10-601 Introduction to Machine Learning

Machine Learning Department
School of Computer Science
Carnegie Mellon University

Linear Regression

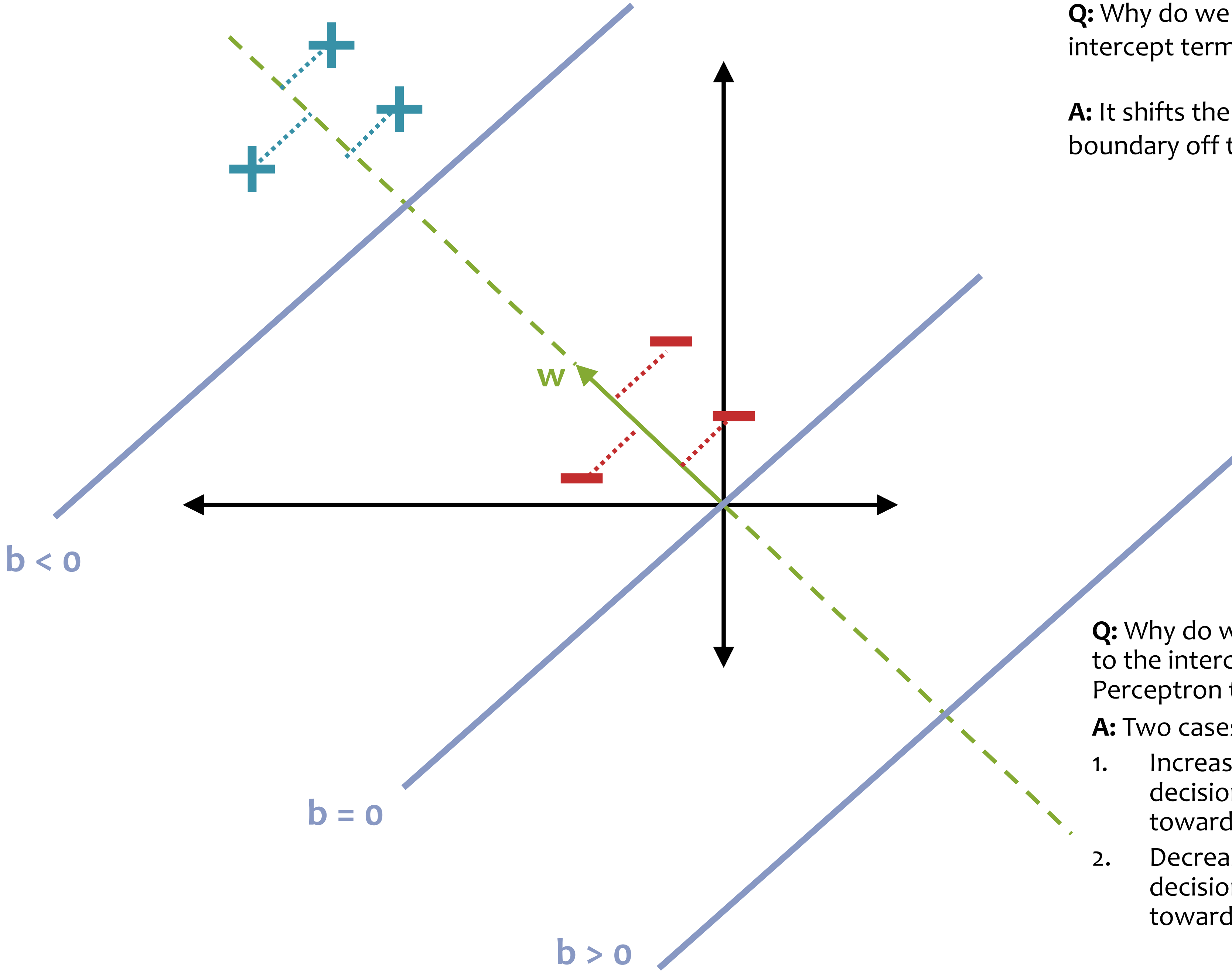
Matt Gormley
Lecture 7
Feb. 5, 2020

Reminders

- **Homework 2: Decision Trees**
  - Out: Wed, Jan. 22
  - Due: Wed, Feb. 05 at 11:59pm
- **Homework 3: KNN, Perceptron, Lin.Reg.**
  - Out: Wed, Feb. 05 (+ 1 day)
  - Due: Wed, Feb. 12 at 11:59pm
- **Today's In-Class Poll**
  - http://p7.mlcourse.org

THE PERCEPTRON ALGORITHM

Intercept Term



Q: Why do we need an intercept term?

A: It shifts the decision boundary off the origin

Q: Why do we add / subtract 1.0 to the intercept term during Perceptron training?

A: Two cases

1. Increasing b shifts the decision boundary towards the negative side
2. Decreasing b shifts the decision boundary towards the positive side

Perceptron Inductive Bias

1. Decision boundary should be linear
2. Most recent mistakes are most important (and should be corrected)

Background: Hyperplanes

Notation Trick: fold the bias *b* and the weights ***w*** into a single vector **θ** by prepending a constant to **x** and increasing dimensionality by one to get **x'**!

Hyperplane (Definition 1):

$$\mathcal{H} = \{\mathbf{x} : \mathbf{w}^T \mathbf{x} = b\}$$

Hyperplane (Definition 2):

$$\mathcal{H} = \{\mathbf{x}' : \boldsymbol{\theta}^T \mathbf{x}' = 0 \text{ and } x'_1 = 1\}$$

$$\boldsymbol{\theta} = [b, w_1, \ldots, w_M]^T$$

Half-spaces:

$$\mathcal{H}^+ = \{\mathbf{x} : \boldsymbol{\theta}^T \mathbf{x} > 0 \text{ and } x_0 = 1\}$$

$$\mathcal{H}^- = \{\mathbf{x} : \boldsymbol{\theta}^T \mathbf{x} < 0 \text{ and } x_0 = 1\}$$

(Online) Perceptron Algorithm

Data: Inputs are continuous vectors of length *M*. Outputs are discrete.

$$(\mathbf{x}^{(1)}, y^{(1)}), (\mathbf{x}^{(2)}, y^{(2)}), \ldots$$

where $\mathbf{x} \in \mathbb{R}^M$ and $y \in \{+1, -1\}$

Prediction: Output determined by hyperplane.

