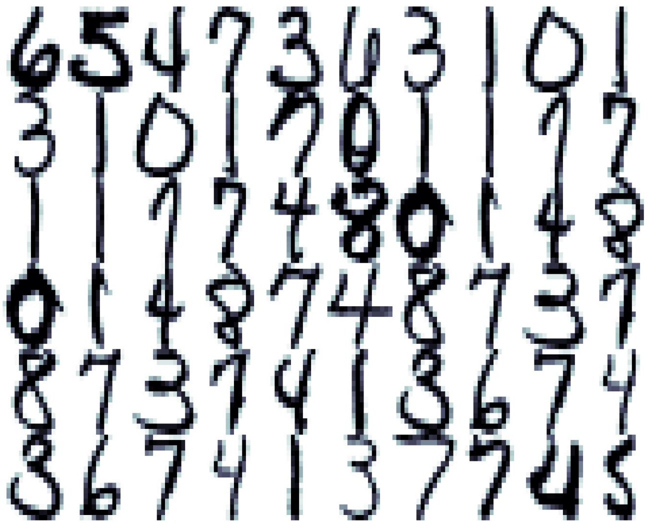
CS6375: Machine Learning

Gautam Kunapuli

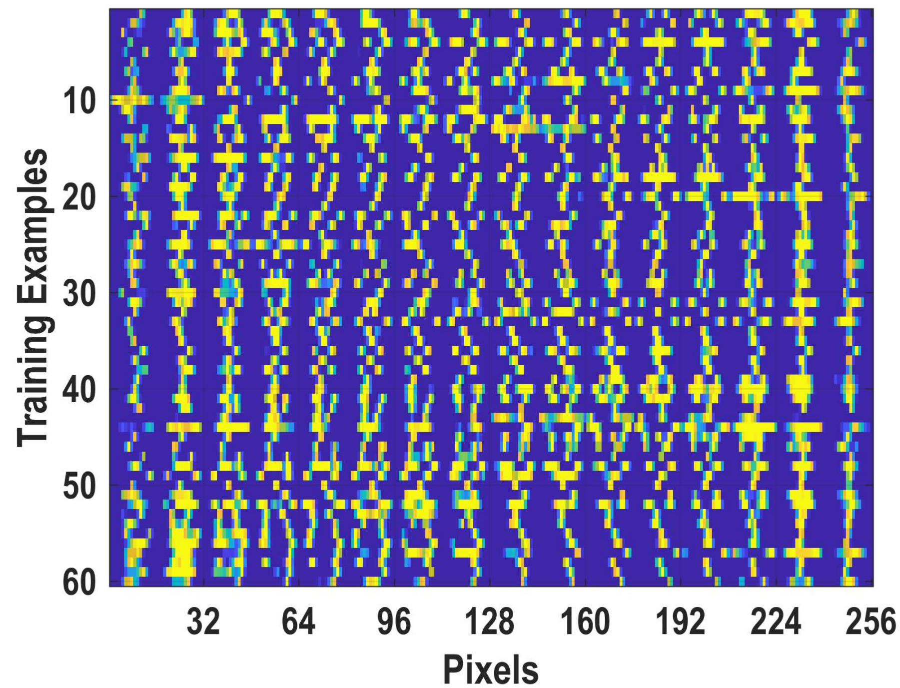
Linear Classification: Perceptron

THE UNIVERSITY OF TEXAS AT DALLAS
Erik Jonsson School of Engineering and Computer Science

Example: Handwritten Digit Recognition



The ***United States Postal Service Zip Code Database*** *contains 16 x 16 pixel images of scanned* ***handwritten*** *digits. Typical* ***human error rate is around 2.5%****.*

