“a set of methods that can automatically detect patterns in data, and then use the uncovered patterns to **predict** future data, or perform other kinds of decision making” – Murphy 2012
Goal: Learn to classify examples
E.g.,
• Images (face recognition)
• Emails (spam filtering)
• Biological samples (tumor classification)
Motivation:
• The analysis of long-term ECG recordings can help physicians understand a patient’s health.
• Labeling heartbeats can be an important step in this task; provides a level of abstraction.
• >100 000 beats in just 24 hours, so needs to be automated.

ECG Recording
Example: Heartbeat Classification

Labeled Training Data
Example: Heartbeat Classification

Task:
Interpret data & make classifications

Labeled Training Data

New Patient Recording
Example: Heartbeat Classification

Labeled Training Data

Task: Interpret data & make classifications

New Patient Recording

Heartbeat Classification
Labeled Feature Vector Representation

Example: Heartbeat Classification

Labeled Feature Vector Representation

$d$-dimensional feature vector

binary label

\[ \{ \bar{x}^{(1)}, -1 \} \]

\[ \rightarrow \]

\[ \{ \bar{x}^{(2)}, +1 \} \]
Example: Heartbeat Classification

Labeled Feature Vector Representation

\[ \vec{x}^{(1)}, -1 \]
\[ \vec{x}^{(2)}, +1 \]

Possible information to encode as features:
- amplitude
- timing info/context
- frequency information
Example: Heartbeat Classification

Labeled Feature Vector Representation

\[ \{ \bar{x}^{(1)}, -1 \} \]

\[ \{ \bar{x}^{(2)}, +1 \} \]
Labeled Feature Vector Representation

\[ \{ \mathbf{x}^{(1)}, -1 \} \]

\[ \{ \mathbf{x}^{(2)}, +1 \} \]

Learning goal: identify the decision boundary that separates the positive from negative examples.

Example: Heartbeat Classification
\[ S_n = \{ \bar{x}^{(i)}, y^{(i)} \}_{i=1}^{n} \quad \bar{x} \in \mathcal{X} \quad y \in \mathcal{Y} \]

**Goal:** learn a mapping from \( \mathcal{X} \rightarrow \mathcal{Y} \) that generalizes to yet unseen data.
Lecture 1: Classification as an ML problem

- Feature vector \( \mathbf{x} = [x_1, x_2, x_3, \ldots, x_d]^T \) \( \in \mathbb{R}^d \)
- Labels \( y \in \{-1, +1\} \) (binary)
- Training set of examples \( \mathcal{S}_n = \{ (\mathbf{x}^{(i)}, y^{(i)}) \}_{i=1}^n \)
- Classifier \( h: \mathbb{R}^d \to \{-1, +1\} \)

**Goal:** Select the best \( h \) from a set of possible classifiers \( \mathcal{H} \) that would have the best chance of correctly classifying new examples.

The problem of selecting \( h \) from \( \mathcal{H} \) is solved by a learning algorithm, typically an optimization problem with respect to \( \mathcal{S}_n \).

**Example:** ECG data sampled at 360 Hz

1 sec

\[ \mathbf{x} = [x_1, x_2, x_3, \ldots, x_{360}] \] \( , \) \( x_k \in \{1, \ldots, 2048\} \)

- Training examples \( n = 50 \) \( \{ (\mathbf{x}^{(i)}, y^{(i)}) \}_{i=1}^{50} \) \( y \in \{-1, +1\} \)
- \( x_3 \) is different in every sample.

Given the small number of training examples, we can trivially come up with a solution that maps each beat to the correct label just based on a look up table that uses \( x_3 \).

But is this a good classifier? No, it overfits.

**Generalization** works well on unseen examples.

**Problem:** Too many choices in \( \mathcal{H} \), so many that we may end up choosing a classifier that does well on the specific training set but fails applied to new data.

**Solution:** constrain \( \mathcal{H} \), but it can’t be too small either or we may end up unable to classify \( \mathcal{S}_n \) (underfit).

**Model selection:** Finding the right balance.
**Linear Classification:**

\[ H : \text{ thresholded linear mappings from feature vectors to labels} \]

\[ h(\mathbf{x}; \mathbf{\Theta}) = \begin{cases} +1 & \mathbf{\Theta} \cdot \mathbf{x} > 0 \text{ where } \mathbf{\Theta} \in \mathbb{R}^d \\ -1 & \mathbf{\Theta} \cdot \mathbf{x} < 0 \end{cases} \]

\[ \mathbf{\Theta} \cdot \mathbf{x} = \Theta_1 x_1 + \Theta_2 x_2 + \Theta_3 x_3 + \ldots + \Theta_d x_d = \text{a linear combination of the input features} \]

Different \( \mathbf{\Theta} \)'s produce different labelings for some \( \mathbf{x} \)

**Geometrically**

Let \( d = 2 \)

\[ \mathbf{\Theta} \cdot \mathbf{x} = \| \mathbf{x} \| \| \mathbf{\Theta} \| \cos \alpha \]

Recall: \( \| \mathbf{x} \| = \sqrt{x_1^2 + x_2^2 + \ldots + x_d^2} \) "L2-norm" always \( \geq 0 \)

What if a point lies on the decision boundary?

\[ \mathbf{x} \cdot \mathbf{\Theta} = \| \mathbf{x} \| \| \mathbf{\Theta} \| \cos 90^\circ = 0 \]

In 2-dimensions \( \Theta_1 x_1 + \Theta_2 x_2 = 0 \Rightarrow x_2 = -\frac{\Theta_1}{\Theta_2} x_1 \) \( \mathbf{\Theta} \) defines the slope of the decision boundary.
How do we select $\overline{\theta}$?