$$\hat{y} = h_{\boldsymbol{\theta}}(\mathbf{x}) = \text{sign}(\boldsymbol{\theta}^T \mathbf{x})$$

$$\text{sign}(a) = \begin{cases} 1, & \text{if } a \geq 0 \\ -1, & \text{otherwise} \end{cases}$$

Assume $\boldsymbol{\theta} = [b, w_1, \ldots, w_M]^T$ and $x_0 = 1$

Learning: Iterative procedure:

- **initialize** parameters to vector of all zeroes
- **while** not converged
  - **receive** next example ($\mathbf{x}^{(i)}$, $y^{(i)}$)
  - **predict** y' = h($\mathbf{x}^{(i)}$)
  - **if** positive mistake: **add** $\mathbf{x}^{(i)}$ to parameters
  - **if** negative mistake: **subtract** $\mathbf{x}^{(i)}$ from parameters

(Online) Perceptron Algorithm

Data: Inputs are continuous vectors of length *M*. Outputs are discrete.

$$(\mathbf{x}^{(1)}, y^{(1)}), (\mathbf{x}^{(2)}, y^{(2)}), \ldots$$

where $\mathbf{x} \in \mathbb{R}^M$ and $y \in \{+1, -1\}$

Prediction: Output determine

$$\hat{y} = h_{\boldsymbol{\theta}}(\mathbf{x}) = \text{sign}(\boldsymbol{\theta}^T \mathbf{x})$$

Assume $\boldsymbol{\theta} = [b, w_1, \ldots, w_M]$

Implementation Trick: same behavior as our "*add on positive mistake and subtract on negative mistake*" version, because $y^{(i)}$ takes care of the sign

Learning:

Algorithm 1 Perceptron Learning Alg

```
procedure PERCEPTRON(𝒟 = {(x
    θ ← 0                              ▷ Initialize parameters
    for i ∈ {1, 2, . . .} do           ▷ For each example
        ŷ ← sign(θ^T x^(i))            ▷ Predict
        if ŷ ≠ y^(i) then              ▷ If mistake
            θ ← θ + y^(i) x^(i)        ▷ Update parameters
    return θ
```

(Batch) Perceptron Algorithm

Learning for Perceptron also works if we have a fixed training dataset, D. We call this the "batch" setting in contrast to the "online" setting that we've discussed so far.

Algorithm 1 Perceptron Learning Algorithm (Batch)

procedure PERCEPTRON($\mathcal{D} = \{(\mathbf{x}^{(1)}, y^{(1)}), \ldots, (\mathbf{x}^{(N)}, y^{(N)})\}$)
    $\boldsymbol{\theta} \leftarrow \mathbf{0}$ ▷ Initialize parameters
    while not converged **do**
        for $i \in \{1, 2, \ldots, N\}$ **do** ▷ For each example
            $\hat{y} \leftarrow \text{sign}(\boldsymbol{\theta}^T \mathbf{x}^{(i)})$ ▷ Predict
            if $\hat{y} \neq y^{(i)}$ **then** ▷ If mistake
                $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + y^{(i)} \mathbf{x}^{(i)}$ ▷ Update parameters
    return $\boldsymbol{\theta}$

(Batch) Perceptron Algorithm

Learning for Perceptron also works if we have a fixed training dataset, D. We call this the “batch” setting in contrast to the “online” setting that we’ve discussed so far.

Discussion:

The Batch Perceptron Algorithm can be derived in two ways.

1. By extending the online Perceptron algorithm to the batch setting (as mentioned above)
2. By applying **Stochastic Gradient Descent (SGD)** to minimize a so-called **Hinge Loss** on a linear separator

Extensions of Perceptron

- **Voted Perceptron**
  - generalizes better than (standard) perceptron
  - memory intensive (keeps around every weight vector seen during training, so each one can vote)
- **Averaged Perceptron**
  - empirically similar performance to voted perceptron
  - can be implemented in a memory efficient way (running averages are efficient)
- **Kernel Perceptron**
  - Choose a kernel K(x', x)
  - Apply the **kernel trick** to Perceptron
  - Resulting algorithm is **still very simple**
- **Structured Perceptron**
  - Basic idea can also be applied when **y** ranges over an exponentially large set
  - Mistake bound **does not** depend on the size of that set

Perceptron Exercises

Question:

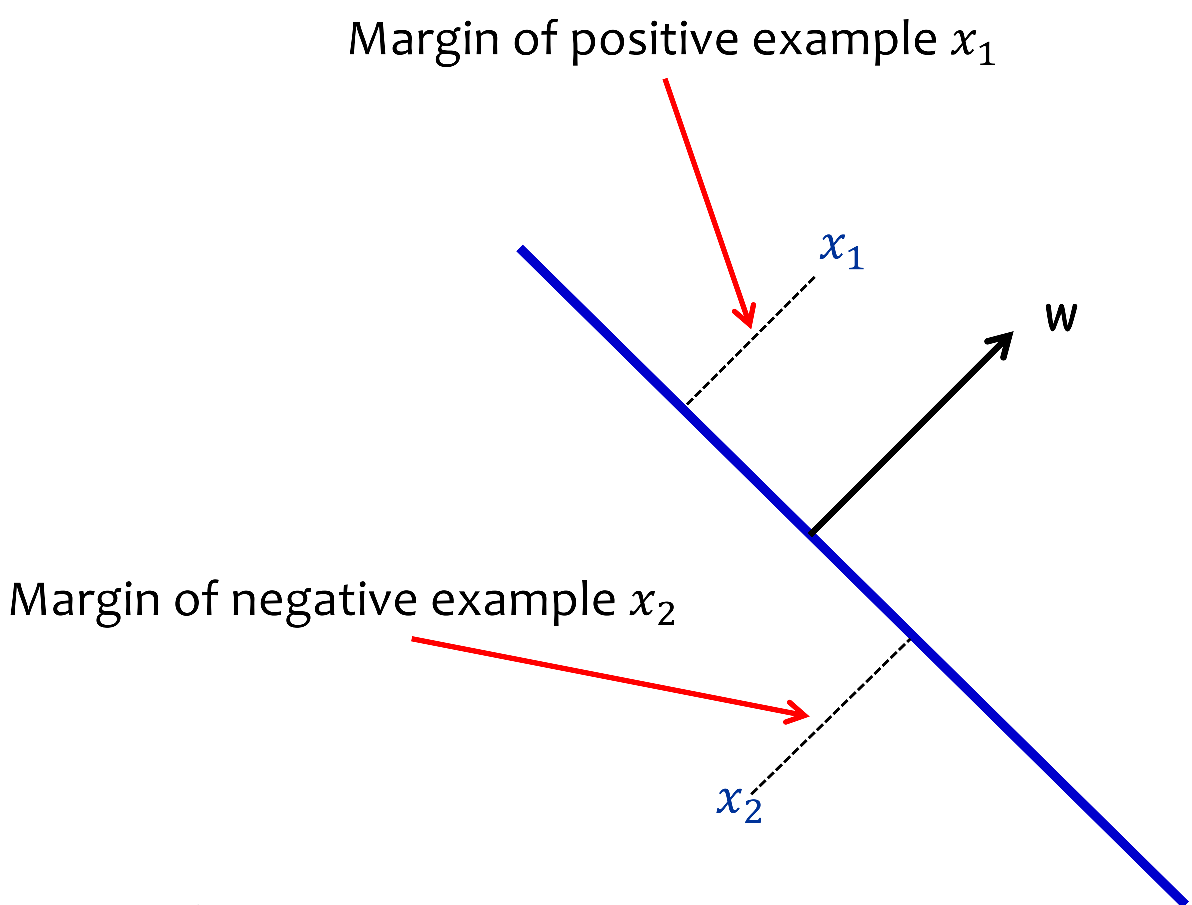
The parameter vector $\mathbf{w}$ *learned by the Perceptron algorithm can be* ***written as a linear combination*** *of the feature vectors* $\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots, \mathbf{x}^{(N)}$.

