Machine Learning

CSE 142

Xin (Eric) Wang

Wednesday, October 27, 2021

Today

- Midterm review
- Linear learning models (cont.)
  - ✓ The perceptron

Notes

- HW1 grades out today
  - Reader office hour: Thursday 2-3pm (tomorrow)
- Midterm exam – Monday, November 1 (in class)
  - **2:40-3:40pm (extra 5 mins to turn in the answers)**
    - Join the Zoom meeting minutes earlier
    - No late submissions
    - Instructions for DRC students will be sent individually
  - **With camera on all the time** (the teaching staffs will be watching);
  - No talking; No phones; No earphones;
  - No Google search; No keyboard typing;
  - Write answers on a white paper (or iPad) using your pen;
  - Picture and upload it to Gradescope before the end time.

Midterm – key topics so far

- A machine learning problem:
  - Task, experience/data, performance
  - Features
  - Models
- Types of ML: supervised, semi-supervised, unsupervised, reinforcement
- Key ML tasks: classification, clustering, regression, dimensionality reduction, anomaly detection
  - Predictive or descriptive
- ML models: geometric, probabilistic, logical

- Generalization and overfitting
- Inductive bias
- Intrinsic dimensionality
- The curse of dimensionality
- Distance measures

Midterm – key topics so far (cont.)

- Contingency table (true positives, false positives, accuracy, precision, etc.)
- Coverage plot, ROC plot

- Scoring classifier
  - Margin, loss function
- Ranking classifier (and error assessment, coverage curve)
- Empirical probabilities, Laplace correction, m-estimate
- Regression

- Concept learning and hypothesis space
  - Conjunctive hypothesis space, plus internal disjunctions
  - Pruning the hypothesis space based on data
  - Least General Generalization (LGG)
  - Complete, consistent, version space

Midterm – key topics so far (cont.)

- Feature trees and decision trees
  - Impurity measures
  - Ranking trees
  - Probability estimation trees

- Basic statistical concepts:
  - Mean, variance, covariance, covariance matrix, uncorrelated variables

- Linear least-squares regression (univariate, multivariate)
- Least-squares classifier
- Homogeneous and non-homogeneous coordinate representations
- Outliers, regularization
- Classifier margin

- The perceptron model

Midterm – kinds of questions

Primarily:

- Short answer questions to gauge understanding of basic ML concepts
- Apply understanding of concepts (e.g., contingency table, distance metrics, CHS) to specific problem/scenario
- Apply methods/algorithms (e.g., classifier, decision tree, regression function)

