CMSC 422 Introduction to Machine Learning

Lecture 24 Support Vector Machines II and Final Summary

Furong Huang / furongh@umd.edu

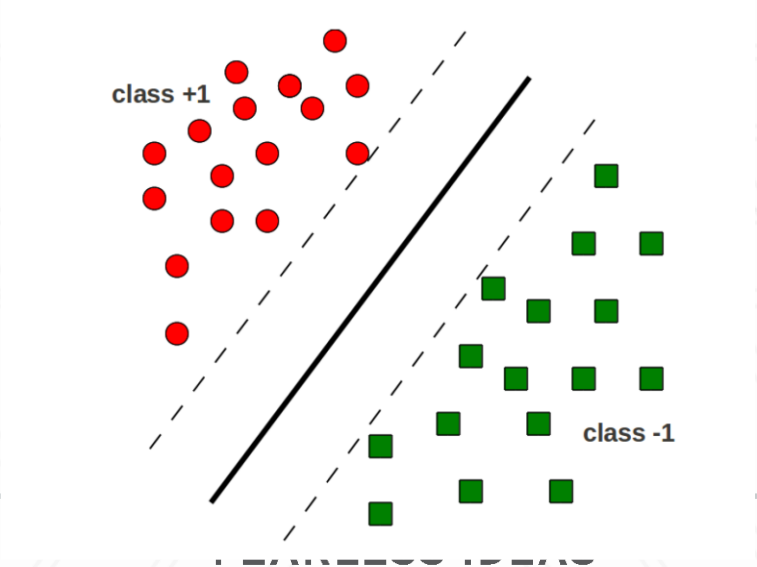
UNIVERSITY OF MARYLAND

What we know about SVM so far

REVIEW

The Maximum Margin Principle

Find the hyperplane with maximum separation margin on the training data



Support Vector Machine (SVM)

A hyperplane based linear classifier defined by $\mathbf{w}$ and b

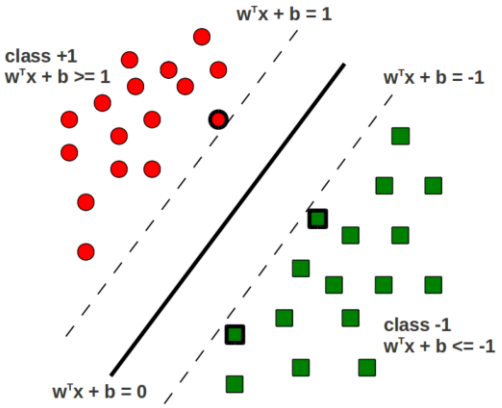
Prediction rule: $y = sign(\mathbf{w}^T\mathbf{x} + b)$

Given: Training data $\{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)\}$

Goal: Learn $\mathbf{w}$ and b that achieve the maximum margin

Characterizing the margin

Let's assume the entire training data is correctly classified by (**w**,b) that achieve the maximum margin



- Assume the hyperplane is such that
  - $\mathbf{w}^T\mathbf{x}_n + b \geq 1$ for $y_n = +1$
  - $\mathbf{w}^T\mathbf{x}_n + b \leq -1$ for $y_n = -1$
  - Equivalently, $y_n(\mathbf{w}^T\mathbf{x}_n + b) \geq 1$
    $\Rightarrow \min_{1\leq n\leq N} |\mathbf{w}^T\mathbf{x}_n + b| = 1$
  - The hyperplane's margin:

$$\gamma = \min_{1\leq n\leq N} \frac{|\mathbf{w}^T\mathbf{x}_n+b|}{||\mathbf{w}||} = \frac{1}{||\mathbf{w}||}$$

Solving the SVM Optimization Problem (assuming linearly separable data)

Our optimization problem is:

$$\begin{aligned} &\text{Minimize} \quad f(\mathbf{w}, b) = \frac{||\mathbf{w}||^2}{2} \\ &\text{subject to} \quad 1 \leq y_n(\mathbf{w}^T\mathbf{x}_n + b), \qquad n = 1, \ldots, N \end{aligned}$$

Introducing Lagrange Multipliers α_n ($n = \{1, \ldots, N\}$), one for each constraint, leads to the **Lagrangian**:

$$\begin{aligned} &\text{Minimize} \quad L(\mathbf{w}, b, \alpha) = \frac{||\mathbf{w}||^2}{2} + \sum_{n=1}^{N} \alpha_n\{1 - y_n(\mathbf{w}^T\mathbf{x}_n + b)\} \\ &\text{subject to} \quad \alpha_n \geq 0; \quad n = 1, \ldots, N \end{aligned}$$

Solving the SVM Optimization Problem (assuming linearly separable data)

Take (partial) derivatives of L_P w.r.t. $\mathbf{w}$, b and set them to zero

$$\frac{\partial L_P}{\partial \mathbf{w}} = 0 \Rightarrow \mathbf{w} = \sum_{n=1}^{N} \alpha_n y_n \mathbf{x}_n, \qquad \frac{\partial L_P}{\partial b} = 0 \Rightarrow \sum_{n=1}^{N} \alpha_n y_n = 0$$

Substituting these in the Primal Lagrangian L_P gives the Dual Lagrangian

$$\text{Maximize} \quad L_D(\mathbf{w}, b, \alpha) = \sum_{n=1}^{N} \alpha_n - \frac{1}{2} \sum_{m,n=1}^{N} \alpha_m \alpha_n y_m y_n (\mathbf{x}_m^T \mathbf{x}_n)$$

$$\text{subject to} \quad \sum_{n=1}^{N} \alpha_n y_n = 0, \quad \alpha_n \geq 0; \quad n = 1, \ldots, N$$

Solving the SVM Optimization Problem (assuming linearly separable data)

Take (partial) derivatives of L_P w.r.t. $\mathbf{w}$, b and set them to zero

$$= \sum_{n=1}^{N} \alpha_n y_n \mathbf{x}_n, \qquad \frac{\partial L_P}{\partial b} = 0 \Rightarrow \sum_{n=1}^{N} \alpha_n y_n = 0$$

A Quadratic Program for which many off-the-shelf solvers exist

Substituting thes the Primal Lagrangian L_P gives the Dual Lagrangian

$$\text{Maximize} \quad L_D(\mathbf{w}, b, \alpha) = \sum_{n=1}^{N} \alpha_n - \frac{1}{2} \sum_{m,n=1}^{N} \alpha_m \alpha_n y_m y_n (\mathbf{x}_m^T \mathbf{x}_n)$$

$$\text{subject to} \quad \sum_{n=1}^{N} \alpha_n y_n = 0, \quad \alpha_n \geq 0; \quad n = 1, \ldots, N$$

SVM: the solution!
(assuming linearly separable data)

Once we have the α_n's, $\mathbf{w}$ and b can be computed as:

$$\mathbf{w} = \sum_{n=1}^{N} \alpha_n y_n \mathbf{x}_n$$

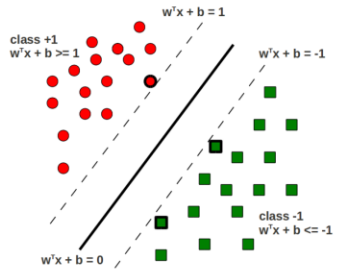
$$b = -\frac{1}{2}\left(\min_{n:y_n=+1} \mathbf{w}^T \mathbf{x}_n + \max_{n:y_n=-1} \mathbf{w}^T \mathbf{x}_n\right)$$

Note: Most α_n's in the solution are zero (sparse solution)

- Reason: Karush-Kuhn-Tucker (KKT) conditions
- For the optimal α_n's

$$\alpha_n\{1 - y_n(\mathbf{w}^T \mathbf{x}_n + b)\} = 0$$

- α_n is non-zero only if $\mathbf{x}_n$ lies on one of the two margin boundaries, i.e., for which $y_n(\mathbf{w}^T \mathbf{x}_n + b) = 1$
- These examples are called support vectors
- Support vectors "support" the margin boundaries



What if the data is not separable?