Approach: find $\overline{\theta}$ that works well on the training data

$S_n = \{x^{(i)}, y^{(i)}\}$

Minimize Training Error: fraction of examples for which the classifier predicts the wrong label

$$
E(\overline{\theta}) = \frac{1}{n} \sum_{i=1}^{n} \left[ y^{(i)} \neq h(x^{(i)}, \overline{\theta}) \right]
= \frac{1}{n} \sum_{i=1}^{n} \left[ \left( \overline{\theta} \cdot x^{(i)} \right) < 0 \right]
= \frac{1}{n} \sum_{i=1}^{n} \text{Loss} \left( y^{(i)} \left( \overline{\theta} \cdot x^{(i)} \right) \right)
$$

"zero-one loss"

Unfortunately, a reasonable algorithm for finding $\overline{\theta}$ that minimizes training error is not easy to solve in general. So we will consider algorithms that approximately minimize training error.

Let's instead consider a loss function that treats these two errors differently.
Approximate training error with hinge loss

\[ R_n(\theta) = \frac{1}{n} \sum_{i=1}^{n} \max \{ 1 - y^{(i)} \theta \cdot x^{(i)}, 0 \} \]

Idea: by minimizing "empirical risk" we can obtain a classifier that generalizes well.

We can easily minimize \( R_n(\theta) \) since it is a convex function, convexity guarantees there's a simple algorithm that will find the minimum.

Recall: a convex function \( f(x) \) is any function that

\[ f(\lambda x + (1-\lambda) x') \leq \lambda f(x) + (1-\lambda) f(x') \quad \text{for any } x, x' \text{ and } \lambda \in [0, 1] \]

- every chord lies above the function

\[ \text{e.g.} \]

\[ f(\lambda x + (1-\lambda) x') \leq \lambda f(x) + (1-\lambda) f(x') \]

\[ \text{characteristic bowl shape} \]

A simple algorithm for finding the minimum?

gradient descent, stochastic gradient descent
Gradient Descent

We will use gradient descent to minimize $R_n(\theta)$ with hinge loss.

Gradient points in the direction $\nabla R_n(\theta)$ increases

$$\nabla R_n(\theta) = \left[ \frac{\partial R_n(\theta)}{\partial \theta_1}, \frac{\partial R_n(\theta)}{\partial \theta_2}, \frac{\partial R_n(\theta)}{\partial \theta_3}, \ldots, \frac{\partial R_n(\theta)}{\partial \theta_k} \right]$$

Idea: take a small step in the opposite direction

$$\tilde{\theta}^{(k+1)} = \tilde{\theta}^{(k)} - \eta \nabla R_n(\theta) \quad \text{at } \tilde{\theta} = \tilde{\theta}^{(k)}$$

Step size or learning rate

Recall $R_n(\theta) = \frac{1}{n} \sum_{i=1}^{n} \max \{1 - y^{(i)} \theta \cdot x^{(i)}, 0\}$

Summation involved in calculating gradient makes gradient descent slow

Stochastic Gradient Descent

Idea: update $\tilde{\theta}$ based on a small batch or a single point

$$\tilde{\theta}^{(k)} = \tilde{\theta}^{(k-1)} \quad k=0$$

while convergence criteria not met:

randomly select $i \in \{1, \ldots, n\}$

$$\tilde{\theta}^{(k+1)} = \tilde{\theta}^{(k)} - \eta \nabla \text{loss_{\theta}} \left( y^{(i)} \tilde{\theta} \cdot x^{(i)} \right) \quad \text{at } \tilde{\theta} = \tilde{\theta}^{(k)}$$

Technicality: $R_n(\theta)$ with hinge loss is piecewise linear

what do we do?

when differentiable $\rightarrow$ no problem

when subdifferentiable $\rightarrow$ choose any gradient around the kink
Note: if $y^{(i)} \bar{\theta} \cdot \bar{x}^{(i)} > 1 \rightarrow$ loss is zero & no update

if $y^{(i)} \bar{\theta} \cdot \bar{x}^{(i)} \leq 1$

then $\nabla_\bar{\theta} \text{Loss} (y^{(i)} \bar{\theta} \cdot \bar{x}^{(i)})$

$= \nabla_\bar{\theta} (1 - y^{(i)} (\bar{\theta} \cdot \bar{x}^{(i)}))$

$= - y^{(i)} \bar{x}^{(i)}$

update rule: if $y^{(i)} \bar{\theta} \cdot \bar{x}^{(i)} \leq 1$

$\bar{\theta}^{(\kappa + 1)} = \bar{\theta}^{(\kappa)} + \eta y^{(i)} \bar{x}^{(i)}$

Notes:

* (stochastic) gradient descent is a general algorithm that can be applied to non-convex functions

* SGD often gets closer to the solution more quickly than GD

* with appropriate learning rate if $p_n(\bar{\theta})$ is convex will almost surely converge to minimum