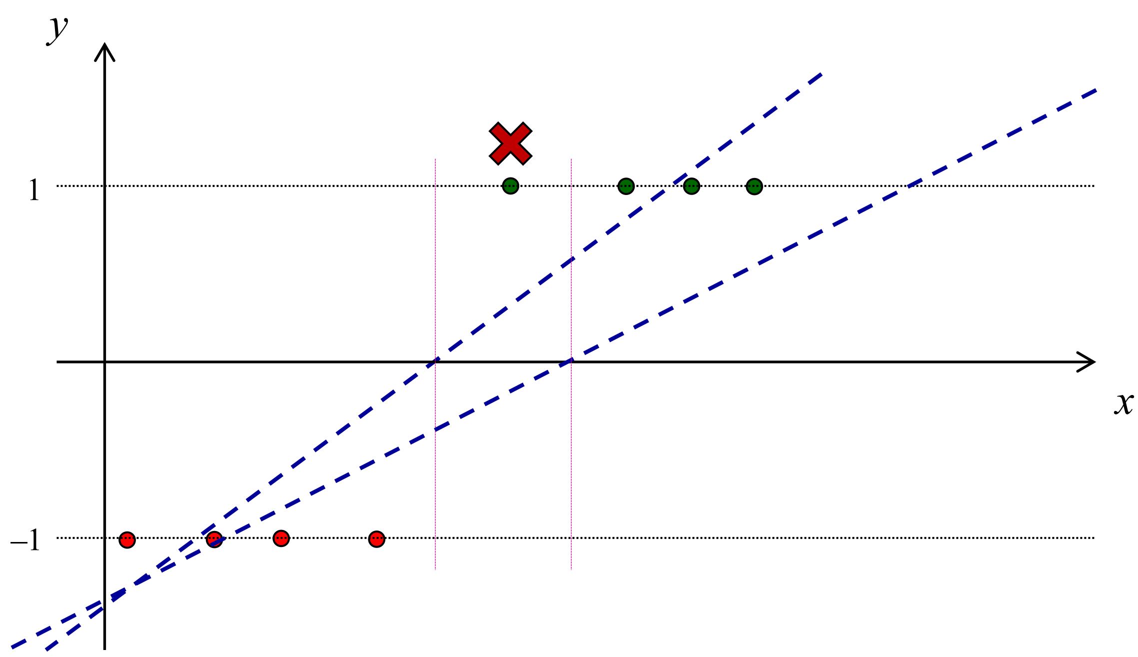
Quiz questions

Review

- For data with N input features, what is the dimensionality of the linear regression function?
  - N (fit a line to 1D data, fit a plane to 2D data, etc.)
- For data with N features, what is the dimensionality of the linear classification boundary?
  - N-1 (a line separates 2D data, a plane separates 3D data, etc.)
- For data with N features, what is (nonhomogeneous) $\boldsymbol{w}$?
  - An N-dimensional vector
- What's the output/result of linear classifier training?
  - $\boldsymbol{w}$ (homogeneous) or ($\boldsymbol{w}$, t) (non-homogeneous)

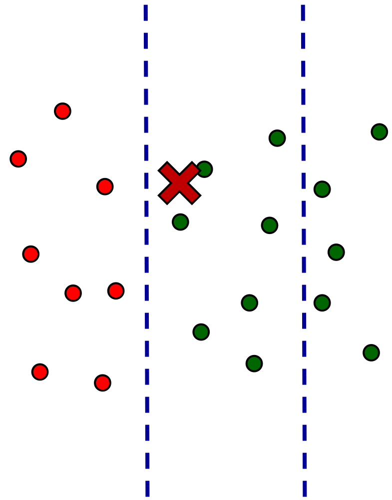
The perceptron

- A least squared classifier is not guaranteed to find a perfect decision boundary for linearly separable data



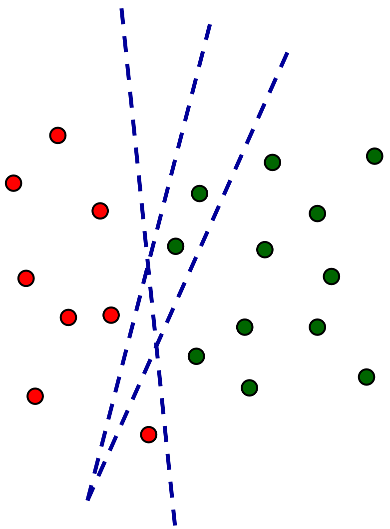
The perceptron

- A least squared classifier is not guaranteed to find a perfect decision boundary for linearly separable data



The perceptron

- The perceptron model is an iterative linear classifier that will achieve perfect separation on linearly separable data
- A perceptron iterates over the training data, updating $\boldsymbol{w}$ every time it encounters an incorrectly classified example



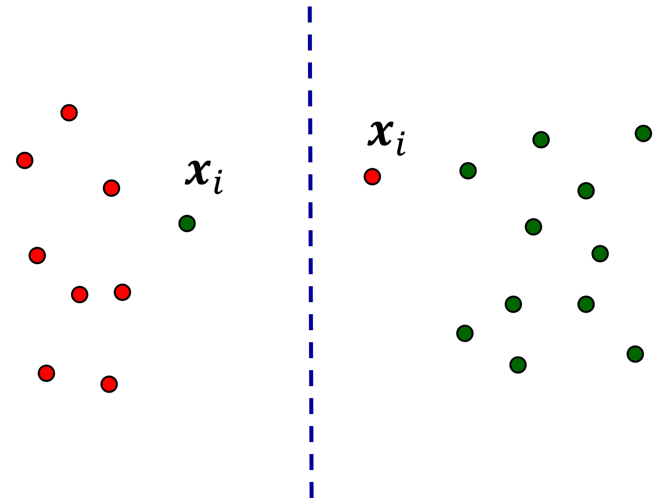
The perceptron

- The perceptron model is an iterative linear classifier that will achieve perfect separation on linearly separable data
- A perceptron iterates over the training data, updating $\boldsymbol{w}$ every time it encounters an incorrectly classified example
  - How to move the boundary for a misclassified example?
  - How much to move it?