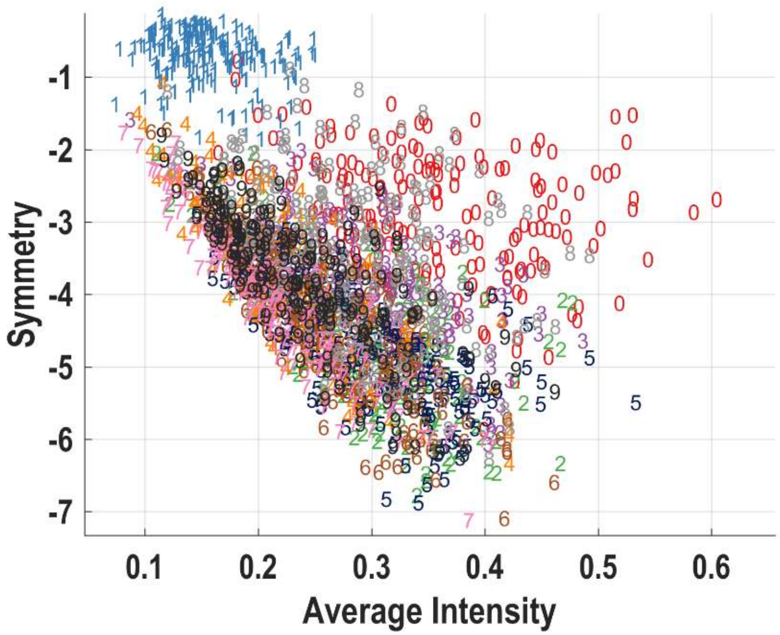


We can reshape each 16 x 16 image matrix into a 256 x 1 image vector; each row is a digit, represented by its 256 (= 16 x 16) pixels.

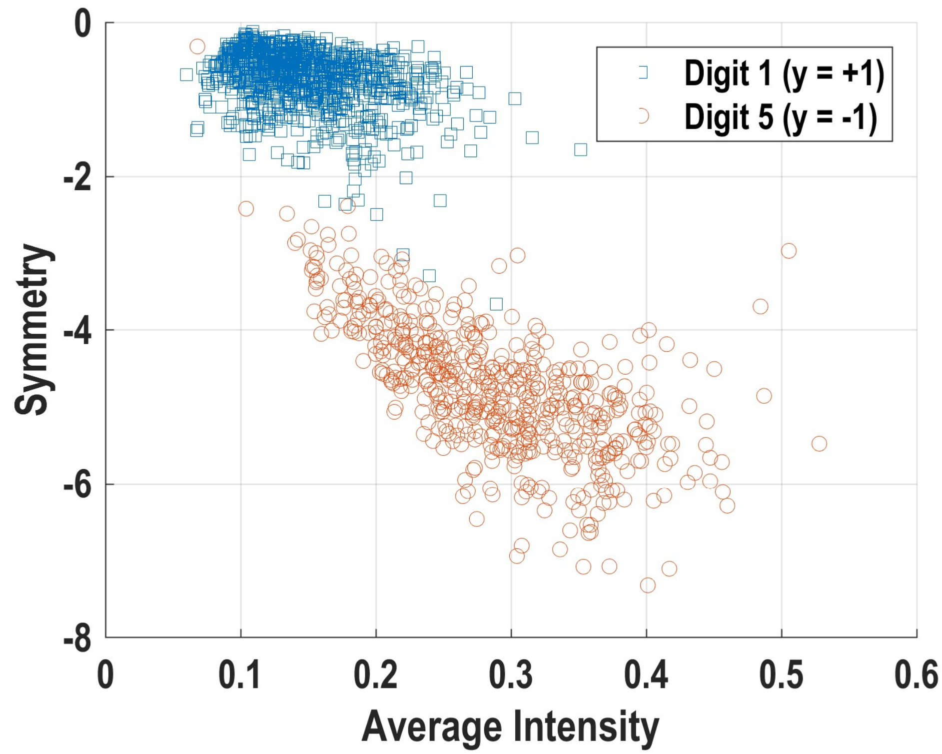
Machine Learning Task: Identify digits from data automatically; that is **classify** each image as a digit. This is an instance of a **classification task**. As there are 10 digits, this is an example of a **multi-class classification problem**.