GENERAL CASE SVM SOLUTION

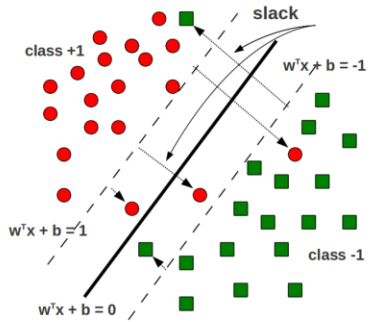
SVM in the non-separable case

no hyperplane can separate the classes perfectly

We still want to find the max margin hyperplane, but

- We will allow some training examples to be **misclassified**
- We will allow some training examples to fall **within** the margin region

SVM in the non-separable case



Recall: For the separable case (training loss $= 0$), the constraints were:

$$y_n(\mathbf{w}^T\mathbf{x}_n + b) \geq 1 \quad \forall n$$

For the non-separable case, we relax the above constraints as:

$$\boxed{y_n(\mathbf{w}^T\mathbf{x}_n + b) \geq 1 - \xi_n \quad \forall n}$$

ξ_n is called slack variable (distance $\mathbf{x}_n$ goes past the margin boundary)

$\xi_n \geq 0, \forall n$, misclassification when $\xi_n > 1$

SVM Optimization Problem

Non-separable case: We will allow misclassified training examples

- .. but we want their number to be minimized
  $\Rightarrow$ by *minimizing* the sum of slack variables $(\sum_{n=1}^{N} \xi_n)$

The optimization problem for the non-separable case

$$\text{Minimize} \quad f(\mathbf{w}, b) = \frac{||\mathbf{w}||^2}{2} + C \sum_{n=1}^{N} \xi_n$$

$$\text{subject to} \quad y_n(\mathbf{w}^T \mathbf{x}_n + b) \geq 1 - \xi_n, \quad \xi_n \geq 0 \qquad n = 1, \ldots, N$$

C hyperparameter dictates which term dominates the minimization

- Small C => prefer large margins and allows more misclassified examples
- Large C => prefer small number of misclassified examples, but at the expense of a small margin

Introducing Lagrange Multipliers…

Our optimization problem is:

$$\text{Minimize} \quad f(\mathbf{w}, b, \xi) = \frac{||\mathbf{w}||^2}{2} + C\sum_{n=1}^{N}\xi_n$$

$$\text{subject to} \quad 1 \leq y_n(\mathbf{w}^T\mathbf{x}_n + b) + \xi_n, \quad 0 \leq \xi_n \qquad n = 1, \ldots, N$$

Introducing Lagrange Multipliers α_n, β_n $(n = \{1, \ldots, N\})$, for the constraints, leads to the Primal Lagrangian:

$$\text{Minimize} \quad L_P(\mathbf{w}, b, \xi, \alpha, \beta) = \frac{||\mathbf{w}||^2}{2} + +C\sum_{n=1}^{N}\xi_n + \sum_{n=1}^{N}\alpha_n\{1 - y_n(\mathbf{w}^T\mathbf{x}_n + b) - \xi_n\} - \sum_{n=1}^{N}\beta_n\xi_n$$

$$\text{subject to} \quad \alpha_n, \beta_n \geq 0; \quad n = 1, \ldots, N$$

Terms in red are those that were not there in the separable case!

Formulating the dual objective

Take (partial) derivatives of L_P w.r.t. $\mathbf{w}$, b, ξ_n and set them to zero

$$\frac{\partial L_P}{\partial \mathbf{w}} = 0 \Rightarrow \mathbf{w} = \sum_{n=1}^{N} \alpha_n y_n \mathbf{x}_n, \quad \frac{\partial L_P}{\partial b} = 0 \Rightarrow \sum_{n=1}^{N} \alpha_n y_n = 0, \quad \frac{\partial L_P}{\partial \xi_n} = 0 \Rightarrow C - \alpha_n - \beta_n = 0$$

Using $C - \alpha_n - \beta_n = 0$ and $\beta_n \geq 0 \Rightarrow \alpha_n \leq C$

Substituting these in the Primal Lagrangian L_P gives the Dual Lagrangian

$$\text{Maximize} \quad L_D(\mathbf{w}, b, \xi, \alpha, \beta) = \sum_{n=1}^{N} \alpha_n - \frac{1}{2} \sum_{m,n=1}^{N} \alpha_m \alpha_n y_m y_n (\mathbf{x}_m^T \mathbf{x}_n)$$

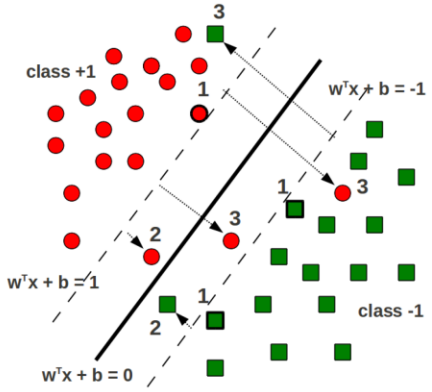
$$\text{subject to} \quad \sum_{n=1}^{N} \alpha_n y_n = 0, \quad 0 \leq \alpha_n \leq C; \quad n = 1, \ldots, N$$

Note

- Given α the solution for w, b has the same form as in the separable case
- α is again sparse, nonzero α_n's correspond to support vectors

Support Vectors in the Non-Separable Case

We now have 3 types of support vectors!



(1) Lying on the margin boundaries $\mathbf{w}^T\mathbf{x} + b = -1$ and $\mathbf{w}^T\mathbf{x} + b = +1$ ($\xi_n = 0$)

(2) Lying within the margin region ($0 < \xi_n < 1$) but still on the correct side

(3) Lying on the wrong side of the hyperplane ($\xi_n \geq 1$)

Notes on training

Solving the quadratic problem is O(N^3)

- Can be prohibitive for large datasets

But many options to speed up training

- Approximate solvers
- Learn from what we know about training linear models

What you should know

What are Support Vector Machines

How to train SVMs

Which optimization problem we need to solve

Geometric interpretation

- What are support vectors and what is their relationship with parameters **w**,b?