A. *True, if you replace "linear" with "polynomial" above*
B. *True, for all datasets*
C. *False, for all datasets*
D. *True, but only for certain datasets*
E. *False, but only for certain datasets*

ANALYSIS OF PERCEPTRON

Geometric Margin

Definition: The margin of example x w.r.t. a linear sep. w is the distance from x to the plane $w \cdot x = 0$ (or the negative if on wrong side)

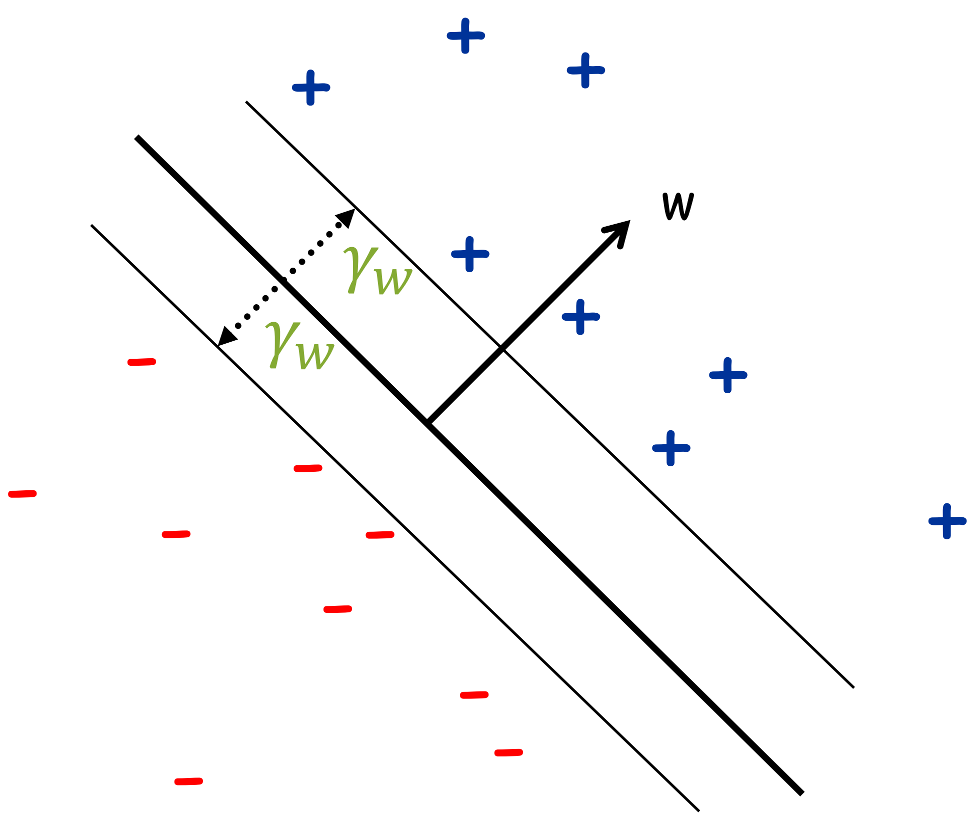


Slide from Nina Balcan

Geometric Margin

Definition: The margin of example x w.r.t. a linear sep. w is the distance from x to the plane $w \cdot x = 0$ (or the negative if on wrong side)

Definition: The margin γ_w of a set of examples S wrt a linear separator w is the smallest margin over points $x \in S$.



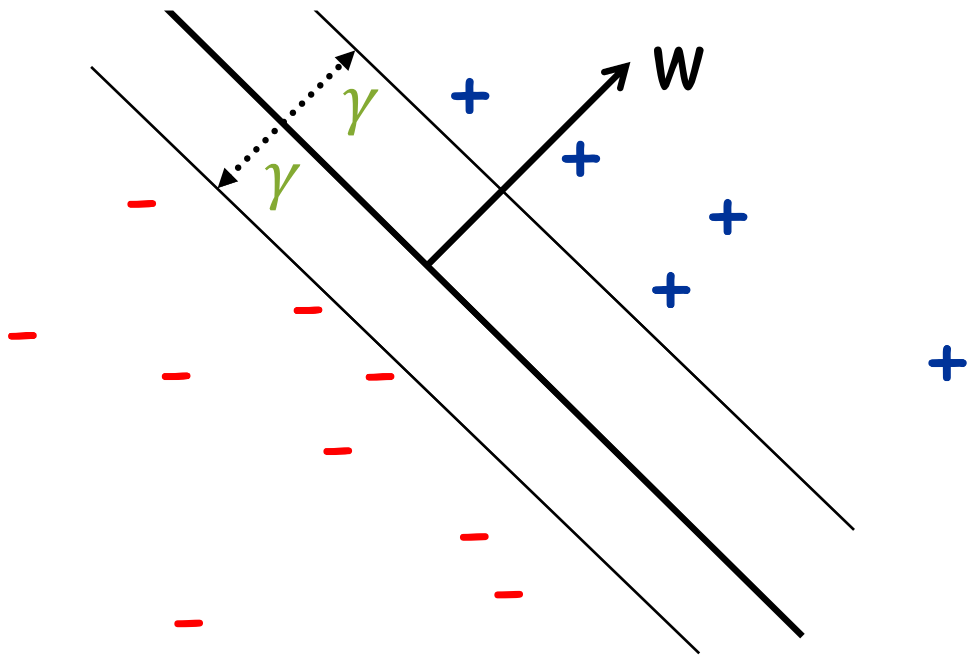
Slide from Nina Balcan

Geometric Margin

Definition: The margin of example x w.r.t. a linear sep. w is the distance from x to the plane $w \cdot x = 0$ (or the negative if on wrong side)

Definition: The margin γ_w of a set of examples S wrt a linear separator w is the smallest margin over points $x \in S$.

