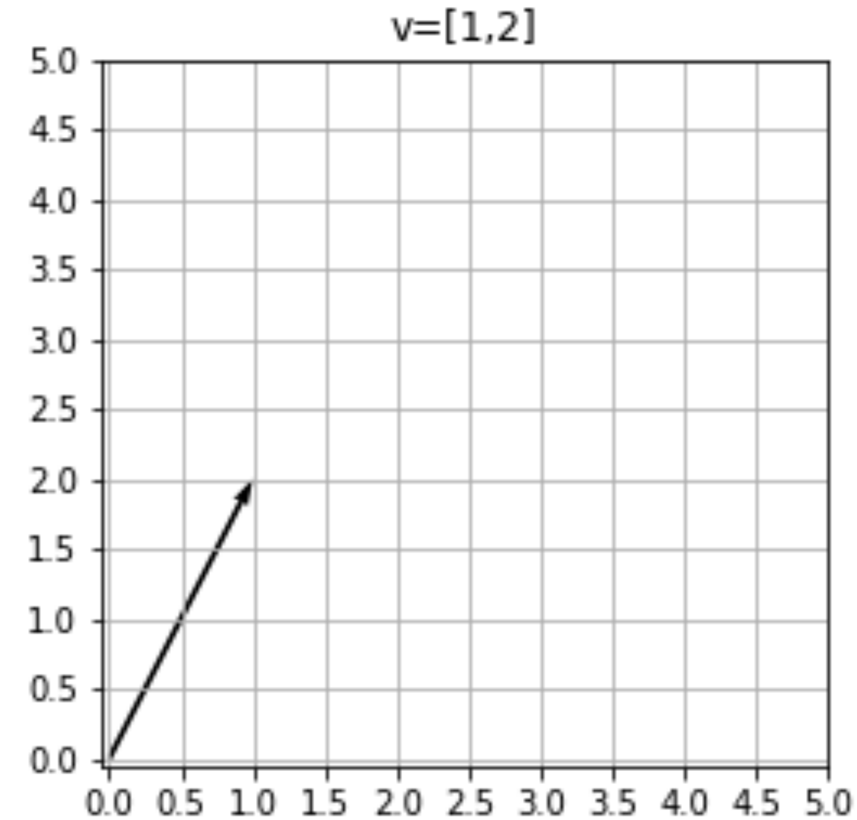
$$\begin{bmatrix} 1 & 2 & 6 \\ 2 & 8 & 4 \\ 6 & 4 & 5 \end{bmatrix}$$

$$|| \begin{bmatrix} 1 \\ 3 \end{bmatrix} ||_2 = \sqrt{1+9} = \sqrt{10}$$

Eigenvalues and eigenvectors

- What can happen to a vector if you multiply it by matrix?

$$\vec{v} = \begin{bmatrix} 1 \\ 2 \end{bmatrix} \quad A = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \quad A\vec{v} = ?$$

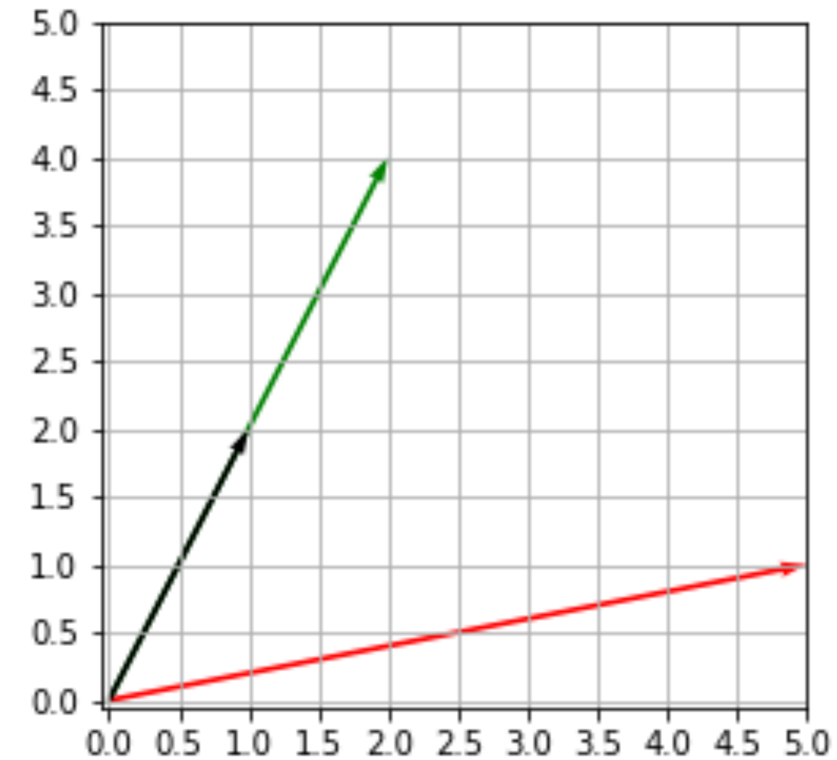


- If A is identity matrix, then

$$A\vec{v} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \vec{v} = \vec{v}$$

- Let's plug in more matrices:

$$\begin{bmatrix} 1 & 2 \\ 0 & 0.5 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \end{bmatrix} = \begin{bmatrix} 5 \\ 1 \end{bmatrix} \quad \begin{bmatrix} 1 & 0.5 \\ 6 & -1 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \end{bmatrix} = \begin{bmatrix} 2 \\ 4 \end{bmatrix}$$