How do SVM relate to the general formulation of linear classifiers

Why/how can SVMs be kernelized

End of semester logistics

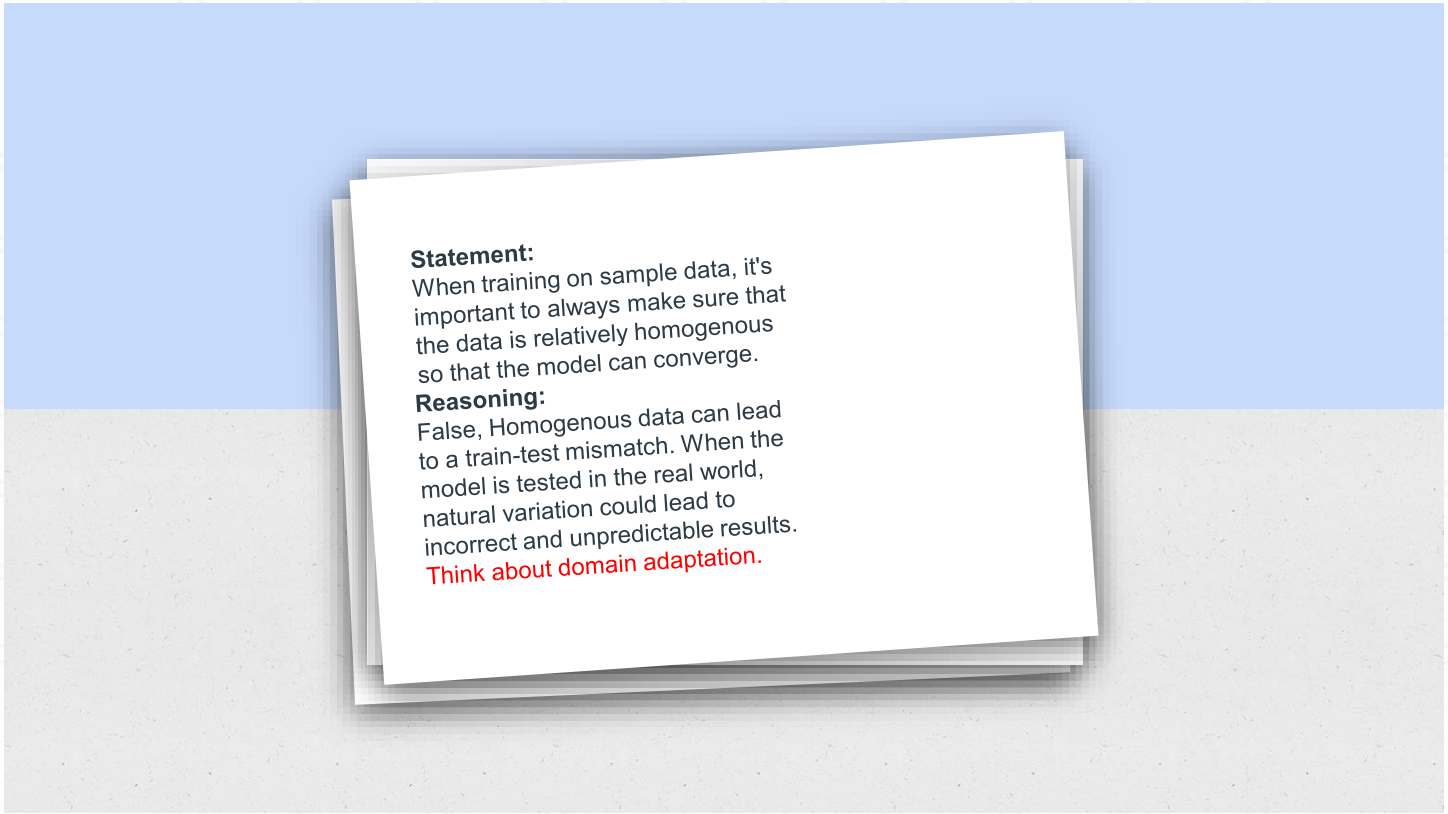
Course evals

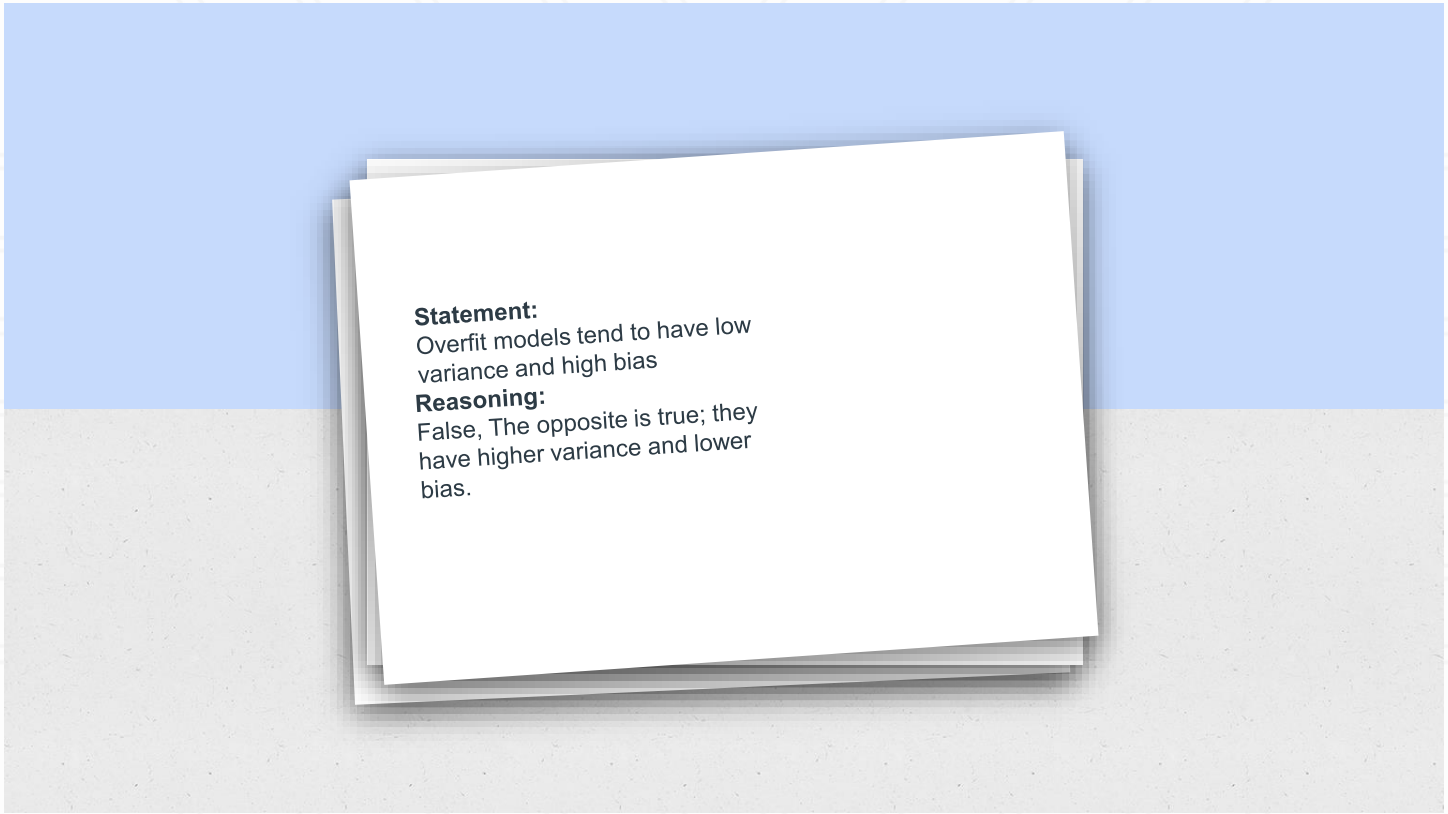
https://www.courseevalum.umd.edu/

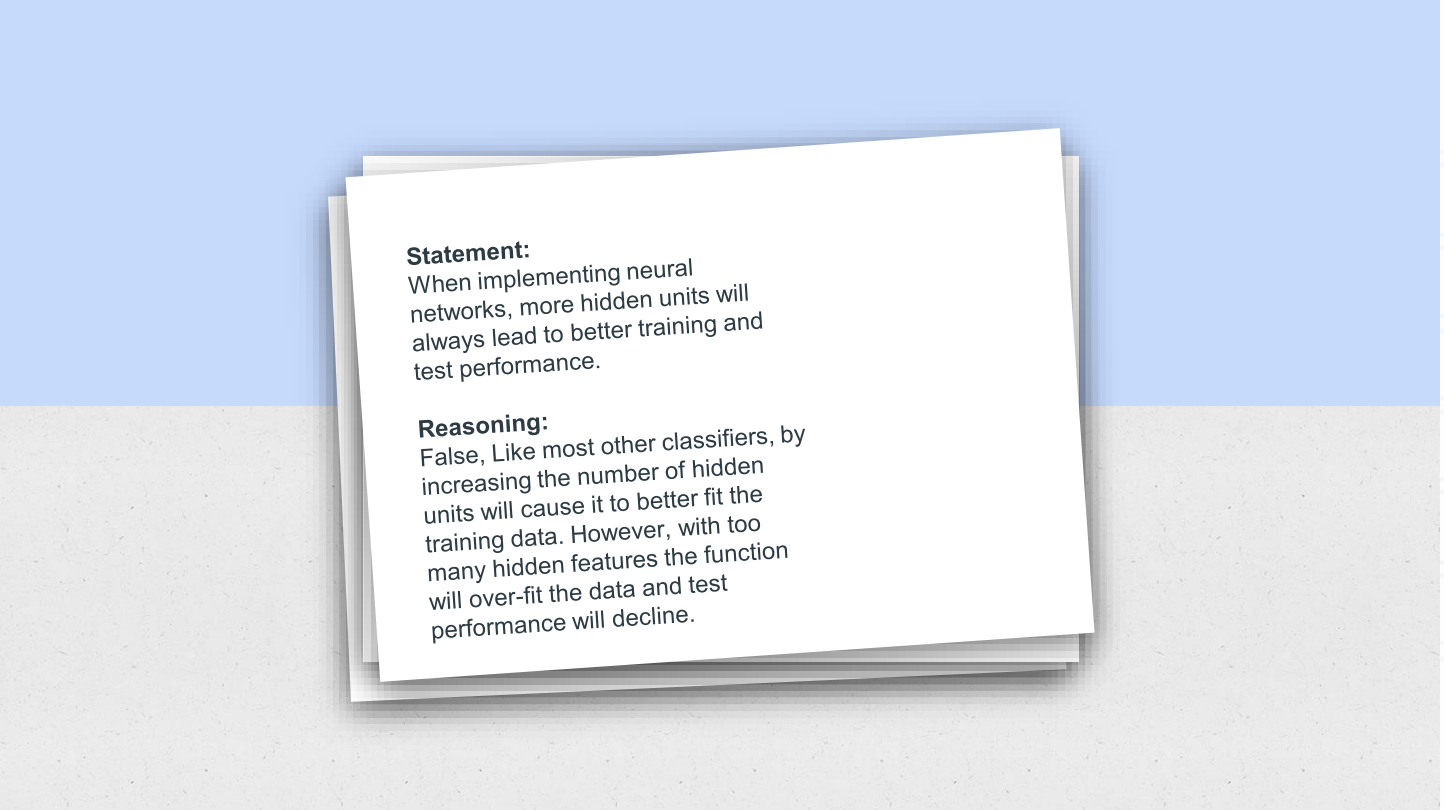
What you should know
Bias and how to deal with it

- **What is the impact of data selection bias in machine learning systems?**
- **How to address train/test mismatch**
  - Unsupervised adaptation
    - Using auxiliary classifier
  - Supervised adaptation
    - Feature augmentation
- Overfitting/Underfitting

Incorrect reasonings are highlighted
Corrections are marked red


Statement:
When training on sample data, it's important to always make sure that the data is relatively homogenous so that the model can converge.
Reasoning:
False, Homogenous data can lead to a train-test mismatch. When the model is tested in the real world, natural variation could lead to incorrect and unpredictable results.
Think about domain adaptation.