Definition: The margin γ of a set of examples S is the maximum γ_w over all linear separators w.

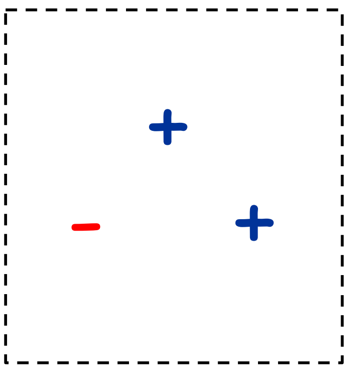


Slide from Nina Balcan

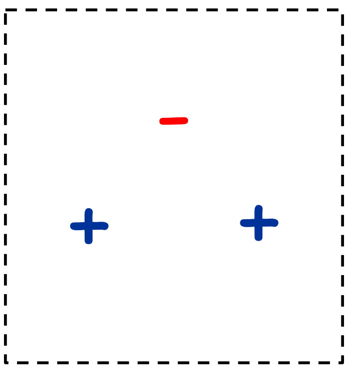
Linear Separability

Def: For a **binary classification** problem, a set of examples S is **linearly separable** if there exists a linear decision boundary that can separate the points

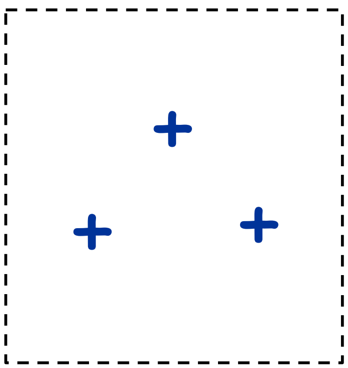
Case 1:



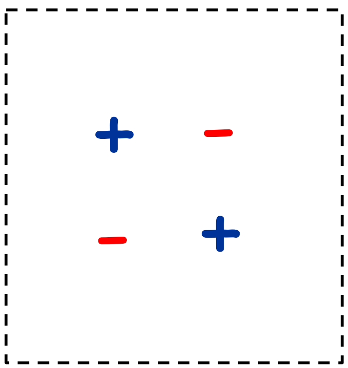
Case 2:



Case 3:



Case 4:

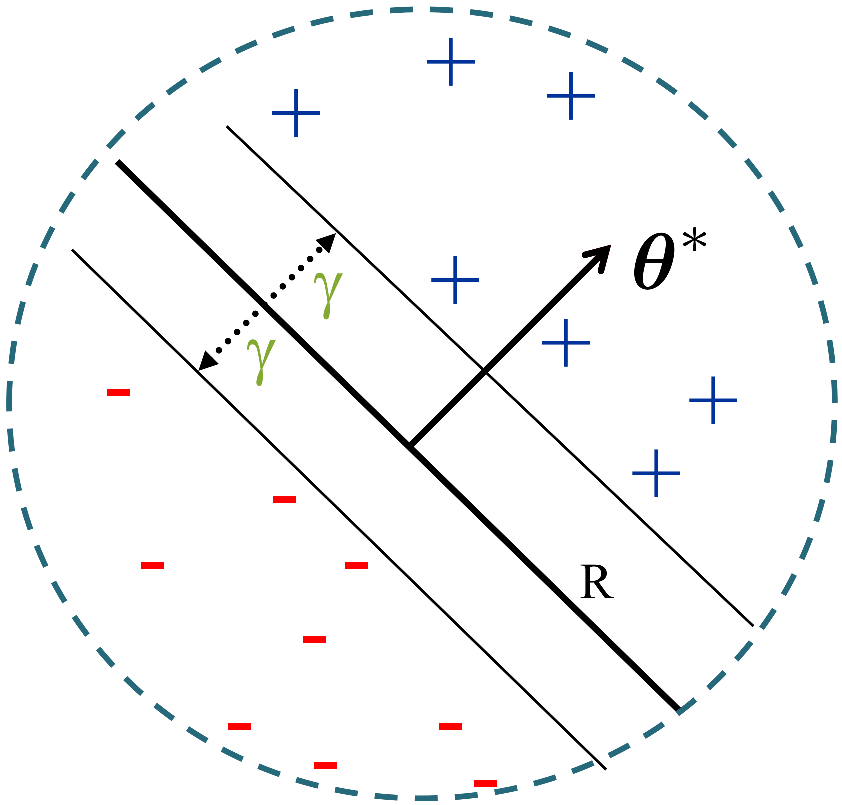


Analysis: Perceptron

Perceptron Mistake Bound

Guarantee: If data has margin γ and all points inside a ball of radius R, then Perceptron makes $\leq (R/\gamma)^2$ mistakes.

(Normalized margin: multiplying all points by 100, or dividing all points by 100, doesn't change the number of mistakes; algo is invariant to scaling.)



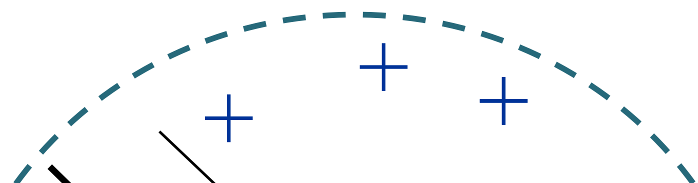
Slide adapted from Nina Balcan

Analysis: Perceptron

Perceptron Mistake Bound

Guarantee: If data has margin γ and all points inside a ball of radius R, then Perceptron makes $\leq (R/\gamma)^2$ mistakes.

(Normalized margin: multiplying all points by 100, or dividing all points by 100, doesn't change the number of mistakes; algo is invariant to scaling.)