$y_i = +1 \quad , \quad \boldsymbol{w}\boldsymbol{x}_i < t$

Goal: find $\boldsymbol{w}'$ such that $\boldsymbol{w}'\boldsymbol{x}_i > \boldsymbol{w}\boldsymbol{x}_i$

Let $\boldsymbol{w}' = \boldsymbol{w} + \eta\boldsymbol{x}_i$ $(0 < \eta \leq 1)$,
then we have $\boldsymbol{w}'\boldsymbol{x}_i = \boldsymbol{w}\boldsymbol{x}_i + \eta\boldsymbol{x}_i\boldsymbol{x}_i > \boldsymbol{w}\boldsymbol{x}_i$



Question: what if $\boldsymbol{x}_i$ is a misclassified negative sample?

The perceptron

- Update rule (homogeneous training data $\boldsymbol{x}_i \in \mathbb{R}^{k+1}$):

  $\boldsymbol{w}' = \boldsymbol{w} + \eta y_i \boldsymbol{x}_i$ where η is the learning rate, $0 < \eta \leq 1$
- Iterate through the training examples until all examples in an epoch are correctly classified
  - each pass over the data is called an epoch
- Guaranteed to converge if the training data is linearly separable – but won't converge otherwise

The perceptron training algorithm

When will this algorithm halt?

$D = \{ (\boldsymbol{x}_i, y_i) \}$

Algorithm Perceptron(D, η) – train a perceptron for linear classification.

Input : labelled training data D in homogeneous coordinates; learning rate η.
Output : weight vector $\mathbf{w}$ defining classifier $\hat{y} = \text{sign}(\mathbf{w} \cdot \mathbf{x})$.

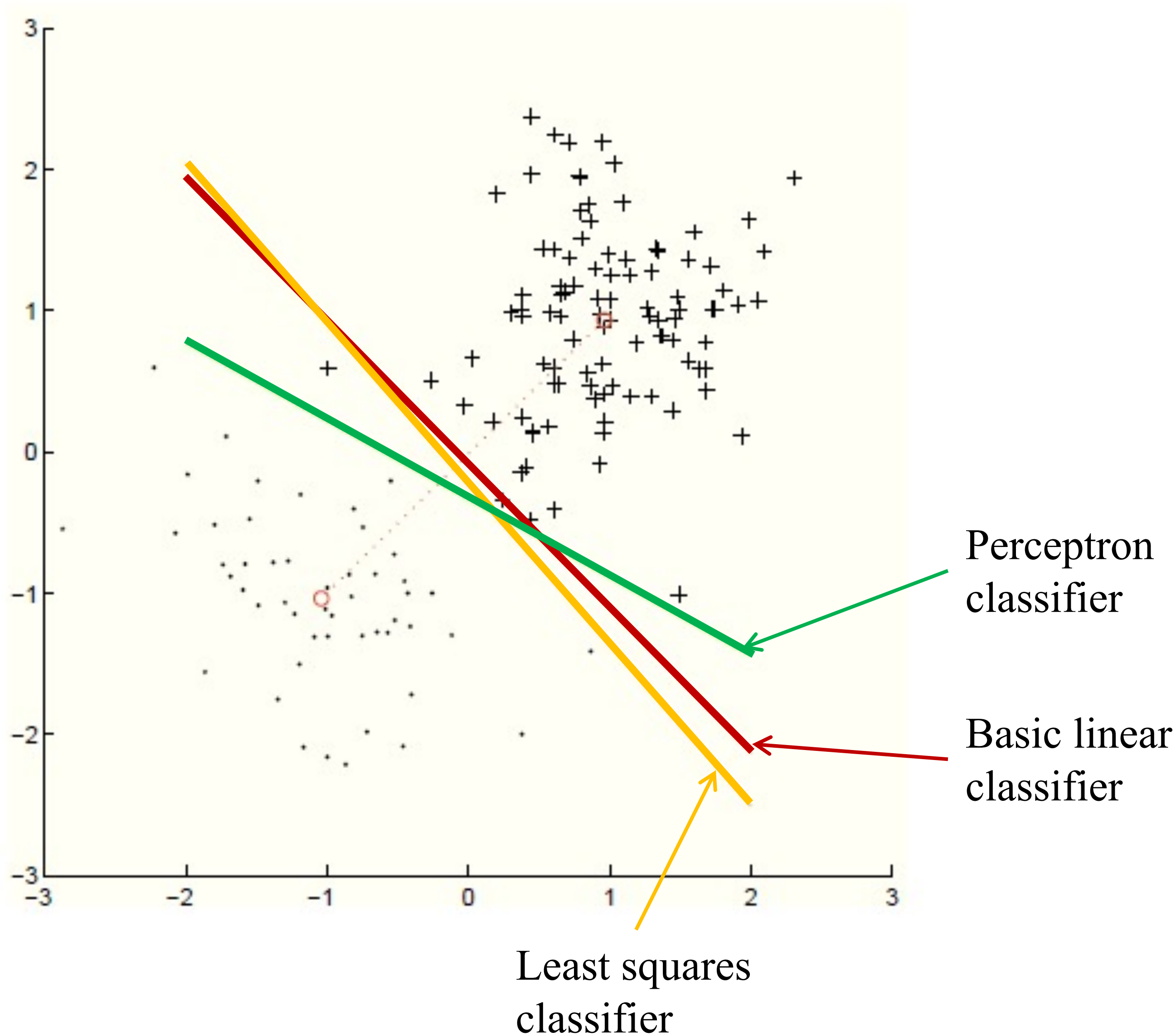
```
w ← 0 ;                    // Other initialisations of the weight vector are possible
converged ← false;
while converged = false do
    converged ← true;
    for i = 1 to |D| do
        if y_i w·x_i ≤ 0          // i.e., ŷ_i ≠ y_i      Misclassified
        then
            w ← w + η y_i x_i ;
            converged ← false;    // We changed w so haven't converged yet
        end
    end
end
```