Example: Handwritten Digit Recognition

*Alternately, we can extract two informative features from each image: **intensity** and **symmetry**. Data set is 2-dimensional.*



*Consider the simpler problem of learning a classifier to separate the 1s from the 5s. This simpler problem is also **linearly separable**.*



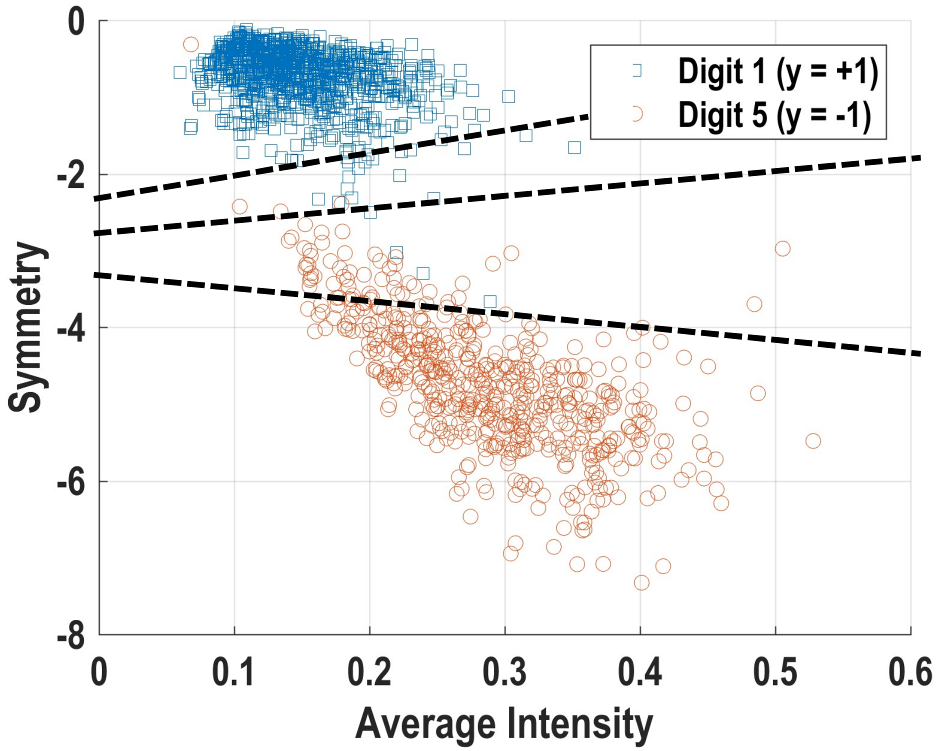
Machine Learning Task: Classify images as 1 or 5. This is an instance of a **binary classification task**.

Linear Classification

Problem Setup: Given data ($\boldsymbol{x}_i$) and classification labels (y_i), find the best model that **separates/classifies current data** **and** **predicts future data**

Example: Develop a model to classify between 1s and 5s.

Here, the independent variables (training data) are average intensity and symmetry of the digit images ($\boldsymbol{x}_i$) and the dependent variable (label) is 1 or 5 (y_i).

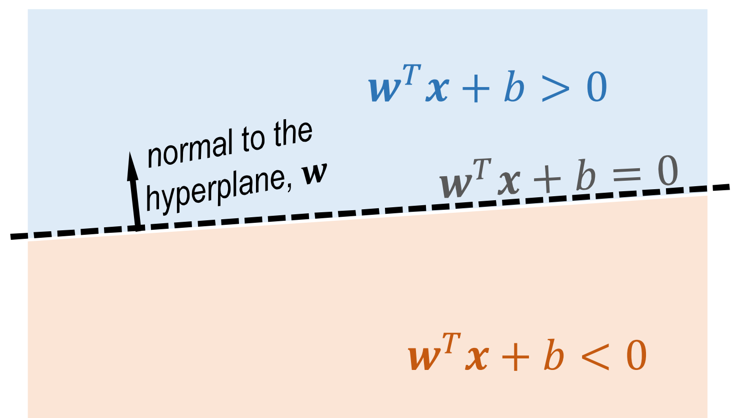


Our hypothesis class is the space of all **linear functions**,

$$y = f(x) = \boldsymbol{w}^T\boldsymbol{x} + b$$

in this 1vs5 digit classification task, the training examples are two-dimensional (intensity, symmetry), that is $\boldsymbol{x} \in \mathbb{R}^2$