Statement:
Overfit models tend to have low variance and high bias
Reasoning:
False, The opposite is true; they have higher variance and lower bias.

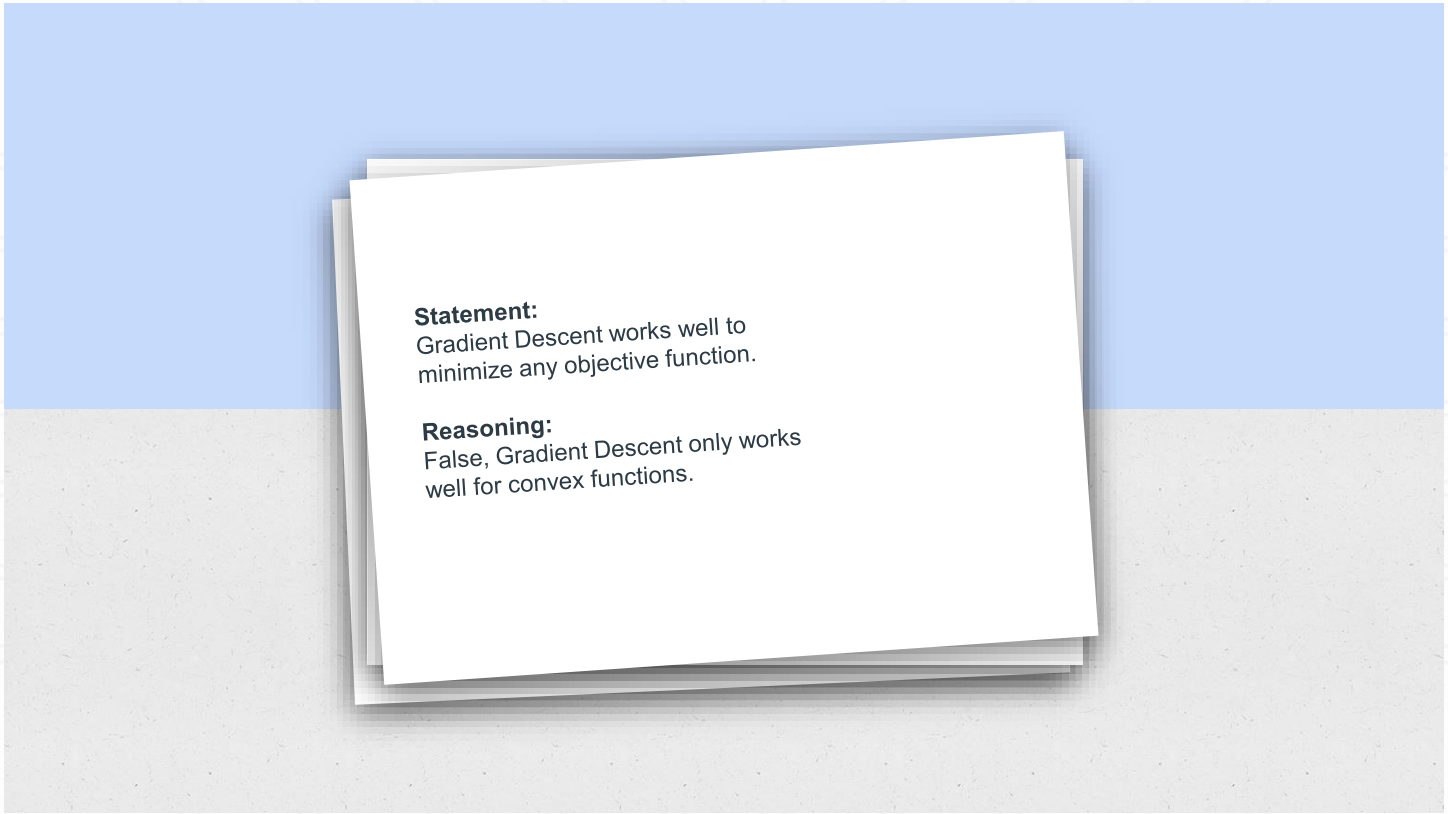

Statement:
When implementing neural networks, more hidden units will always lead to better training and test performance.
Reasoning:
False, Like most other classifiers, by increasing the number of hidden units will cause it to better fit the training data. However, with too many hidden features the function will over-fit the data and test performance will decline.

What you should know
Linear Models

- What are linear models?
  - A general framework for binary classification
  - How optimization objectives are defined
    - Loss function and regularizes
  - Separate model definition from training algorithm (Gradient Descent)

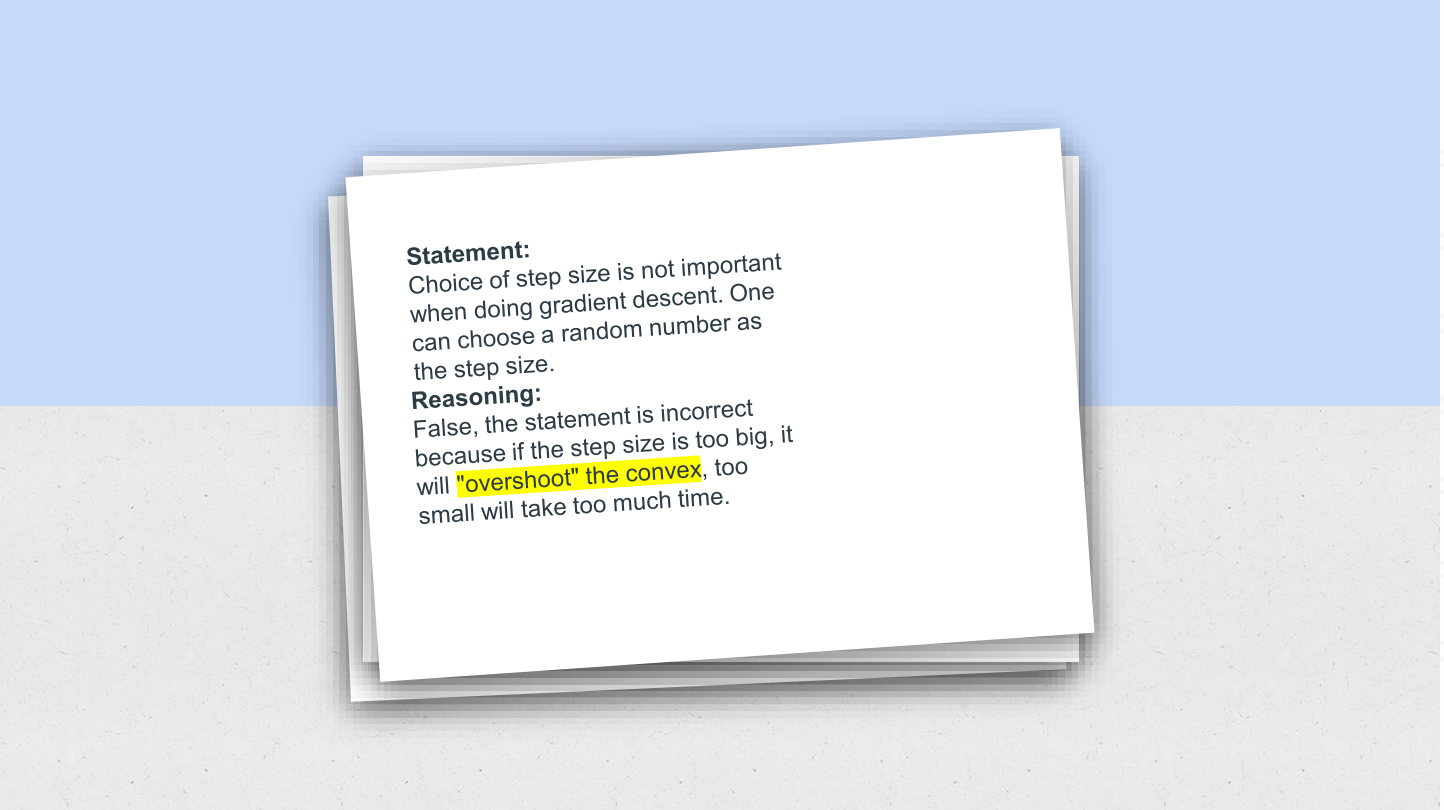
What you should know
Gradient Descent

- Gradient descent
  - A generic algorithm to minimize objective functions
  - What are the properties of the objectives for which it works well?
  - Subgradient descent (i.e. what to do at points where derivative is not defined)
  - Why choice of step size, initialization matters


Statement:
Gradient Descent works well to minimize any objective function.
Reasoning:
False, Gradient Descent only works well for convex functions.