Def: We say that the (batch) perceptron algorithm has **converged** if it stops making mistakes on the training data (perfectly classifies the training data).

Main Takeaway: For **linearly separable** data, if the perceptron algorithm cycles repeatedly through the data, it will **converge** in a finite # of steps.

Slide adapted from Nina Balcan

Analysis: Perceptron

Perceptron Mistake Bound

Theorem 0.1 (Block (1962), Novikoff (1962)).
Given dataset: $\mathcal{D} = \{(\mathbf{x}^{(i)}, y^{(i)})\}_{i=1}^{N}$.
Suppose:

1. *Finite size inputs:* $||x^{(i)}|| \leq R$
2. *Linearly separable data:* $\exists \boldsymbol{\theta}^*$ *s.t.* $||\boldsymbol{\theta}^*|| = 1$ *and* $y^{(i)}(\boldsymbol{\theta}^* \cdot \mathbf{x}^{(i)}) \geq \gamma, \forall i$

Then: The number of mistakes made by the Perceptron algorithm on this dataset is

$$k \leq (R/\gamma)^2$$

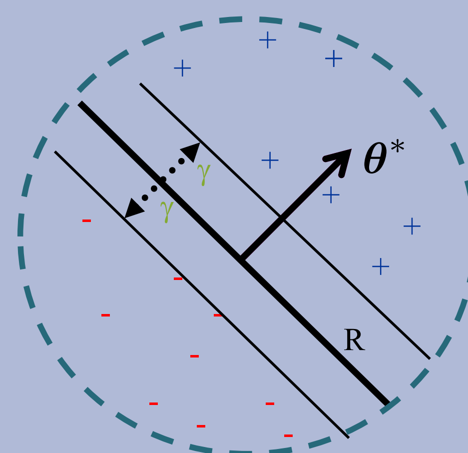


Figure from Nina Balcan

Analysis: Percept

Common Misunderstanding:
The **radius** is **centered at the origin,** not at the center of the points.

Perceptron Mistake Boun

Theorem 0.1 (Block (1962), Novikoff (19

Given dataset: $\mathcal{D} = \{(\mathbf{x}^{(i)}, y^{(i)})\}_{i=1}^{N}$

Suppose:

1. *Finite size inputs:* $||x^{(i)}|| \leq R$
2. *Linearly separable data:* $\exists \boldsymbol{\theta}^*$ *s.t.* $||\boldsymbol{\theta}^*|| = 1$ *and* $y^{(i)}(\boldsymbol{\theta}^* \cdot \mathbf{x}^{(i)}) \geq \gamma, \forall i$

Then: The number of mistakes made by the Perceptron algorithm on this dataset is

$$k \leq (R/\gamma)^2$$

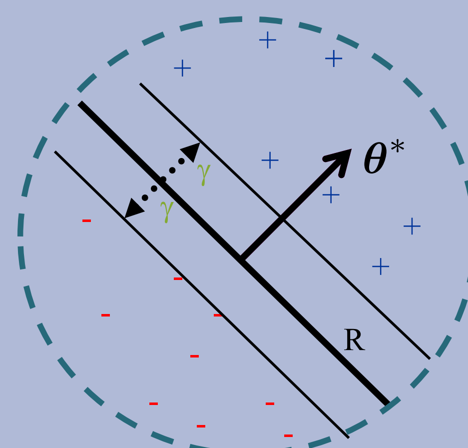


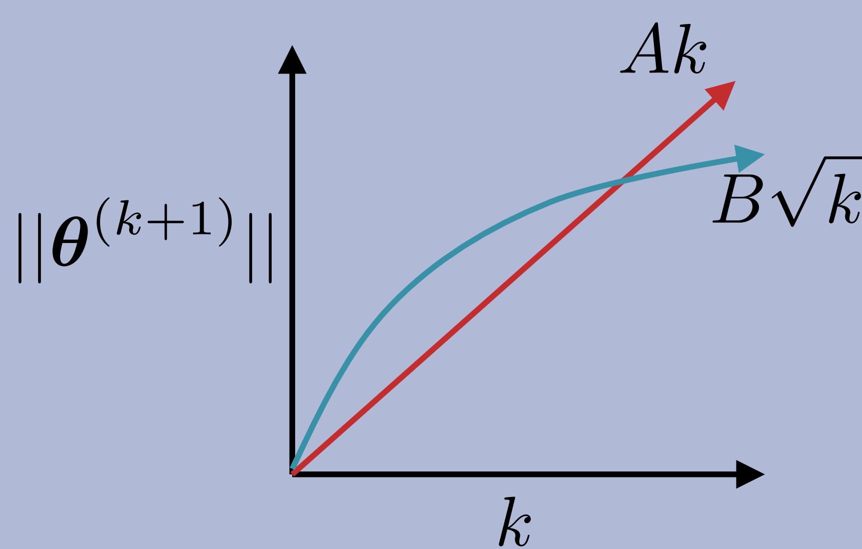
Figure from Nina Balcan

Analysis: Perceptron

Proof of Perceptron Mistake Bound:

We will show that there exist constants A and B s.t.

$$Ak \leq ||\boldsymbol{\theta}^{(k+1)}|| \leq B\sqrt{k}$$



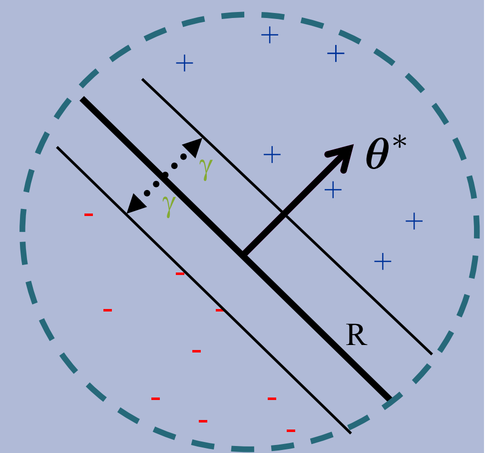
Analysis: Perceptron

Theorem 0.1 (Block (1962), Novikoff (1962)).
Given dataset: $\mathcal{D} = \{(\mathbf{x}^{(i)}, y^{(i)})\}_{i=1}^{N}$.
Suppose:

1. *Finite size inputs:* $||x^{(i)}|| \leq R$
2. *Linearly separable data:* $\exists \boldsymbol{\theta}^*$ *s.t.* $||\boldsymbol{\theta}^*|| = 1$ *and* $y^{(i)}(\boldsymbol{\theta}^* \cdot \mathbf{x}^{(i)}) \geq \gamma, \forall i$