In n dimensions, a **hyperplane** is a solution to the equation, $\boldsymbol{w}^T\boldsymbol{x} + b = 0$ with $\boldsymbol{w} \in \mathbb{R}^n$ and $b \in \mathbb{R}$. Hyperplanes **divide** $\mathbb{R}^n$ **into two distinct sets** of points (called open half-spaces)

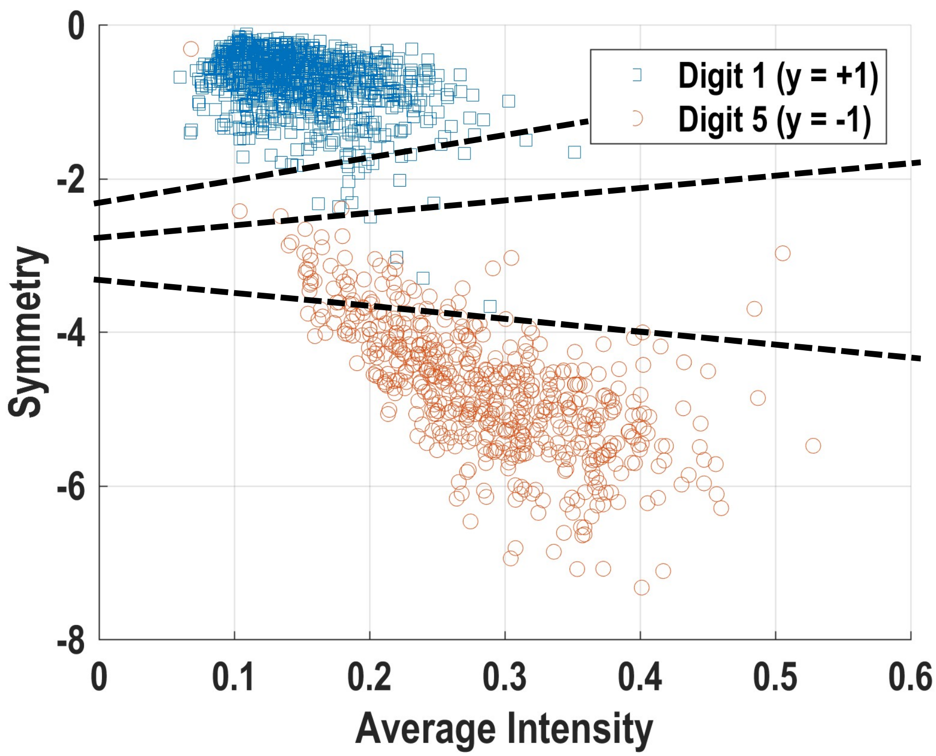


*For a classification problem, the labels are **not continuous**, but **nominal**. Here, denote the labels for **Digit 1 as y = +1** and the labels for **Digit 5 as y = -1**.*

Linear Classification

Problem Setup: Given data ($\boldsymbol{x}_i$) and classification labels (y_i), find the best model that **separates/classifies current data <u>and</u> predicts future data**

Problem: Given n training examples $(\boldsymbol{x}_i, y_i)$, $i = 1, \ldots, n$, where $y_i = \{+1, -1\}$, find the best model $(\boldsymbol{w}, b)$



*In linear regression, we measure fit using the squared loss over the error, that is, we use a **squared loss function**,*

$$L(f(x_i), y_i) = \frac{1}{2}\left(y_i - (w \cdot x_i + b)\right)^2$$

Is this still a good loss function?