When multiplied by a matrix, vector

A. can be stretched

B. can be rotated

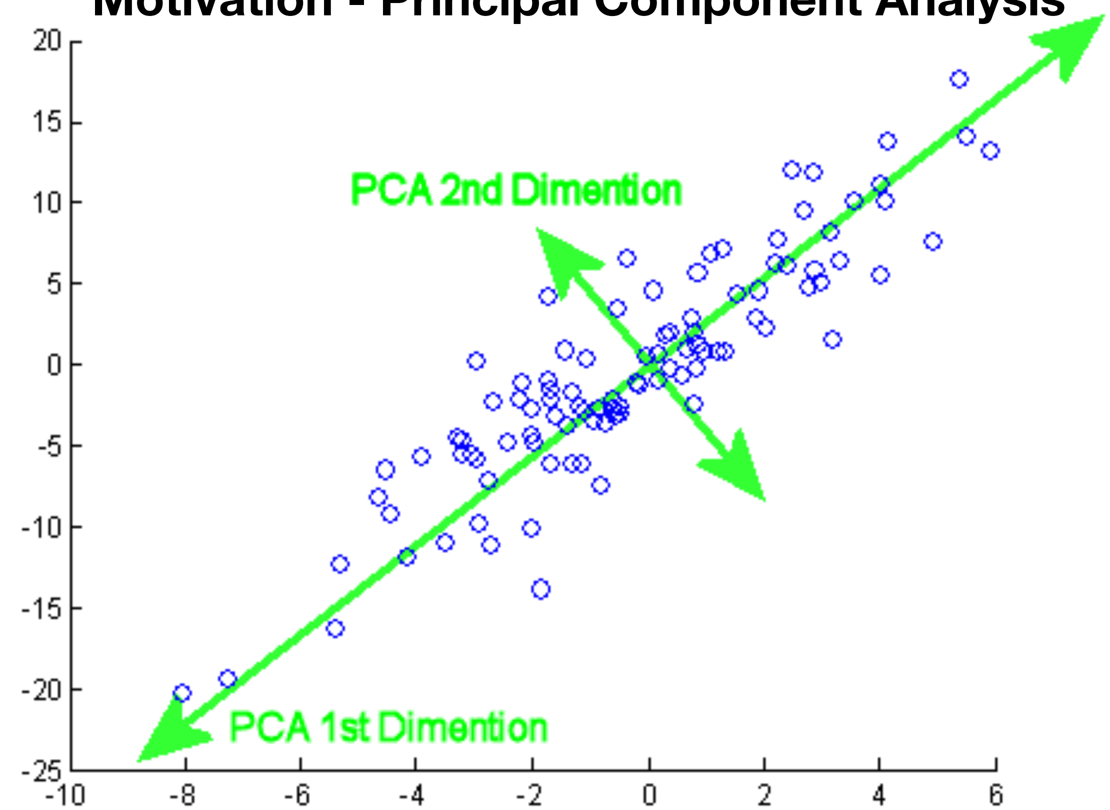
Eigenvalues and eigenvectors

- Definition: vector $\vec{v}$ is called ***eigenvector*** of matrix A and scalar λ is called ***eigenvalue*** of matrix A if

$$A\vec{v} = \lambda\vec{v}$$

- So matrix A only performs rescaling of the vector, but not rotation!
- Why is this important?
  - From all unit eigenvectors of matrix A, consider the eigenvector that corresponds to the maximum eigenvalue.
  - This eigenvector specifies the direction along which the action of matrix is maximal.
  - No other vector when acted by matrix A will get stretched as much as this eigenvector.

Motivation - Principal Component Analysis



- Shortly, we can get directions along which matrix has the biggest effect.

Eigendecomposition

- Let A be square nxn matrix with n linearly independent eigenvectors q_i
- Then we have a system of equations:

$$\begin{cases} Aq_1 = \lambda_1 q_1 \\ \dots \\ Aq_n = \lambda_n q_n \end{cases}$$

- In matrix form:

$$A \begin{bmatrix} | & & | \\ q_1 & \dots & q_n \\ | & & | \end{bmatrix} = \begin{bmatrix} | & & | \\ q_1 & \dots & q_n \\ | & & | \end{bmatrix} \begin{bmatrix} \lambda_1 & 0 & \dots & 0 \\ 0 & \lambda_2 & \dots & 0 \\ 0 & 0 & \dots & \lambda_n \end{bmatrix}$$