Then: The number of mistakes made by the Perceptron algorithm on this dataset is

$$k \leq (R/\gamma)^2$$



Algorithm 1 Perceptron Learning Algorithm (Online)

procedure PERCEPTRON($\mathcal{D} = \{(\mathbf{x}^{(1)}, y^{(1)}), (\mathbf{x}^{(2)}, y^{(2)}), \ldots\}$)
  $\boldsymbol{\theta} \leftarrow \mathbf{0}$, $k = 1$ ▷ Initialize parameters
  for $i \in \{1, 2, \ldots\}$ **do** ▷ For each example
    if $y^{(i)}(\boldsymbol{\theta}^{(k)} \cdot \mathbf{x}^{(i)}) \leq 0$ **then** ▷ If mistake
      $\boldsymbol{\theta}^{(k+1)} \leftarrow \boldsymbol{\theta}^{(k)} + y^{(i)}\mathbf{x}^{(i)}$ ▷ Update parameters
      $k \leftarrow k + 1$
  return $\boldsymbol{\theta}$

Analysis: Perceptron

Proof of Perceptron Mistake Bound:

Part 1: for some A, $Ak \leq ||\boldsymbol{\theta}^{(k+1)}||$

$$\boldsymbol{\theta}^{(k+1)} \cdot \boldsymbol{\theta}^* = (\boldsymbol{\theta}^{(k)} + y^{(i)}\mathbf{x}^{(i)})\boldsymbol{\theta}^*$$

by Perceptron algorithm update

$$= \boldsymbol{\theta}^{(k)} \cdot \boldsymbol{\theta}^* + y^{(i)}(\boldsymbol{\theta}^* \cdot \mathbf{x}^{(i)})$$

$$\geq \boldsymbol{\theta}^{(k)} \cdot \boldsymbol{\theta}^* + \gamma$$

by assumption

$$\Rightarrow \boldsymbol{\theta}^{(k+1)} \cdot \boldsymbol{\theta}^* \geq k\gamma$$

by induction on k since $\theta^{(1)} = \mathbf{0}$

$$\Rightarrow ||\boldsymbol{\theta}^{(k+1)}|| \geq k\gamma$$

since $\underbrace{||\mathbf{w}|| \times ||\mathbf{u}|| \geq \mathbf{w} \cdot \mathbf{u}}_{\text{Cauchy-Schwartz inequality}}$ and $||\theta^*|| = 1$

Analysis: Perceptron

Proof of Perceptron Mistake Bound:

Part 2: for some B, $||\boldsymbol{\theta}^{(k+1)}|| \leq B\sqrt{k}$

$$||\boldsymbol{\theta}^{(k+1)}||^2 = ||\boldsymbol{\theta}^{(k)} + y^{(i)}\mathbf{x}^{(i)}||^2$$

by Perceptron algorithm update

$$= ||\boldsymbol{\theta}^{(k)}||^2 + (y^{(i)})^2||\mathbf{x}^{(i)}||^2 + 2y^{(i)}(\boldsymbol{\theta}^{(k)} \cdot \mathbf{x}^{(i)})$$

$$\leq ||\boldsymbol{\theta}^{(k)}||^2 + (y^{(i)})^2||\mathbf{x}^{(i)}||^2$$

since kth mistake $\Rightarrow y^{(i)}(\boldsymbol{\theta}^{(k)} \cdot \mathbf{x}^{(i)}) \leq 0$

$$= ||\boldsymbol{\theta}^{(k)}||^2 + R^2$$

since $(y^{(i)})^2||\mathbf{x}^{(i)}||^2 = ||\mathbf{x}^{(i)}||^2 = R^2$ by assumption and $(y^{(i)})^2 = 1$

$$\Rightarrow ||\boldsymbol{\theta}^{(k+1)}||^2 \leq kR^2$$

by induction on k since $(\theta^{(1)})^2 = 0$

$$\Rightarrow ||\boldsymbol{\theta}^{(k+1)}|| \leq \sqrt{k}R$$

Analysis: Perceptron

Proof of Perceptron Mistake Bound:

Part 3: Combining the bounds finishes the proof.

$$k\gamma \leq ||\boldsymbol{\theta}^{(k+1)}|| \leq \sqrt{k}R$$

$$\Rightarrow k \leq \underbrace{(R/\gamma)^2}_{\text{The total number of mistakes must be less than this}}$$

Analysis: Perceptron

What if the data is *not* linearly separable?

1. Perceptron will **not converge** in this case (it can't!)
2. However, Freund & Schapire (1999) show that by projecting the points (hypothetically) into a higher dimensional space, we can achieve a similar bound on the number of mistakes made on **one pass** through the sequence of examples

Theorem 2. *Let* $\langle(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m)\rangle$ *be a sequence of labeled examples with* $\|\mathbf{x}_i\| \leq R$. *Let* $\mathbf{u}$ *be any vector with* $\|\mathbf{u}\| = 1$ *and let* $\gamma > 0$. *Define the deviation of each example as*

$$d_i = \max\{0, \gamma - y_i(\mathbf{u} \cdot \mathbf{x}_i)\},$$

and define $D = \sqrt{\sum_{i=1}^m d_i^2}$. *Then the number of mistakes of the online perceptron algorithm on this sequence is bounded by*

$$\left(\frac{R + D}{\gamma}\right)^2.$$

Perceptron Exercises

Question:

Unlike Decision Trees and K-Nearest Neighbors, the Perceptron algorithm ***does not suffer from overfitting*** *because it does not have any hyperparameters that could be over-tuned on the training data.*

A. *True*
B. *False*
C. *True and False*

Summary: Perceptron

- Perceptron is a **linear classifier**
- **Simple learning algorithm**: when a mistake is made, add / subtract the features
- Perceptron will converge if the data are **linearly separable**, it will **not** converge if the data are **linearly inseparable**
- For linearly separable and inseparable data, we can **bound the number of mistakes** (geometric argument)
- **Extensions** support nonlinear separators and structured prediction