*Count the **number of misclassifications**:*

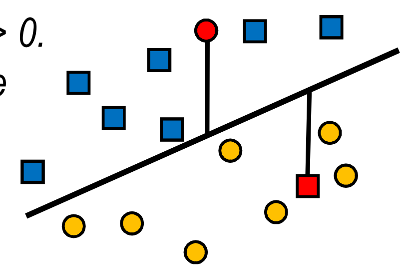
$$L(f(x_i), y_i) = \frac{1}{2}\left|y_i - \operatorname{sign}(w \cdot x_i + b)\right|$$

Loss function is not differentiable, difficult to optimize

*Penalize each **misclassification by the size of the violation**, using the **modified hinge loss***

$$L(f(x_i), y_i) = \max\{0, -y_i \cdot (w \cdot x_i + b)\}$$

*Only misclassified points will have a loss > 0. Correctly classified points will always have loss = 0. **Why?***



Linear Classification

Problem Setup: Given data ($\boldsymbol{x}_i$) and classification labels (y_i), find the best model that **separates/classifies current data and predicts future data**

Problem: Given n training examples $(\boldsymbol{x}_i, y_i)$, $i = 1, \dots, n$, where $y_i = \{+1, -1\}$, find the best model $(\boldsymbol{w}, b)$ by solving

$$\underset{w,b}{\text{minimize}} \ \sum_{i=1}^{n} \max\{0, -y_i \cdot (\boldsymbol{w}^T \boldsymbol{x}_i + b)\}$$

This is an **(unconstrained) optimization problem** in the variables $(\boldsymbol{w}, b)$. The **optimal solution** will be our model.

Solution Approach: Solve using optimization techniques, e.g., **gradient descent**. However, the loss function is convex, but **not differentiable everywhere**!

Linear Classification

Problem Setup: Given data ($\boldsymbol{x}_i$) and classification labels (y_i), find the best model that **separates/classifies current data <u>and</u> predicts future data**

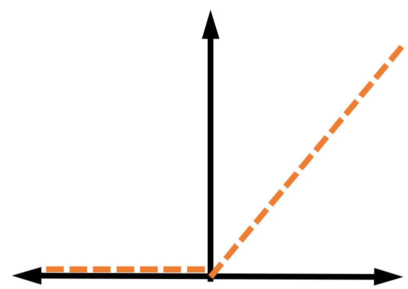
Problem: Given n training examples $(\boldsymbol{x}_i, y_i)$, $i = 1, \ldots, n$, where $y_i = \{+1, -1\}$, find the best model $(\boldsymbol{w}, b)$ by solving

$$\underset{w,b}{\text{minimize}} \ \sum_{i=1}^{n} \max\{0, -y_i \cdot (\boldsymbol{w}^T \boldsymbol{x}_i + b)\}$$

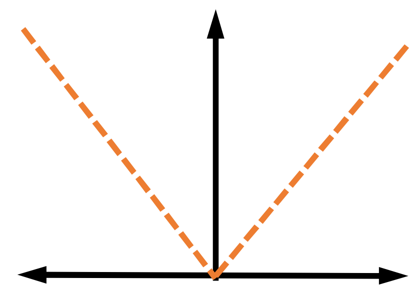
This is an **(unconstrained) optimization problem** in the variables $(\boldsymbol{w}, b)$. The **optimal solution** will be our model.

Solution Approach: Solve using optimization techniques, e.g., **gradient descent**. However, the loss function is convex, but **not differentiable everywhere**!

Piecewise continuous functions such as $\max(0, x)$ and $|x|$ are not differentiable everywhere (in this case, at $x = 0$).



$$\max(0, x) = \begin{cases} x & \text{if } x > 0 \\ 0 & \text{if } x = 0 \\ 0 & \text{if } x < 0 \end{cases}$$



$$|x| = \begin{cases} x & \text{if } x > 0 \\ 0 & \text{if } x = 0 \\ -x & \text{if } x < 0 \end{cases}$$

Linear Classification

Problem Setup: Given data ($\boldsymbol{x}_i$) and classification labels (y_i), find the best model that **separates/classifies current data** **and** **predicts future data**