Statement:
Gradient descent points you in the direction that maximizes your loss.

Reasoning:
True, this is why you need to do $-\nabla_w f(w)$, because it helps minimize loss
The gradient of the loss function with respect to the parameters w points in the direction of maximum increase for the loss

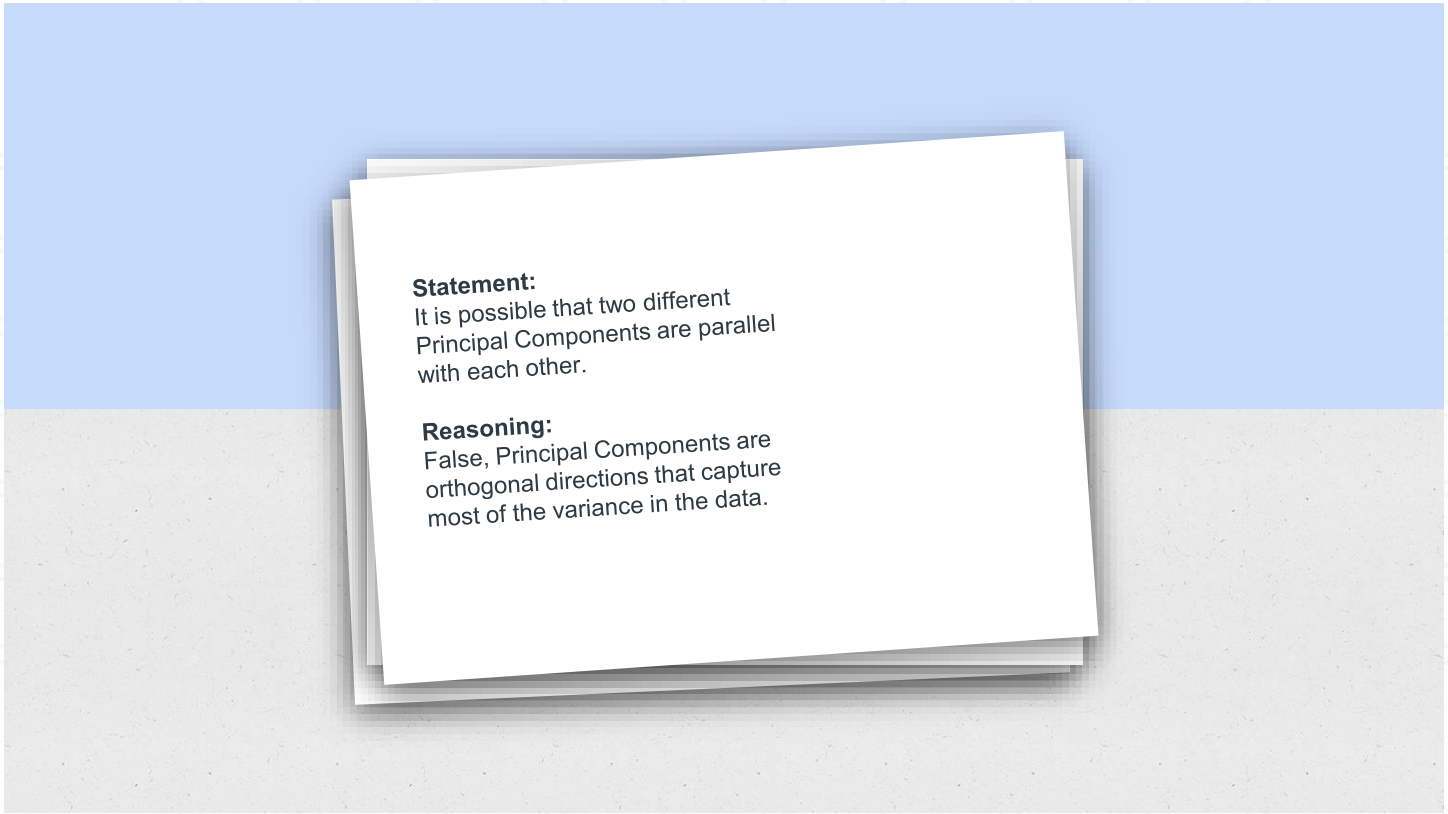

Statement:
Choice of step size is not important when doing gradient descent. One can choose a random number as the step size.
Reasoning:
False, the statement is incorrect because if the step size is too big, it will "overshoot" the convex, too small will take too much time.

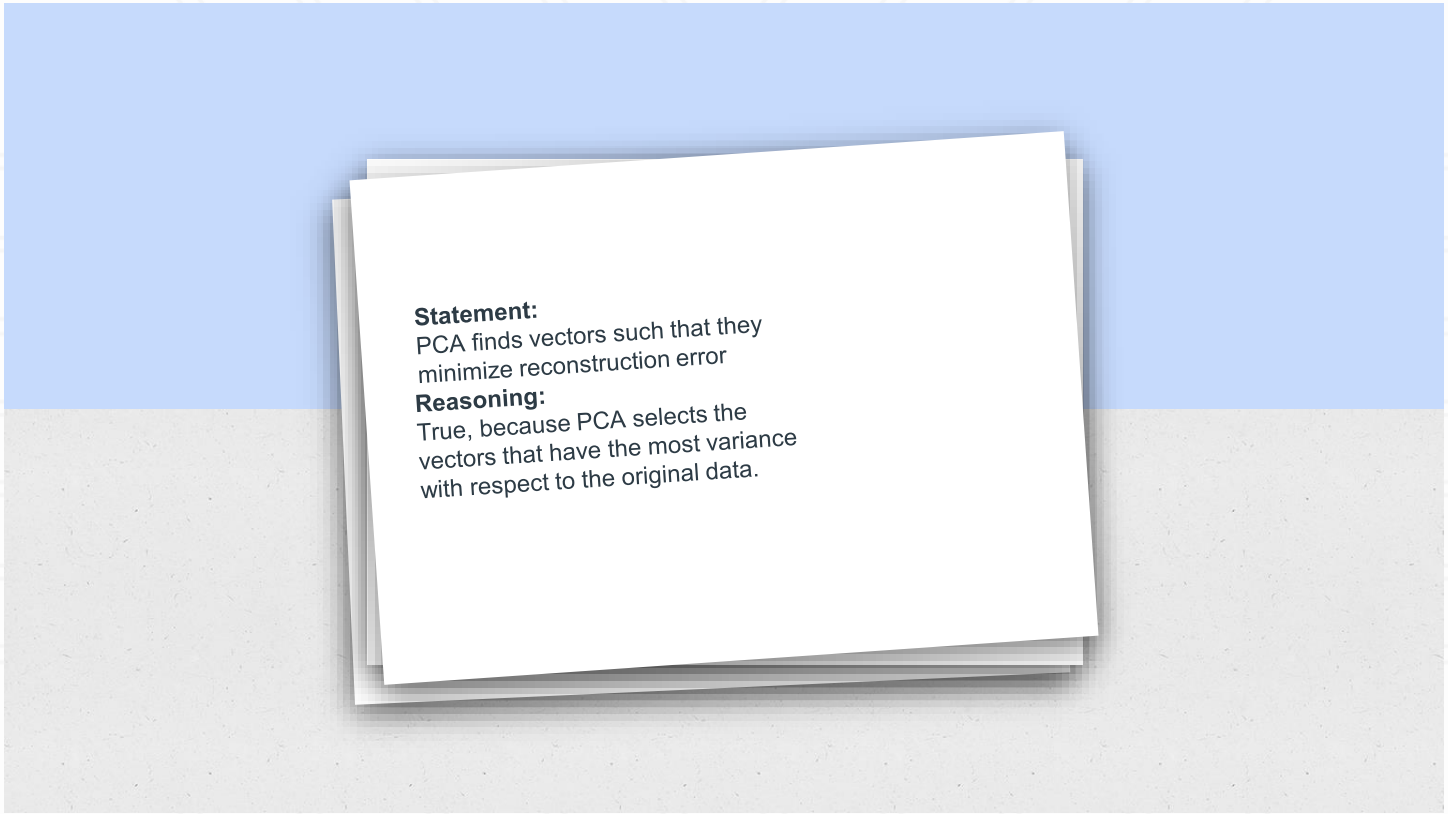
What you should know
Probabilistic Models

- The Naïve Bayes classifier
  - Conditional independence assumption
  - How to train it?
  - How to make predictions?
  - How does it relate to other classifiers we know?
- Fundamental Machine Learning concepts
  - Iid assumption
  - Bayes optimal classifer
  - Maximum likelihood estimation
  - Generative story

What you should know
Principal Component Analysis

- Goal: Find a projection of the data onto directions that maximize variance of the original data set
- PCA **optimization objective** and resulting **algorithm**
- Why this is useful


Statement:
It is possible that two different Principal Components are parallel with each other.
Reasoning:
False, Principal Components are orthogonal directions that capture most of the variance in the data.