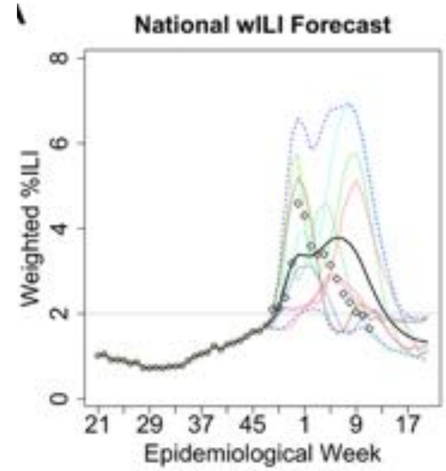
Perceptron Learning Objectives

You should be able to…

- Explain the difference between online learning and batch learning
- Implement the perceptron algorithm for binary classification [CIML]
- Determine whether the perceptron algorithm will converge based on properties of the dataset, and the limitations of the convergence guarantees
- Describe the inductive bias of perceptron and the limitations of linear models
- Draw the decision boundary of a linear model
- Identify whether a dataset is linearly separable or not
- Defend the use of a bias term in perceptron

REGRESSION

Regression



Goal:

- Given a training dataset of pairs ($\mathbf{x}$,y) where
  - $\mathbf{x}$ is a vector
  - y is a scalar
- Learn a function (aka. curve or line) y' = h(x) that best fits the training data

Example Applications:

- Stock price prediction
- Forecasting epidemics
- Speech synthesis
- Generation of images (e.g. *Deep Dream*)
- Predicting the number of tourists on Machu Picchu on a given day

Regression

Example Application:
Forecasting Epidemics

- Input features, **x**: attributes of the epidemic
- Output, y: Weighted %ILI, prevalence of the disease
- Setting: observe past prevalence to predict future prevalence

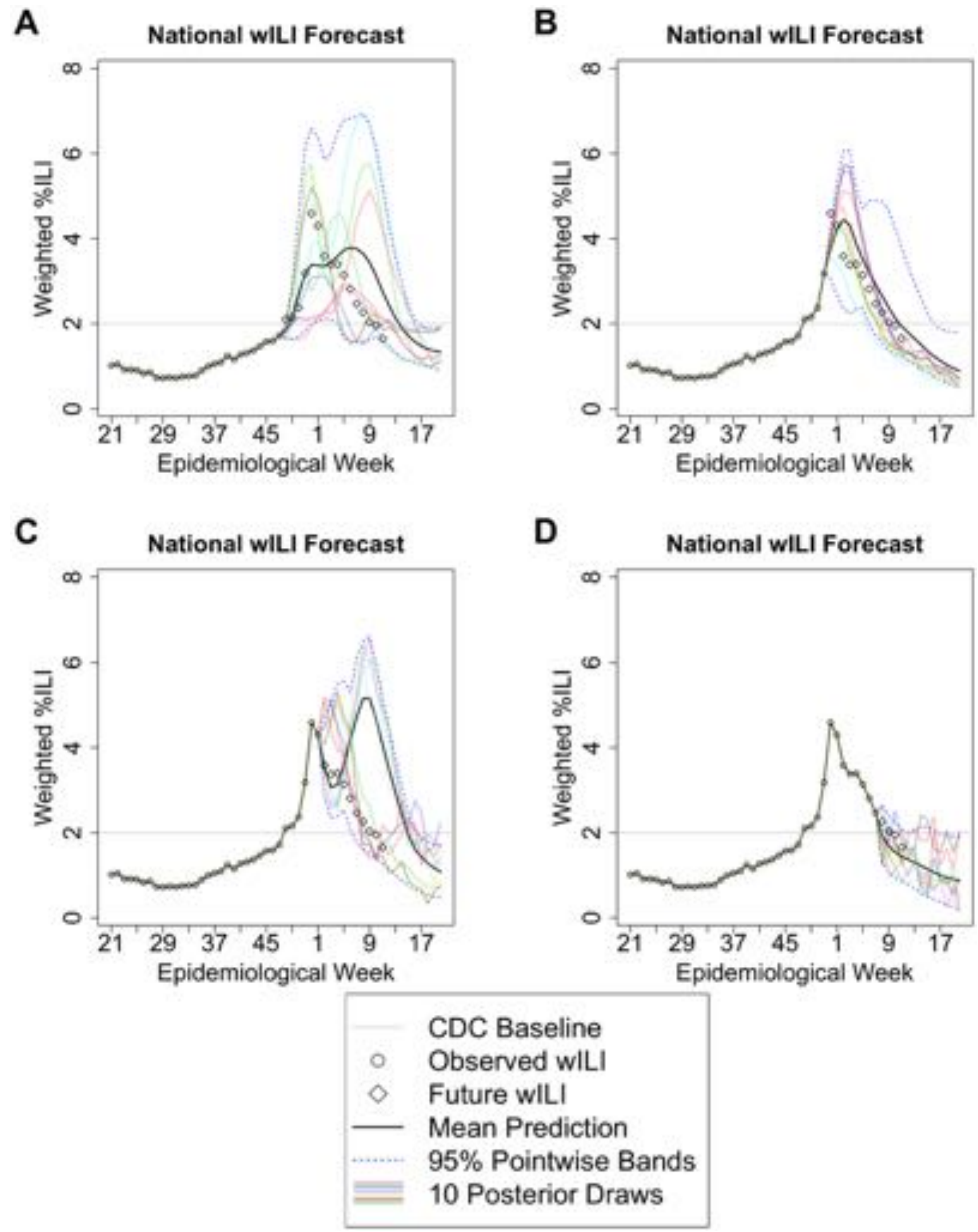
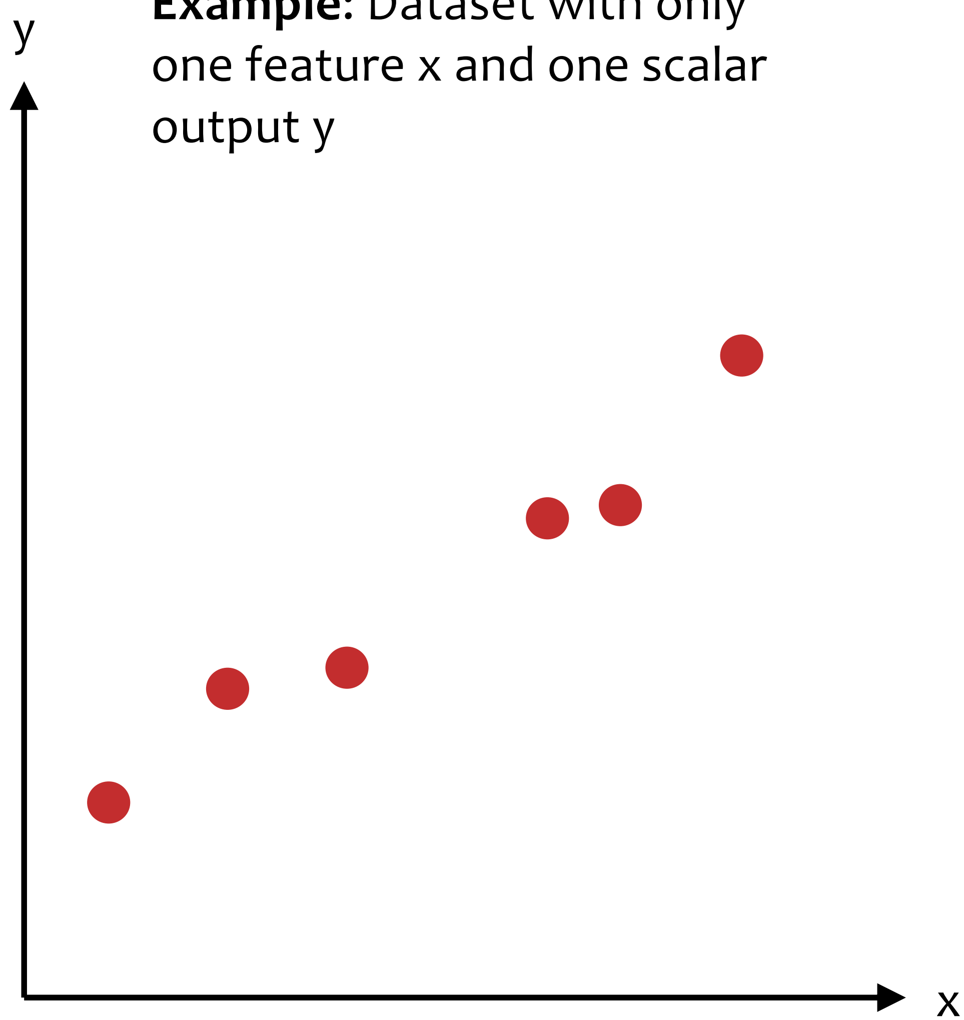