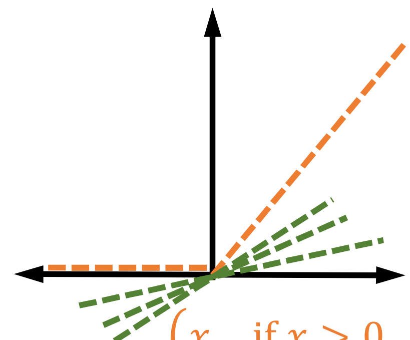
Problem: Given n training examples $(\boldsymbol{x}_i, y_i)$, $i = 1, \dots, n$, where $y_i = \{+1, -1\}$, find the best model $(\boldsymbol{w}, b)$ by solving

$$\underset{w,b}{\text{minimize}} \ \sum_{i=1}^{n} \max\{0, -y_i \cdot (\boldsymbol{w}^T \boldsymbol{x}_i + b)\}$$

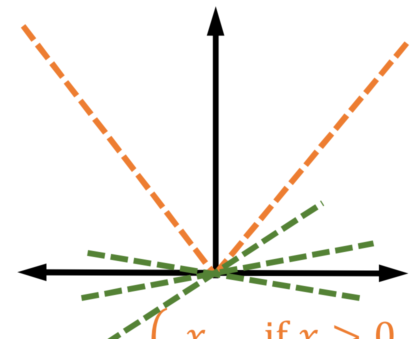
This is an **(unconstrained) optimization problem** in the variables $(\boldsymbol{w}, b)$. The **optimal solution** will be our model.

Solution Approach: Solve using optimization techniques, e.g., **gradient descent**. However, the loss function is convex, but **not differentiable everywhere**!

Piecewise continuous functions such as $\max(0, x)$ and $|x|$ are not differentiable everywhere. We compute the **sub-gradient** instead.



$$\max(0, x) = \begin{cases} x & \text{if } x > 0 \\ 0 & \text{if } x = 0 \\ 0 & \text{if } x < 0 \end{cases}$$



$$|x| = \begin{cases} x & \text{if } x > 0 \\ 0 & \text{if } x = 0 \\ -x & \text{if } x < 0 \end{cases}$$

$$\frac{\partial}{\partial x} \max(0, x) = \begin{cases} 1 & \text{if } x > 0 \\ [0, 1] & \text{if } x = 0 \\ 0 & \text{if } x < 0 \end{cases} \qquad \frac{\partial}{\partial x} |x| = \begin{cases} 1 & \text{if } x > 0 \\ [-1, 1] & \text{if } x = 0 \\ -1 & \text{if } x < 0 \end{cases}$$

For a convex function $f(x)$, a **sub-gradient** at a point x_0 is **any tangent line or plane** through the point x_0 that underestimates (supports) the function **everywhere**.

Perceptron

Problem Setup: Given data ($\boldsymbol{x}_i$) and classification labels (y_i), find the best model that **separates/classifies current data and predicts future data**

Problem: Given n training examples $(\boldsymbol{x}_i, y_i)$, $i = 1, \dots, n$, where $y_i = \{+1, -1\}$, find the best model $(\boldsymbol{w}, b)$ by solving

$$\underset{w,b}{\text{minimize}} \sum_{i=1}^{n} \max\{0, -y_i \cdot (\boldsymbol{w}^T \boldsymbol{x}_i + b)\}$$

Solution Approach 1: Solve using optimization techniques, e.g., **sub-gradient descent** (compare with gradient descent used for regression)

Initalize: $w = w_0, b = b_0, t = 0$
Iterate until convergence
Compute updates:
$$w_{t+1} = w_t - \eta_t \nabla_w L(f(\boldsymbol{x}), y)$$
$$b_{t+1} = b_t - \eta_t \nabla_b L(f(\boldsymbol{x}), y)$$
Check for convergence
Continue to next iteration: $t = t + 1$

$$\nabla_w L(f(\boldsymbol{x}_i), y_i) = \sum_{i:-y_i f(\boldsymbol{x}_i)>0} -y_i \cdot \boldsymbol{x}_i$$

$$\nabla_b L(f(\boldsymbol{x}_i), y_i) = \sum_{i:-y_i f(\boldsymbol{x}_i)>0} -y_i$$