epoch

If a positive example is misclassified, add it to $\boldsymbol{w}$
If a negative example is misclassified, subtract it from $\boldsymbol{w}$

All components of homogeneous $\boldsymbol{w}$ are updated (including $\boldsymbol{w}_{k+1} = -t$)

Linear classifier comparison



Perceptron duality

$\boldsymbol{w}' = \boldsymbol{w} + \eta y_i \boldsymbol{x}_i$

- Every time a training example $\boldsymbol{x}_i$ is misclassified, the amount $\eta y_i \boldsymbol{x}_i$ is added to the weight vector $\boldsymbol{w}$
- After training is completed, each example $\boldsymbol{x}_i$ has been misclassified α_i times ($\alpha_i \geq 0$)
- Thus the weight vector can be written as

$$\boldsymbol{w} = \eta \sum_{\boldsymbol{i}} \alpha_i y_i \boldsymbol{x}_i$$

Assuming the initial value of $\boldsymbol{w}$ was initialized to $\boldsymbol{0}$

So the weight vector is a linear combination of the training instances

- So, alternatively, we can view perceptron learning as learning the α_i coefficients and then, when finished, constructing $\boldsymbol{w}$
  - This perspective comes up again (soon) in support vector machines

Perceptron training in dual form

Algorithm DualPerceptron(D) – perceptron training in dual form.

Input : labelled training data D in homogeneous coordinates.

Output : coefficients α_i defining weight vector $\mathbf{w} = \sum_{i=1}^{|D|} \alpha_i y_i \mathbf{x}_i$.

$\alpha_i \leftarrow 0$ for $1 \le i \le |D|$;

converged←false;

while *converged* = false **do**

  converged←true;

  for $i = 1$ to $|D|$ **do**

    if $y_i \sum_{j=1}^{|D|} \alpha_j y_j \boxed{\mathbf{x}_i \cdot \mathbf{x}_j} \le 0$ **then**

      $\alpha_i \leftarrow \alpha_i + 1$;

      converged←false;

    end

  end

end

Misclassified (from regular Perceptron algorithm)

$\boxed{\textbf{if } y_i \mathbf{w} \cdot \mathbf{x}_i \le 0 \qquad // \text{ i.e., } \hat{y}_i \ne y_i}$

$$\text{if } \; y_i \underbrace{\sum_{j=1}^{|D|} \alpha_j y_j \boldsymbol{x}_j}_{\boldsymbol{w}/\eta} \cdot \boldsymbol{x}_i \le 0$$