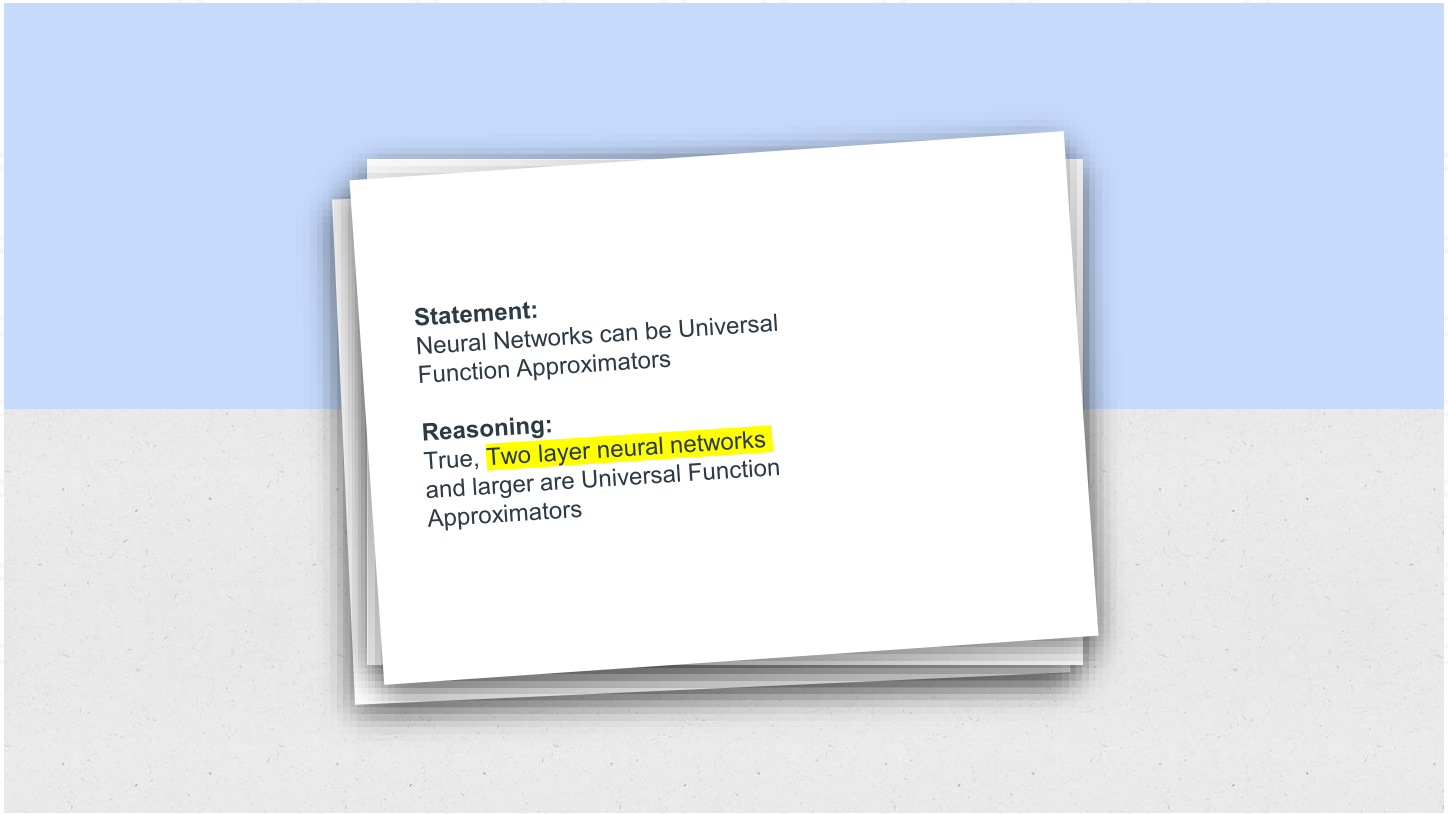

Statement:
PCA finds vectors such that they minimize reconstruction error
Reasoning:
True, because PCA selects the vectors that have the most variance with respect to the original data.