$$AQ = Q\Lambda$$

$$A = Q\Lambda Q^{-1}$$

eigendecomposition of matrix A

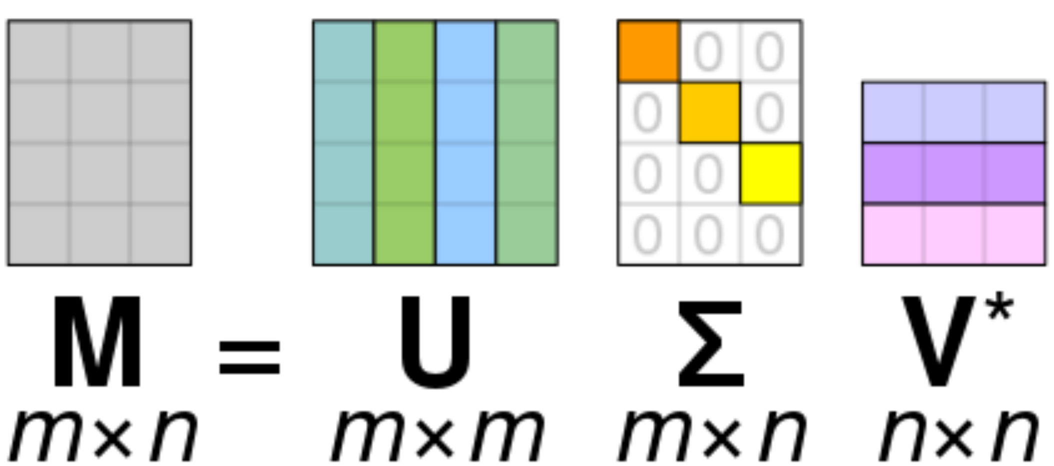
SVD

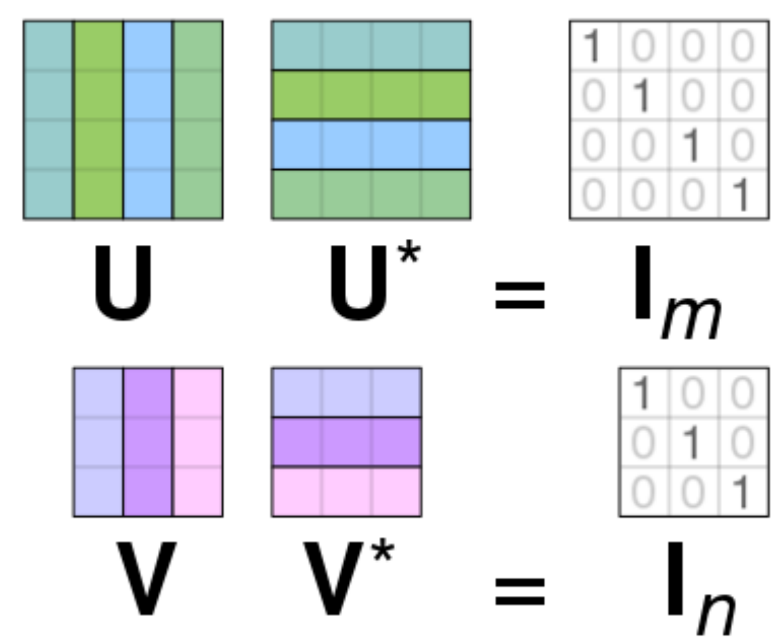
- **SVD** = singular value decomposition

- What if matrix A is not square? Then eigendecomposition cannot be applied.

- SVD is a matrix factorization technique that can be applied to non-square matrices:

$$A = U\Sigma V^T$$

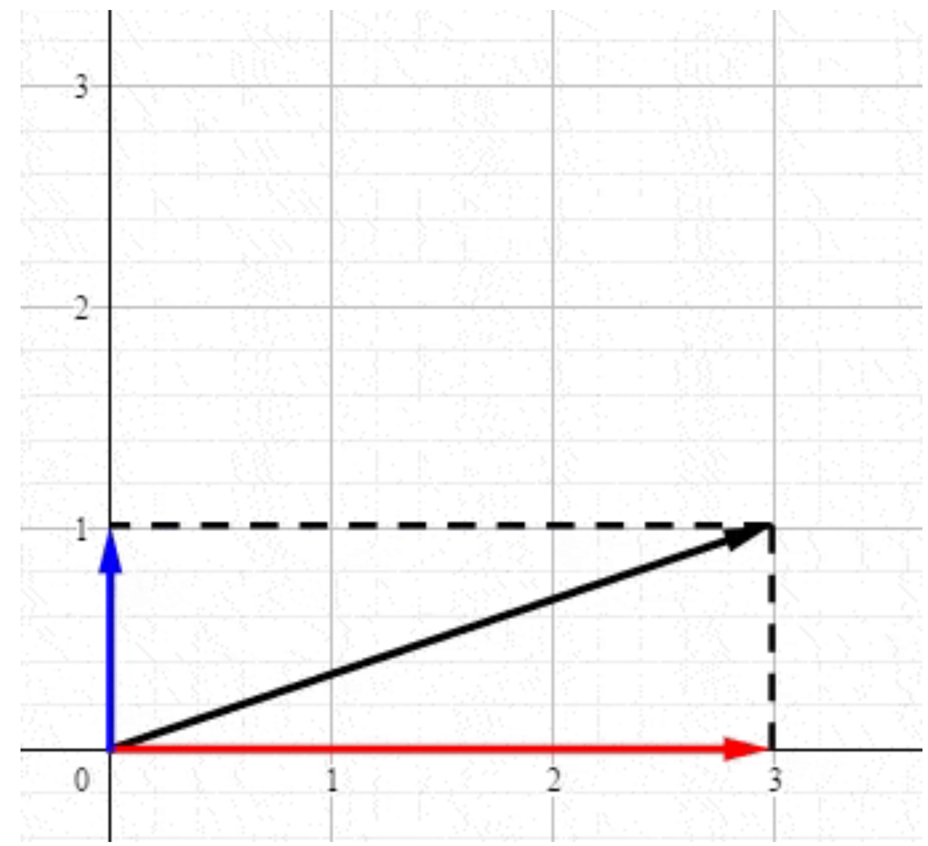




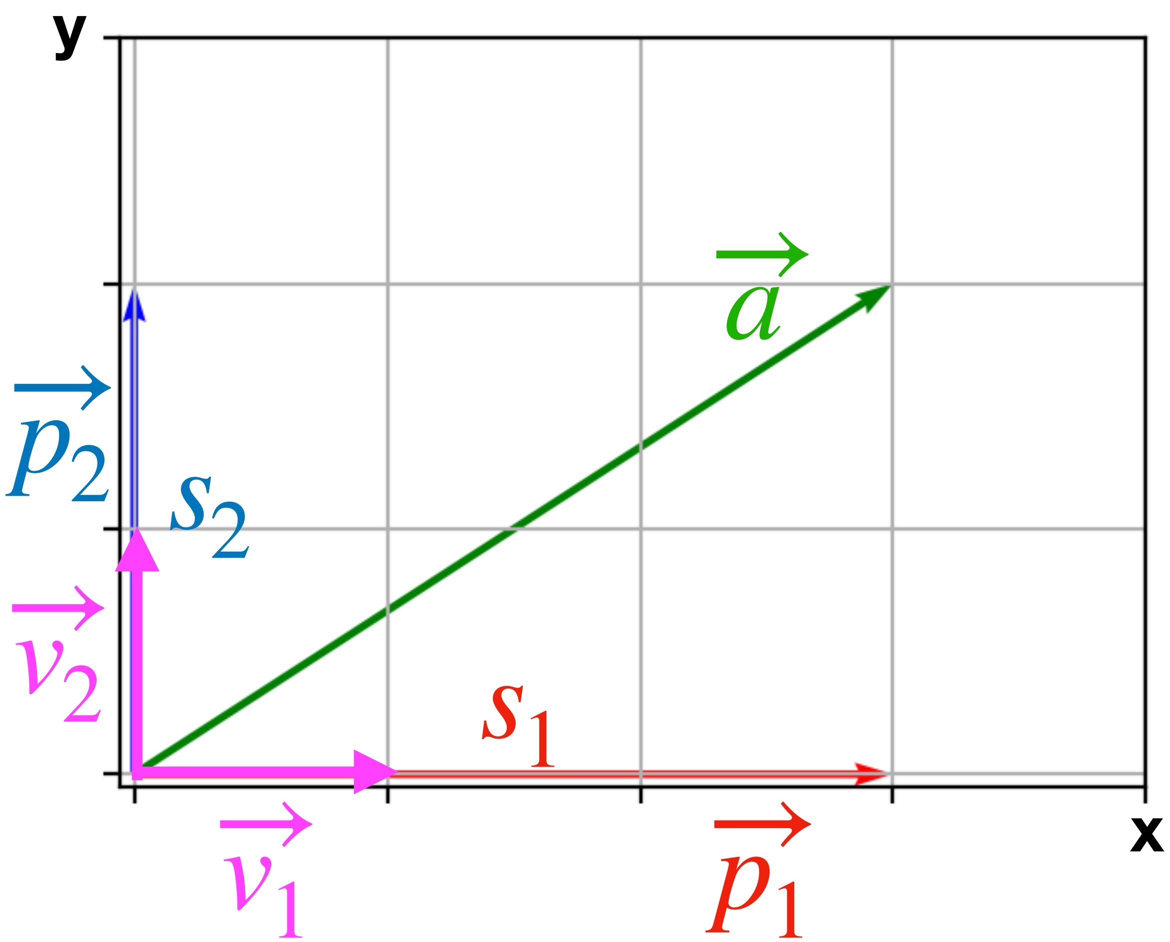
Intuition behind SVD

- Intuition from physics: any force vector can be decomposed into its components along x and y axis.

- SVD is about *decomposing vectors onto orthogonal axes*.



Intuition behind SVD



- Any vector can be expressed in terms of:
    1. projection direction unit vectors ($\vec{v_1}, \vec{v_2}$)
    2. the lengths of projections onto them (s_1, s_2)

$$s_1 = a^T v_1 \quad s_2 = a^T v_2$$

$$s_1 = [3 \quad 1] \begin{bmatrix} 1 \\ 0 \end{bmatrix} = 3$$

$$[a_1 \quad a_2] \begin{bmatrix} | & | \\ v_1 & v_2 \\ | & | \end{bmatrix} = \begin{bmatrix} a^T v_1 & a^T v_2 \end{bmatrix} = [s_1 \quad s_2]$$

$$a^T [v_1 \quad v_2] = [s_1 \quad s_2]$$

Intuition behind SVD

- Case of more vectors:

$$\begin{bmatrix} a_1 & a_2 \\ b_1 & b_2 \end{bmatrix} \begin{bmatrix} | & | \\ v_1 & v_2 \\ | & | \end{bmatrix} = \begin{bmatrix} a^T v_1 & a^T v_2 \\ b^T v_1 & b^T v_2 \end{bmatrix} = \begin{bmatrix} s_{a1} & s_{a2} \\ s_{b1} & s_{b2} \end{bmatrix}$$

$$AV = S$$

$$A \cdot V = S$$

Matrix of points

The dot product performs the projection

Matrix of decomposition axes

Matrix of the lengths of projections

$$A = SV^{-1} = SV^T$$

- In original SVD we had: $A = U\Sigma V^T$
- We're just left to get $S = U\Sigma$

Intuition behind SVD

$$\begin{bmatrix} a_1 & a_2 \\ b_1 & b_2 \end{bmatrix} \begin{bmatrix} | & | \\ v_1 & v_2 \\ | & | \end{bmatrix} = \begin{bmatrix} a^T v_1 & a^T v_2 \\ b^T v_1 & b^T v_2 \end{bmatrix} = \begin{bmatrix} s_{a1} & s_{a2} \\ s_{b1} & s_{b2} \end{bmatrix}$$

$$AV = S$$

$$S = \begin{pmatrix} s_{a1} & s_{a2} \\ s_{b1} & s_{b2} \end{pmatrix}$$

A column vector containing the lengths of projections of each point on the 1st axis *v1*

A column vector containing the lengths of projections of each point on the 2nd axis *v2*

- Look more closely at columns of S
- It turns out that it's best to normalize column vectors of S (make them of unit length).
- This is done by dividing each column vector by its magnitude.

Intuition behind SVD

How can we "divide" matrix to get normalized columns?

Numerical example: $M = \begin{bmatrix} 2 & 3 \\ 2 & 3 \end{bmatrix}$

If we just divide first column by 2, then we surely have to multiply by another matrix to preserve the equality:

$$M = \begin{bmatrix} 1 & 3 \\ 1 & 3 \end{bmatrix} \begin{bmatrix} ? & ? \\ ? & ? \end{bmatrix} = \begin{bmatrix} 2 & 3 \\ 2 & 3 \end{bmatrix}$$

The unknown matrix is just the identity matrix with the first element replaced by the divisor 2:

$$M = \begin{bmatrix} 1 & 3 \\ 1 & 3 \end{bmatrix} \begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 2 & 3 \\ 2 & 3 \end{bmatrix}$$

For the second column: $M = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} 2 & 0 \\ 0 & 3 \end{bmatrix} = \begin{bmatrix} 2 & 3 \\ 2 & 3 \end{bmatrix}$

General idea: divide columns of M by their magnitude and then multiply by a diagonal matrix of magnitudes. We can do the same to S matrix!