*gradient only depends on the **misclassified examples***

Perceptron

Problem Setup: Given data ($\boldsymbol{x}_i$) and classification labels (y_i), find the best model that **separates/classifies current data and predicts future data**

Problem: Given n training examples $(\boldsymbol{x}_i, y_i)$, $i = 1, \dots, n$, where $y_i = \{+1, -1\}$, find the best model $(\boldsymbol{w}, b)$ by solving

$$\underset{w,b}{\text{minimize}} \ \sum_{i=1}^{n} \max\{0, -y_i \cdot (\boldsymbol{w}^T \boldsymbol{x}_i + b)\}$$

***approximate** the gradient **by sampling** a few examples uniformly at random and averaging; in the extreme case, select only a single example*

*stochastic gradient descent **converges** under mild assumptions on the step size (it should decrease*

Solution Approach 2: To make training more practical, **stochastic sub-gradient descent** is used instead of sub-gradient descent

Initalize: $w = w_0, b = b_0$
for $i = 1, \dots, n$
  Select a random training example, $(\boldsymbol{x}_i, y_i)$
  Compute updates **if $(\boldsymbol{x}_i, y_i)$ misclassified**

$$w_{i+1} = w_i - \eta_i \nabla_w L(f(\boldsymbol{x}_i), y_i)$$
$$b_{i+1} = b_i - \eta_i \nabla_b L(f(\boldsymbol{x}_i), y_i)$$

  Else

$$w_{i+1} = w_i$$
$$b_{i+1} = b_i$$

$$\nabla_w L(f(\boldsymbol{x}_i), y_i) = -y_i \boldsymbol{x}_i$$

$$\nabla_b L(f(\boldsymbol{x}_i), y_i) = -y_i$$

Perceptron

Problem Setup: Given data ($\boldsymbol{x}_i$) and classification labels (y_i), find the best model that **separates/classifies current data and predicts future data**

Problem: Given n training examples $(\boldsymbol{x}_i, y_i)$, $i = 1, \ldots, n$, where $y_i = \{+1, -1\}$, find the best model $(\boldsymbol{w}, b)$ by solving

$$\underset{w,b}{\text{minimize}} \ \sum_{i=1}^{n} \max\{0, -y_i \cdot (\boldsymbol{w}^T \boldsymbol{x}_i + b)\}$$

Solution Approach 2: To make training more practical, **stochastic sub-gradient descent** is used instead of sub-gradient descent

Initalize: $w = w_0, b = b_0$
for $i = 1, \ldots, n$
Select a random training example, $(\boldsymbol{x}_i, y_i)$
Compute updates **if $(\boldsymbol{x}_i, y_i)$ misclassified**

$$w_{i+1} = w_i - \eta_i \nabla_w L(f(\boldsymbol{x}_i), y_i)$$
$$b_{i+1} = b_i - \eta_i \nabla_b L(f(\boldsymbol{x}_i), y_i)$$

Else

$$w_{i+1} = w_i$$
$$b_{i+1} = b_i$$

$$\nabla_w L(f(\boldsymbol{x}_i), y_i) = -y_i \boldsymbol{x}_i$$

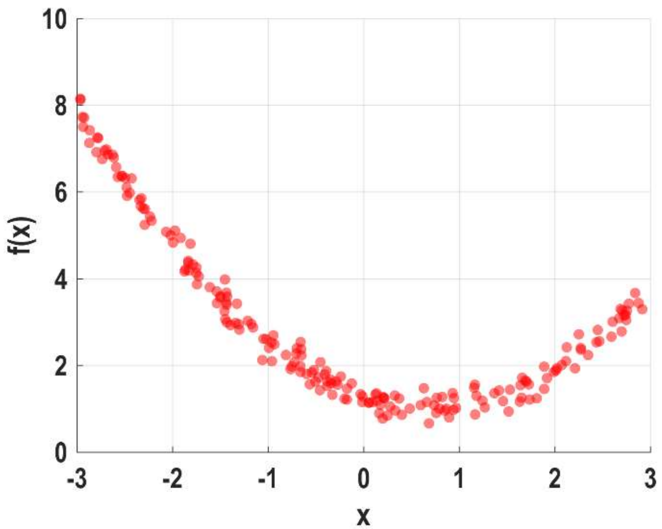
$$\nabla_b L(f(\boldsymbol{x}_i), y_i) = -y_i$$