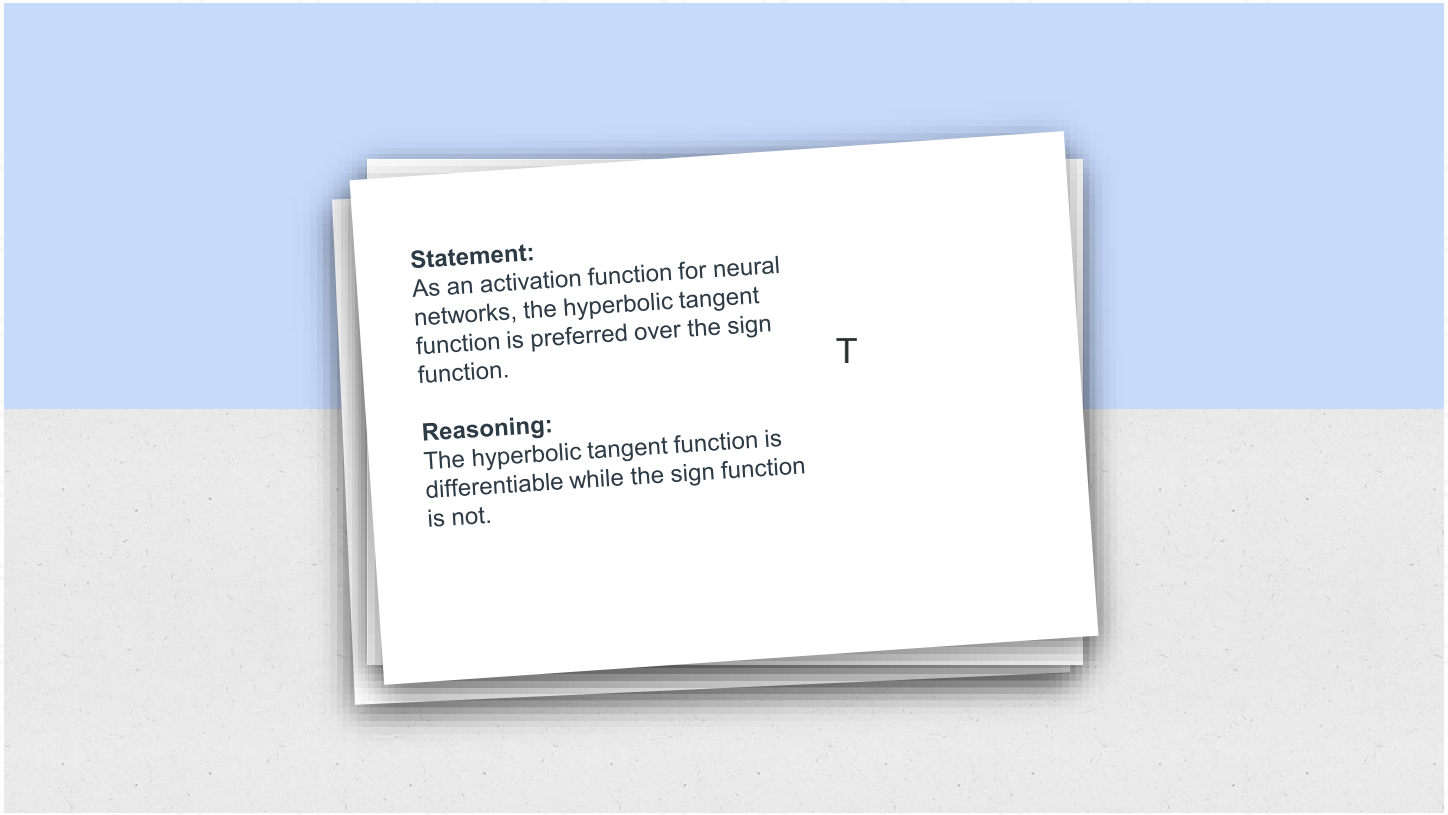
What you should know
Neural Networks

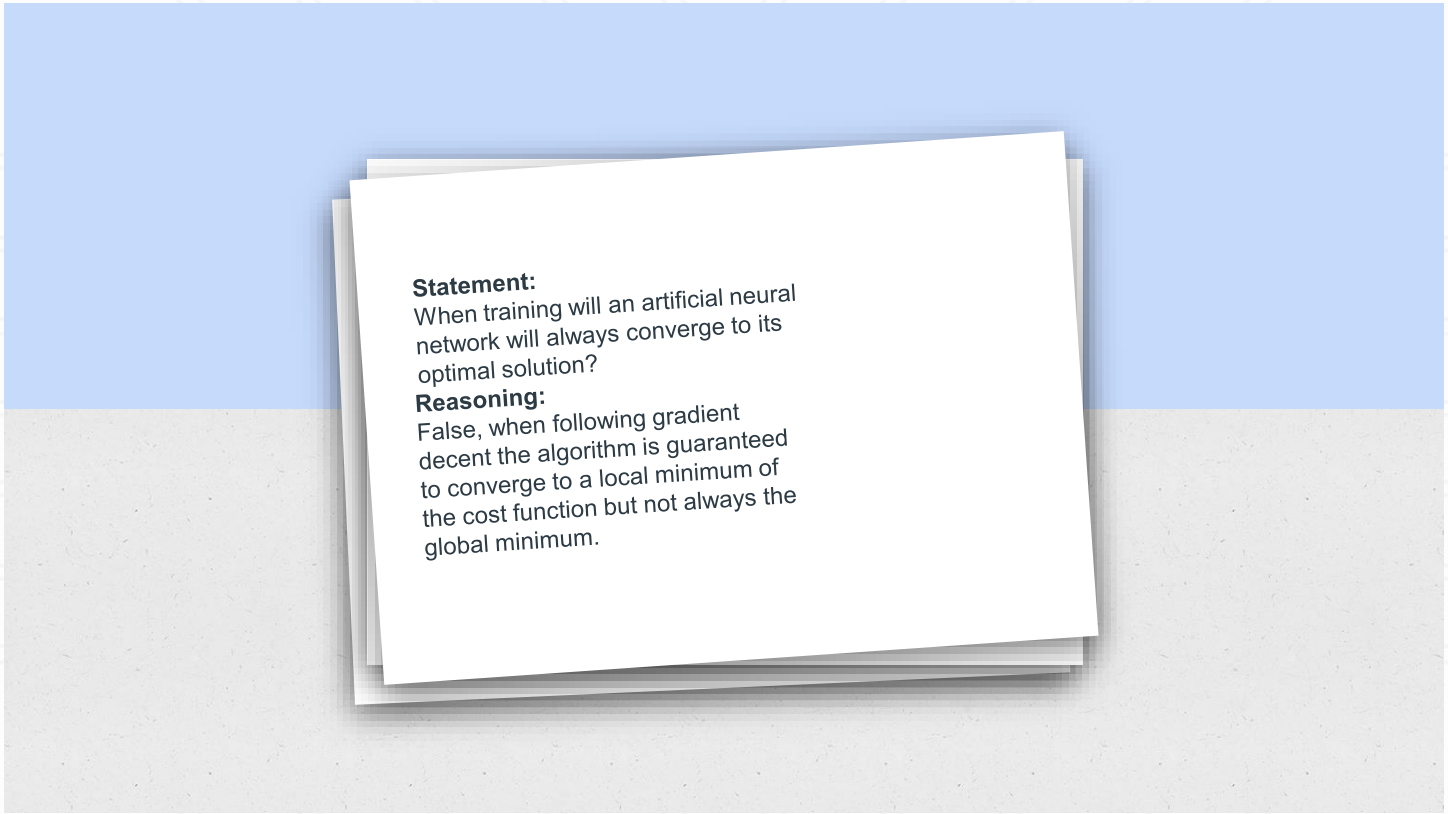
– What are Neural Networks?
    - Multilayer perceptron
– How to make a prediction given an input?
    - Forward propagation: Matrix operations + non-linearities
– Why are neural networks powerful?
- Universal function approximators!
– How to train neural networks?
- The backpropagation algorithm
    – How to step through it, and how to derive update rules

What you should know
Deep Learning

- Why training deep networks is challenging
  - Computationally expensive, vanishing gradient
- Practical techniques for training deep networks
  - Computational graph
  - Stochastic gradient descent
  - Momentum
  - Weight decay


Statement:
Neural Networks can be Universal Function Approximators
Reasoning:
True, Two layer neural networks and larger are Universal Function Approximators


Statement:
As an activation function for neural networks, the hyperbolic tangent function is preferred over the sign function.
T
Reasoning:
The hyperbolic tangent function is differentiable while the sign function is not.

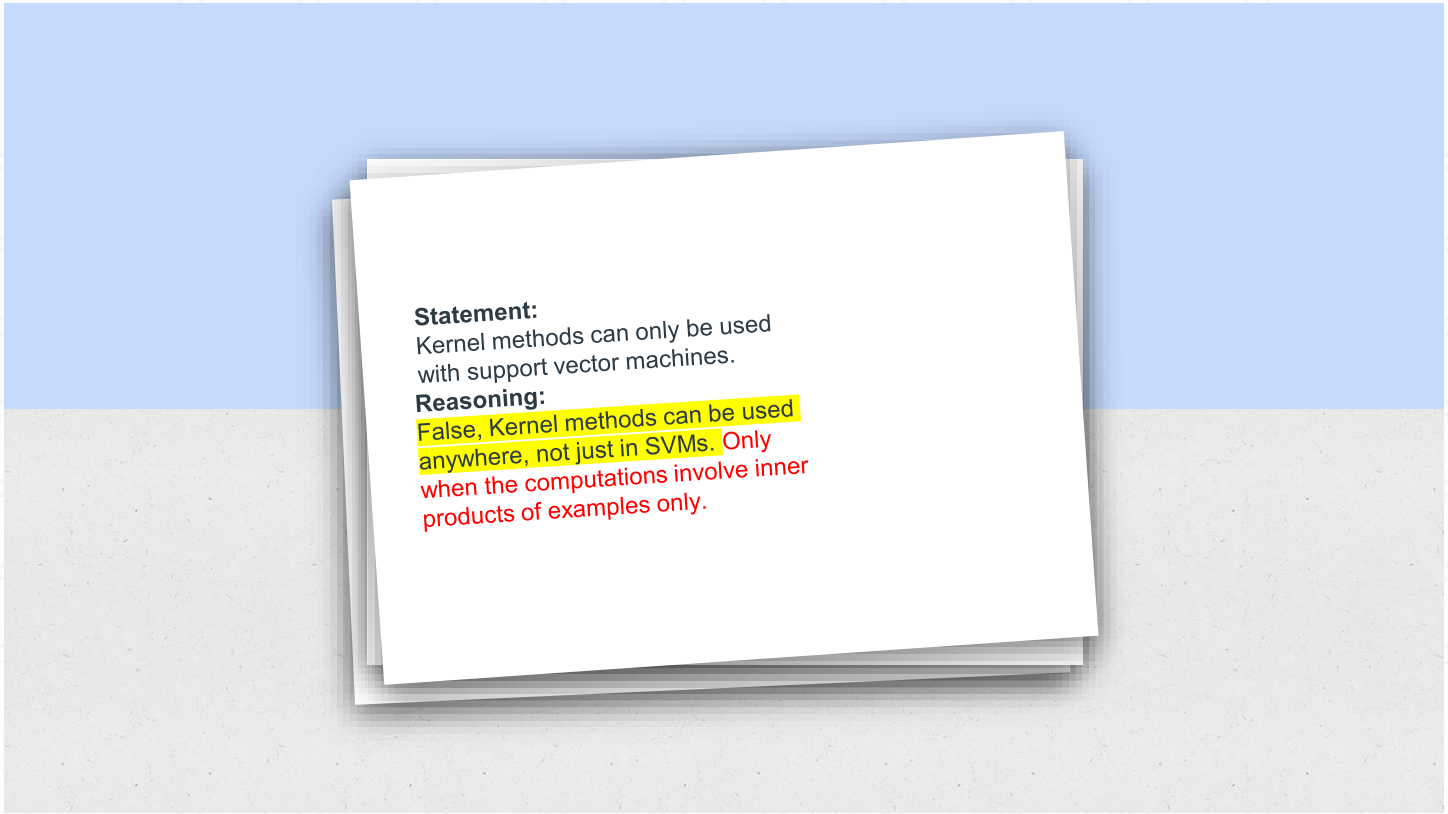

Statement:
When training will an artificial neural network will always converge to its optimal solution?
Reasoning:
False, when following gradient decent the algorithm is guaranteed to converge to a local minimum of the cost function but not always the global minimum.