Intuition behind SVD

$$S = \begin{pmatrix} s_{a1} & s_{a2} \\ s_{b1} & s_{b2} \end{pmatrix}$$

$$\text{Magnitude of 1st column} = \sigma_1 = \sqrt{(s_{a1})^2 + (s_{b1})^2}$$

$$\text{Magnitude of 2nd column} = \sigma_2 = \sqrt{(s_{a2})^2 + (s_{b2})^2}$$

$$S = \begin{pmatrix} \frac{s_{a1}}{\sigma_1} & \frac{s_{a2}}{\sigma_2} \\ \\ \frac{s_{b1}}{\sigma_1} & \frac{s_{b2}}{\sigma_2} \end{pmatrix} \begin{pmatrix} \sigma_1 & 0 \\ 0 & \sigma_2 \end{pmatrix} = \underbrace{\begin{pmatrix} u_{a1} & u_{a2} \\ u_{b1} & u_{b2} \end{pmatrix}}_{\downarrow \; U} \underbrace{\begin{pmatrix} \sigma_1 & 0 \\ 0 & \sigma_2 \end{pmatrix}}_{\downarrow \; \Sigma}$$

$$A = SV^T = U\Sigma V^T$$

Midterm review

For midterm review we'll go through:

1. Important ML concepts
2. Exercises

ML concepts 1

- **What is supervised learning?**

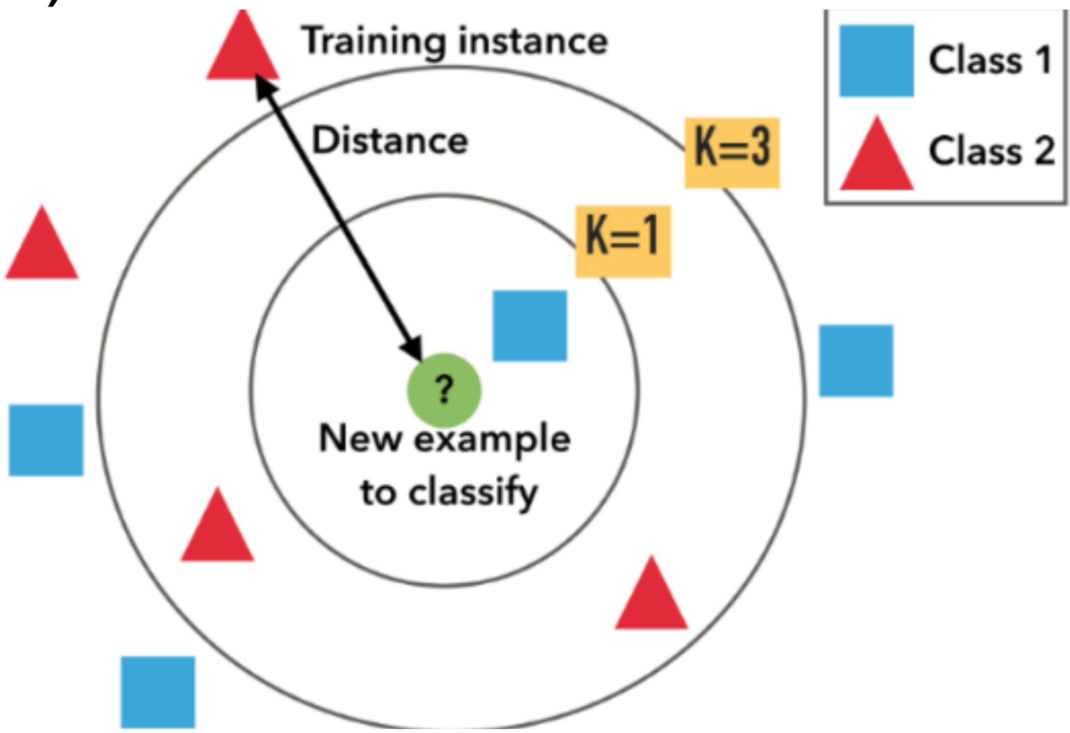
Answer: ML setting when our training set consists of inputs and their corresponding labels.

- **Difference between regression / classification?**

Answer: in ***classification*** we are predicting a discrete target (like cat or dog class), while in ***regression*** we are predicting a continuous-valued target (like temperature).

- **What does kNN do?**

Answer: k Nearest Neighbors is an algorithm that predicts value of a new example based on its k nearest labeled neighbors.



ML concepts 2

- **How does decision tree work?**

Answer: decision trees make predictions by sequentially splitting data on different attributes.

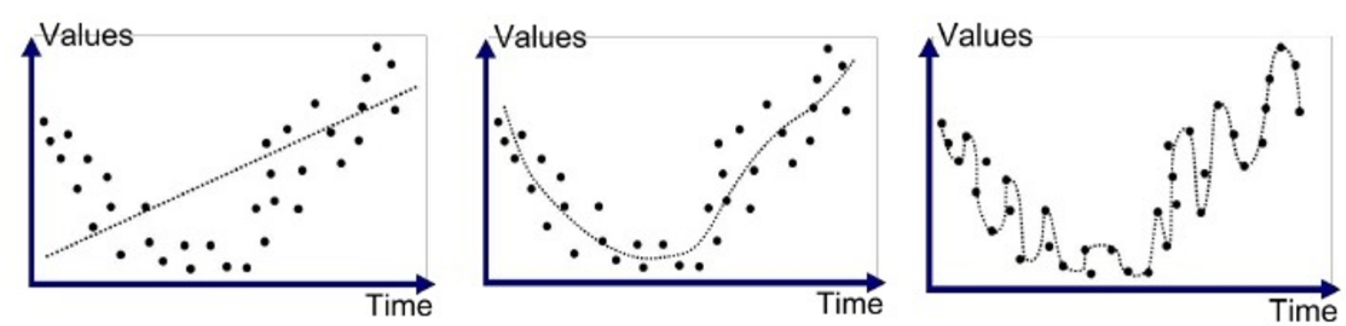
- **Name 2 advantages of kNN vs decision tree and vice versa.**

kNN: can incorporate interesting distance measures; few hyperparameters

decision trees: fast at test time; more interpretable; better deals with missing values (pass through both branches)

- **What is overfitting?**

Answer: it's a case when the model gets good performance on a particular dataset by "memorizing" it, but fails to generalize to new data.