Drawbacks:

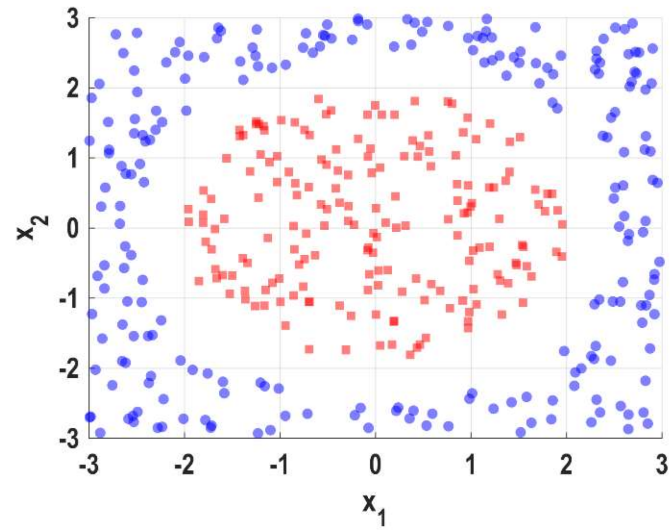
- No convergence guarantees if the observations are **not** linearly separable
- **Can overfit**: there can be a number of perfect classifiers, but the perceptron algorithm doesn't have any mechanism for choosing between them

Limitations of Linear Hypotheses

Regression: Non-linear regression functions



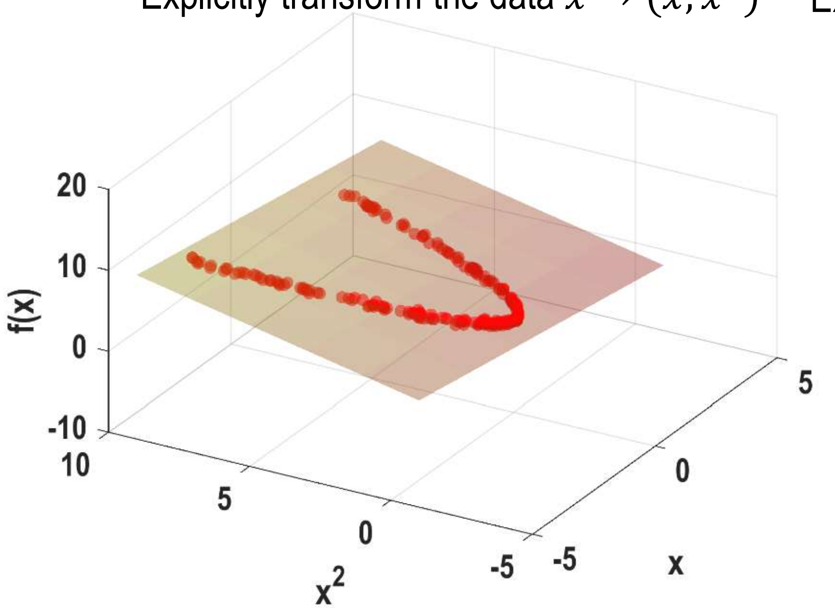
Classification: Linearly inseparable classes



Transforming the data into a **higher-dimensional** space makes the problem linear in that space at expense of adding more features.

Coming up with such transformations is not easy.

Explicitly transform the data $x \rightarrow (x, x^2)$



Explicitly transform the data $(x_1, x_2) \rightarrow (x_1^2, x_2^2, \sqrt{2}x_1x_2)$