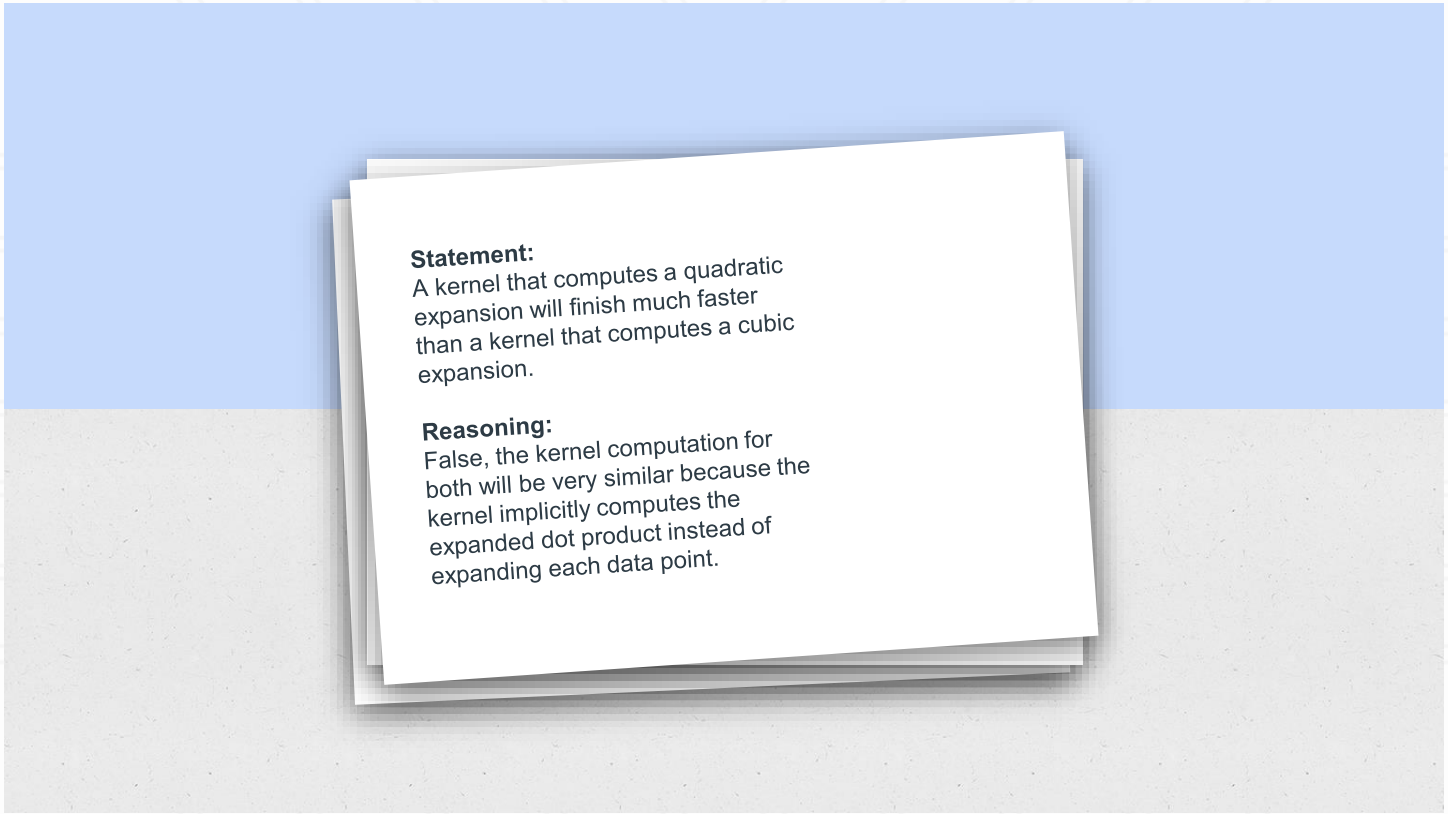
What you should know
Kernels

- Kernel functions
  - What they are, why they are useful, how they relate to feature combination
- Kernelized perceptron
  - You should be able to derive it and implement it

What you should know
SVMs

- What are Support Vector Machines
  - Hard margin vs. soft margin SVMs
- How to train SVMs
  - Which optimization problem we need to solve
- Geometric interpretation
  - What are support vectors and what is their relationship with parameters **w**,b?
- How do SVM relate to the general formulation of linear classifiers
- Why/how can SVMs be kernelized


Statement:
Kernel methods can only be used with support vector machines.
Reasoning:
False, Kernel methods can be used anywhere, not just in SVMs. Only when the computations involve inner products of examples only.


Statement:
A kernel that computes a quadratic expansion will finish much faster than a kernel that computes a cubic expansion.
Reasoning:
False, the kernel computation for both will be very similar because the kernel implicitly computes the expanded dot product instead of expanding each data point.


Statement:
Support vectors are those that lie precisely 1 unit away from the maximum margin decision boundary.
Reasoning:
True, these points are called the support vectors because they "support" the decision boundary. (definition in the ciml book) Not always 1 unit.

Statement:
If you remove one of the support vectors from the training set for a support vector machine, the size of the maximum margin decreases.

Reasoning:
False, The maximum size of the margin either stays the same or increases, as support vectors keep the margin from expanding.

Machine Learning

- Paradigm: "Programming by example"
  - Replace "human writing code" with "human supplying data"
- Most central issue: generalization
  - How to abstract from "training" examples to "test" examples?

Course Goal

- Now, by the end of the semester, you should be able to
  - Look at a problem and identify if ML is an appropriate solution
  - If so, identity what types of algorithms might be applicable
  - Apply those algorithms
- This course is not
  - A survey of ML algorithms
  - A tutorial on ML toolkits such as Pytorch, TensorFlow, …

Key ingredients needed for learning

- Training vs. test examples
  - Memorizing the training examples is not enough!
  - Need to generalize to make good predictions on test examples
- Inductive bias
  - Many classifier hypotheses are plausible
  - Need assumptions about the nature of the relation between examples and classes

Machine Learning as Function Approximation

Problem setting

- Set of possible instances X
- Unknown target function $f: X \rightarrow Y$
- Set of function hypotheses $H = \{h \mid h: X \rightarrow Y\}$

Input

- Training examples $\{(x^{(1)}, y^{(1)}), \ldots (x^{(N)}, y^{(N)})\}$ of unknown target function f

Output

- Hypothesis $h \in H$ that best approximates target function f

Formalizing Induction

- Given
  - a loss function l
  - a sample from some unknown data distribution D
- Our task is to compute a function f that has low expected error over D with respect to l.

$$\mathbb{E}_{(x,y)\sim D}\{l(y, f(x))\} = \sum_{(x,y)} D(x,y)\, l(y, f(x))$$

Beyond CMSC422

- Many relevant courses in machine learning and applied machine learning in CS@UMD
  - Artificial Intelligence (CMSC421), Robotics (CMSC498F), Language (CMSC289J , CMSC470), Vision (CMSC 426), ...
- Experiment with tools and datasets
  - pyTorch, tensorflow, scikit-learn, vowpal wabbit, theano,...
  - Kaggle …
- Center for Machine Learning @UMD https://ml.umd.edu/
- Keep up to date on cutting-edge machine learning
  - Attend research seminars in the department (center for machine learning at UMIACS)
  - Other online resources (e.g., talking Machines podcast)

Beyond CMSC422

- Many opportunities to create new high impact applications with ML
- But there is a gap between theory and practice
  - With real data, Fairness, Accountability, Transparency, Privacy are key concerns
  - “To make great products, do machine learning like the great software engineer you are, not the great machine learning expert you aren’t” -Martin Zinkevich’s Best practices for ML engineering

UNIVERSITY OF
MARYLAND

Furong Huang
4124 IRB, College Park, MD 20740
301.405.8010 / furongh@umd.edu