Underfitted   Good Fit/Robust   Overfitted

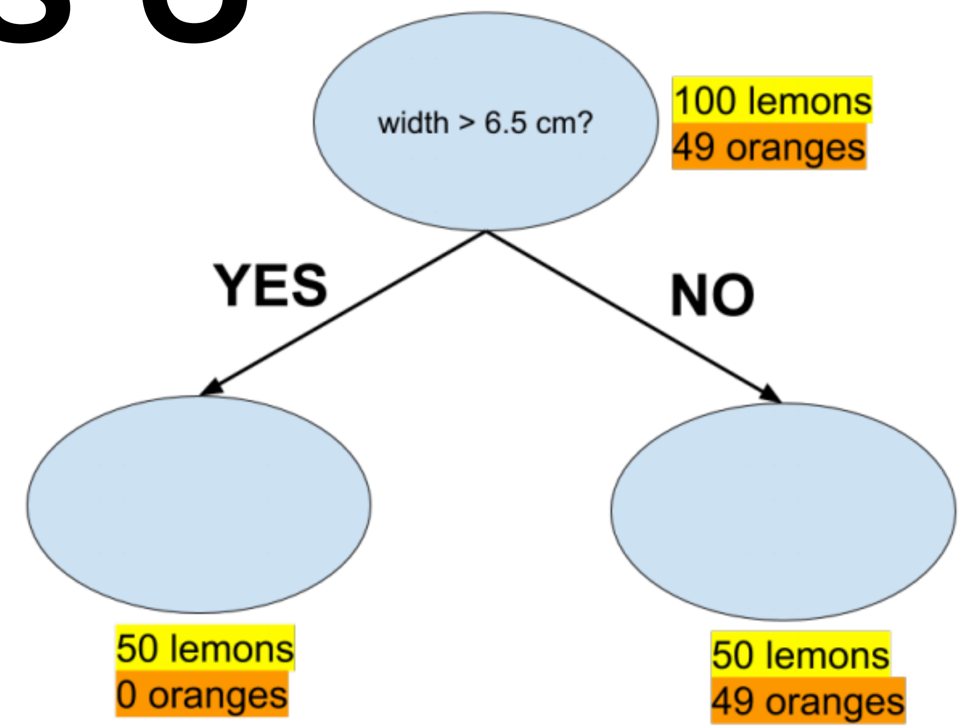
- **Why do we need a validation set?**

Answer: to prevent overfitting.

ML concepts 3

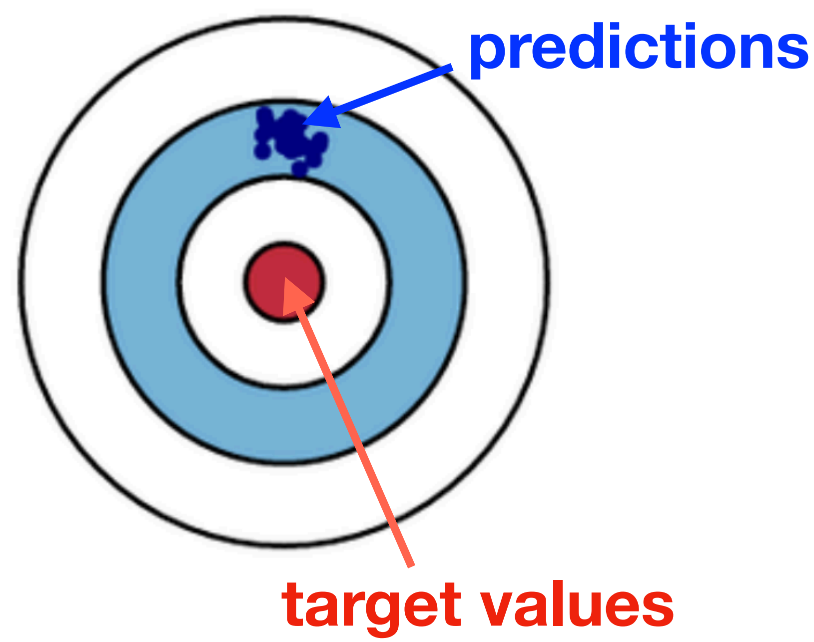
- **Based on which measure we can choose a good decision tree split?**

<u>Answer</u>: ***entropy*** (measures how chaotic / disorganized is our label distribution in a split).



- **What does this picture tells us about our data (bias / variance)?**

<u>Answer</u>: high bias & low variance.



- Are decision trees and kNN supervised / unsupervised algorithms?

<u>Answer</u>: supervised (we need labels).

ML concepts 4

- **Write a model for binary linear classification …**

$$z = \mathbf{w}^T\mathbf{x} + b$$

$$y = \begin{cases} 1 & \text{if } z \geq r \\ 0 & \text{if } z < r \end{cases}$$

- **What are the two ways of finding good values for model's parameters (w, b)?**