Fig 2. 2013–2014 national forecast, retrospectively, using the final revisions of wILI values, using revised wILI data through epidemiological weeks (A) 47, (B) 51, (C) 1, and (D) 7.

Figure from Brooks et al. (2015)

Regression

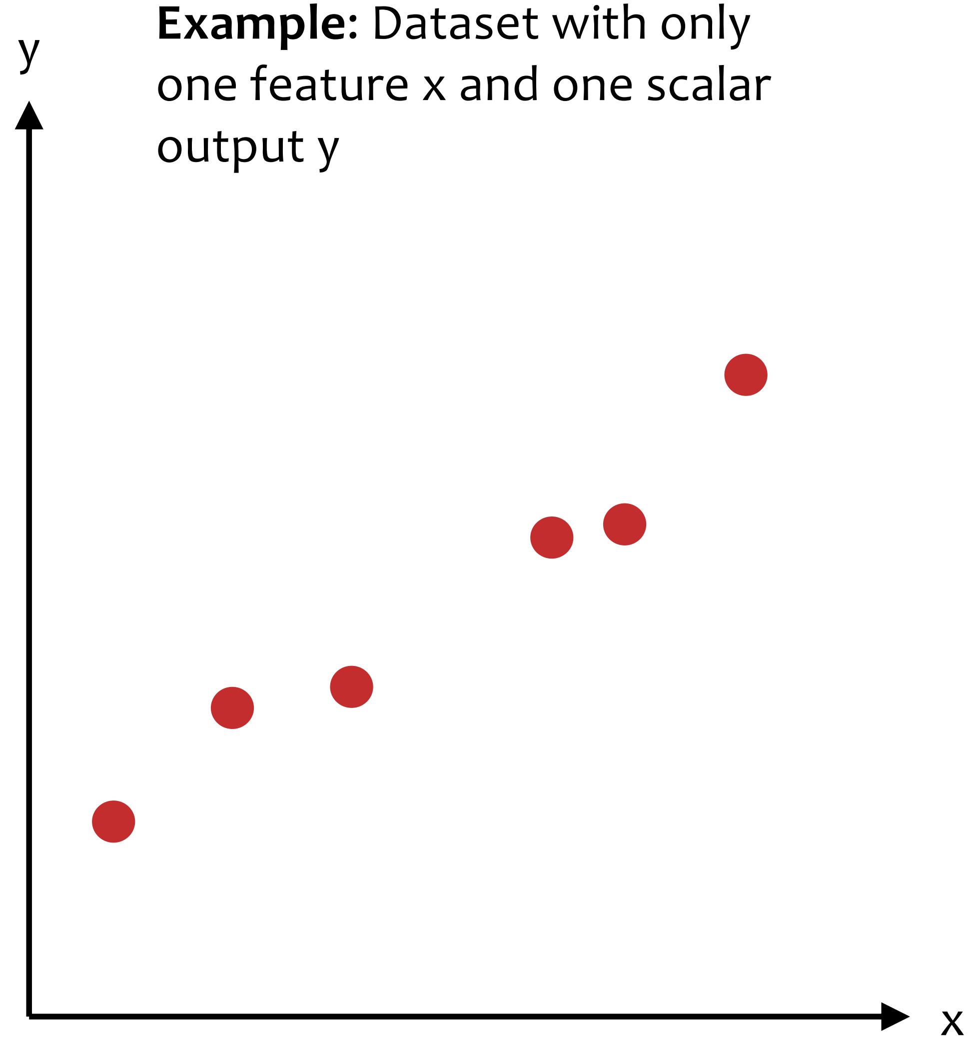
Example: Dataset with only one feature x and one scalar output y



Q: What is the function that best fits these points?

k-NN Regression

Example: Dataset with only one feature x and one scalar output y



k=1 Nearest Neighbor Regression

- *Train*: store all (x, y) pairs
- *Predict*: pick the nearest x in training data and return its y

k=2 Nearest Neighbor Distance Weighted Regression

- *Train*: store all (x, y) pairs
- *Predict*: pick the nearest two instances $x^{(n1)}$ and $x^{(n2)}$ in training data and return the weighted average of their y values

LINEAR REGRESSION

Regression Problems

Chalkboard

- Definition of Regression
- Linear functions
- Residuals
- Notation trick: fold in the intercept