Dot products of training examples

Contained in the Gram matrix $\mathbf{G} = \mathbf{X}^{\mathrm{T}}\mathbf{X}$

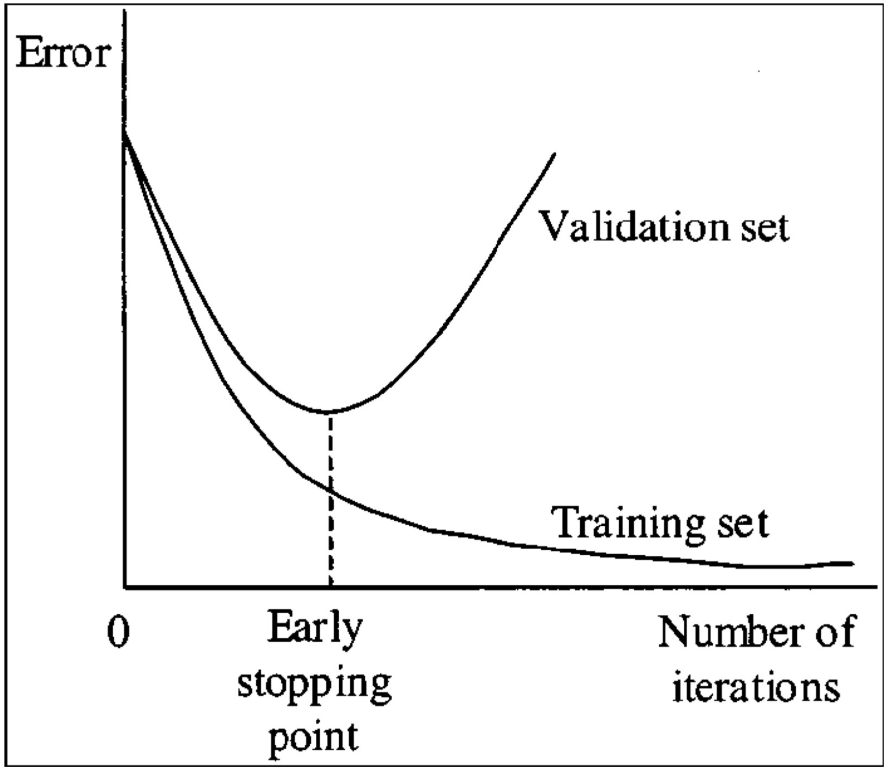
$\mathbf{G}_{ij} = \boldsymbol{x}_i^{\mathrm{T}} \boldsymbol{x}_j$

The Gram matrix is typically computed in advance for computational efficiency

This notation assumes $\mathbf{X}$ is a matrix in which each column is a vector $\boldsymbol{x}_i$

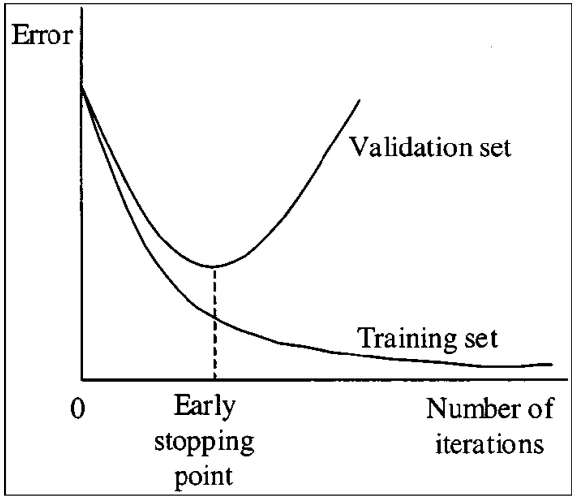
What if non linearly separable?

- In real-world problems, there is nearly always the case.
- The perceptron will not be able to converge.
  - Too many iterations may result in overfitting



Regularization in Perceptron

- Regularization in regression: penalize large weights
- Number of epochs
- Parameter averaging
  - E.g., average perceptron, voted perceptron
- Early stopping
  - use a held-out validation set
  - measure performance with current weights
  - stop when performance plateaus



Good luck to your midterm!

Be prepared