  A. direct solution

  B. iterative solution - gradient descent

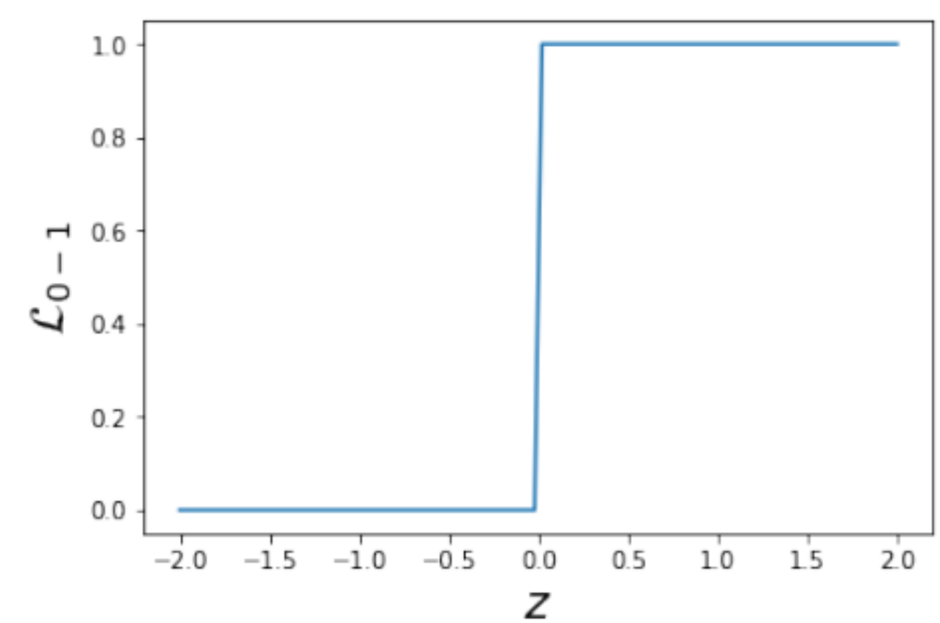
- **What is loss function?**

Answer: it's a function that evaluates how well specific algorithm models the given data; loss function takes predicted values and target values as inputs.

- **Write 0-1 loss function. Why is it bad?**

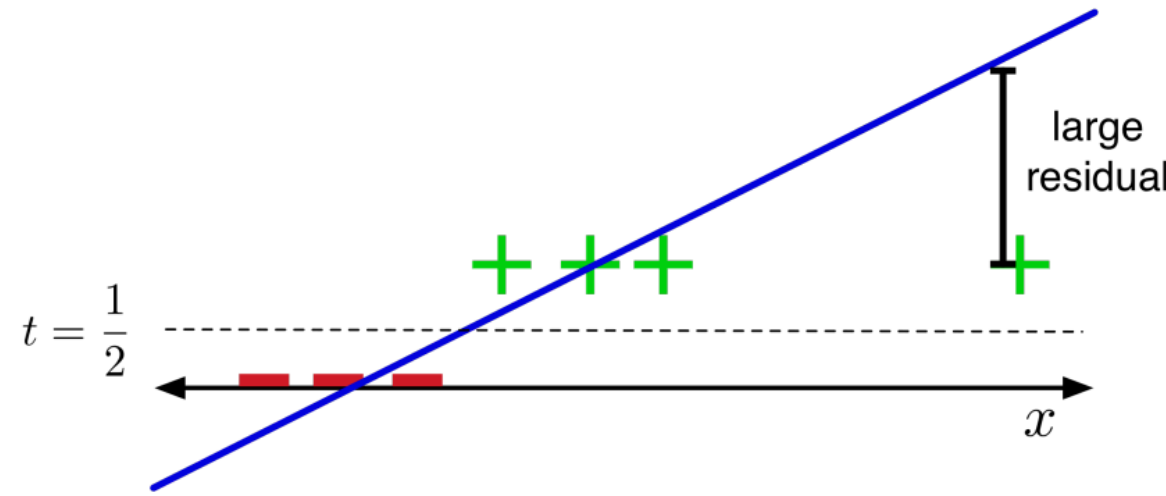
Answer: 0-1 loss is bad because it's not informative - its derivative is 0 everywhere it's defined.

$$\mathcal{L}_{0-1}(y,t) = \begin{cases} 0 & \text{if } y = t \\ 1 & \text{if } y \neq t \end{cases}$$

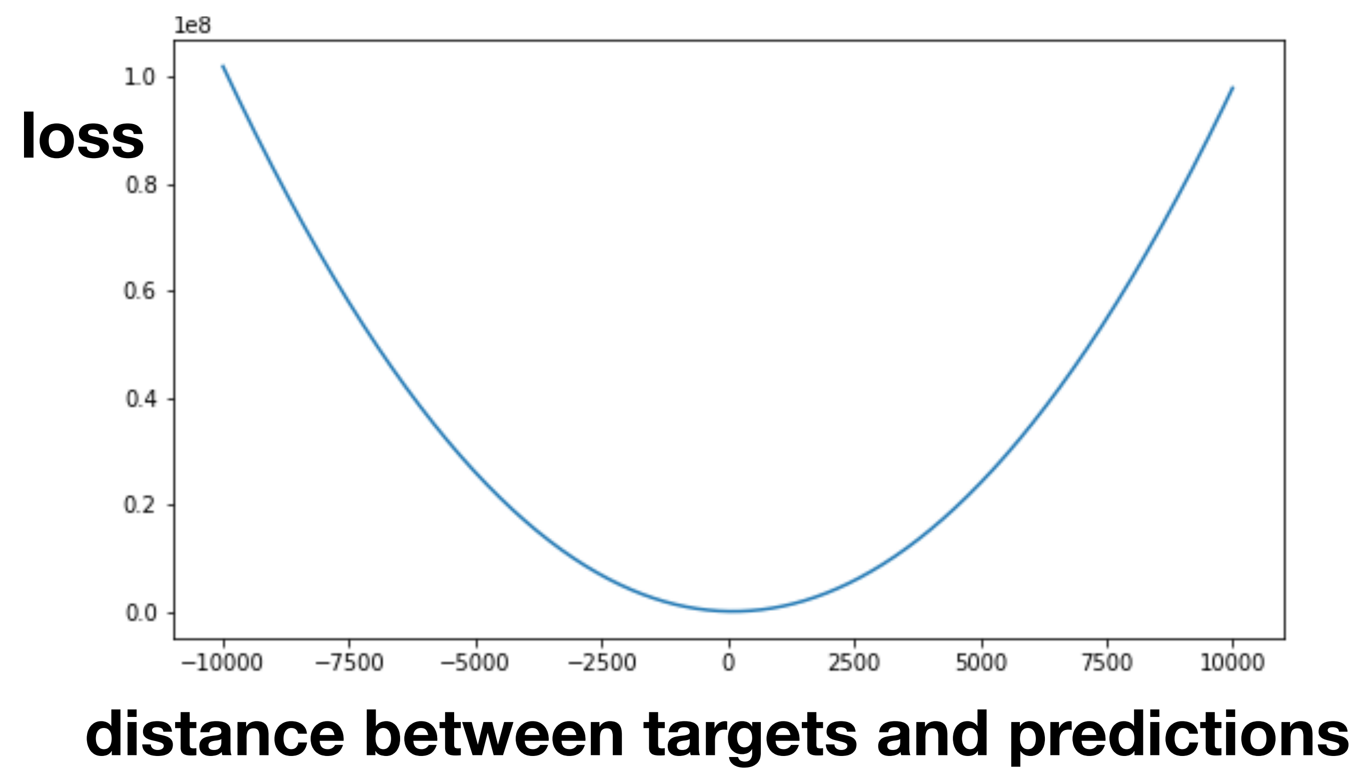


ML concepts 5

- **What are the problems with squared error loss function? How to solve it?**



Answer: squared error loss gives a big penalty for correct predictions that are made with high confidence.



Solution is to predict values only in [0,1] interval. For that we use **sigmoid function** to squash y into [0,1]:

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

$$z = \mathbf{w}^\top \mathbf{x} + b$$
$$y = \sigma(z)$$
$$\mathcal{L}_{\mathrm{SE}}(y, t) = \frac{1}{2}(y - t)^2.$$

ML concepts 6

- **What is the difference between parameters / hyper parameters of the model?**

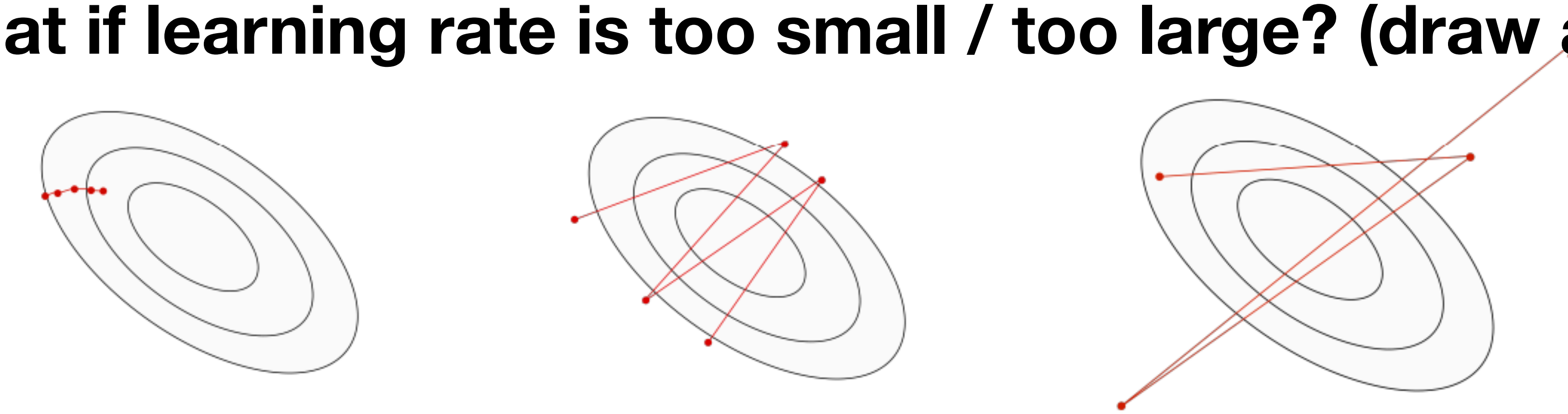
Answer: ***parameters*** are learned through training (by iteratively performing gradient descent updates) - weights and biases, ***hyperparameters*** are "manually" adjusted and set before training - number of hidden layers of a neural network, k for kNN, learning rate etc.

- **What is learning rate?**

Answer: learning rate is a hyper parameter that controls how much the weights are updated at each iteration.

$$w_j \leftarrow w_j - \alpha \frac{\partial \mathcal{J}}{\partial w_j}$$

- **What if learning rate is too small / too large? (draw a picture)**



α too small: slow progress

α too large: oscillations

α much too large: instability

ML concepts 7

- **What is regularization? Why do we need it?**

Answer: regularization is a technique of adding an extra term to the loss function. It reduces overfitting by keeping the weights of the model smaller.

- **What is softmax? Calculate** $softmax(\begin{bmatrix} 2 \\ 1 \\ 0.1 \end{bmatrix})$

Answer: softmax is an ***activation function*** for multi-class classification that maps input ***logits*** to probabilities.

$$softmax(\begin{bmatrix} 2 \\ 1 \\ 0.1 \end{bmatrix}) = \begin{bmatrix} e^2/(e^2 + e^1 + e^{0.1}) \\ e^1/(e^2 + e^1 + e^{0.1}) \\ e^{0.1}/(e^2 + e^1 + e^{0.1}) \end{bmatrix} = \begin{bmatrix} 0.7 \\ 0.2 \\ 0.1 \end{bmatrix}$$

$$y_k = \text{softmax}(z_1, \ldots, z_K)_k = \frac{e^{z_k}}{\sum_{k'} e^{z_{k'}}}$$

Example 1

Find a linear classifier with weights w1, w2, w3, and b which correctly classifies all of these training examples:

x_1	x_2	x_3	t
0	0	0	1
0	1	0	0
0	1	1	1
1	1	1	0

$$w_1x_1 + w_2x_2 + w_3x_3 + b \gtrless 0$$

Answer: write a system of inequalities and find one solution (there would be many possible answers).

$$\begin{aligned} b &> 0 \\ w_2 + b &< 0 \\ w_2 + w_3 + b &> 0 \\ w_1 + w_2 + w_3 + b &< 0 \end{aligned} \qquad \begin{aligned} b &= 1 \\ w_1 &= -2 \\ w_2 &= -2 \\ w_3 &= 2 \end{aligned}$$