Summary: Supervised Machine Learning

Learning from *labeled* examples

- each example is a vector of features $\mathbf{x}$ and a target/label $\mathbf{y}$
    - n denotes length of vector $\mathbf{x}$
    - superscript to distinguish between examples $\mathbf{x^{(i)}}, \mathbf{y^{(i)}}$

Prediction: creating a model $\boldsymbol{h}$

- Given training example $\mathbf{x^{(i)}}$, we construct a function h to predict its label
$$\hat{\mathbf{y}}^{\mathbf{(i)}} = h(\mathbf{x^{(i)}}; \Theta)$$
- We use a "hat" to denote predictions: $\hat{\mathbf{y}}^{\mathbf{(i)}}$
- The behavior h is determined by parameters Θ

Fitting a model $\boldsymbol{h}$: Finding optimal values for $\boldsymbol{\Theta}$

The collection of examples used for fitting (training) a model is called the *training set*:

$$\langle \mathbf{X}, \mathbf{y} \rangle = [\mathbf{x}^{(\mathbf{i})}, \mathbf{y}^{(\mathbf{i})} | 1 \leq i \leq m]$$

where m is the size of training set and each $\mathbf{x}^{(\mathbf{i})}$ is a feature vector of length n.

$$\mathbf{X} = \begin{pmatrix} (\mathbf{x}^{(1)})^T \\ (\mathbf{x}^{(2)})^T \\ \vdots \\ (\mathbf{x}^{(m)})^T \end{pmatrix} = \begin{pmatrix} \mathbf{x}_1^{(1)} \dots \mathbf{x}_n^{(1)} \\ \mathbf{x}_1^{(2)} \dots \mathbf{x}_n^{(2)} \\ \vdots \\ \mathbf{x}_1^{(m)} \dots \mathbf{x}_n^{(m)} \end{pmatrix}$$

Fitting a model: Loss/Cost, Utility

Ideal: for each i in training dataset:

- prediction $\mathbf{\hat{y}^{(i)}} = \mathbf{h}(\mathbf{x^{(i)}}; \Theta)$ exactly equal to target $\mathbf{y^{(i)}}$

$$\mathbf{\hat{y}^{(i)}} = \mathbf{y}$$

Reality: prediction often has some "error"

- error measured by a *distance* function: smaller (closer to target) is better
- Call the distance between $\mathbf{\hat{y}^{(i)}}, \mathbf{y^{(i)}}$ the *Loss* (or *Cost*) for example i:

Per-example loss

$$\mathcal{L}_\Theta^{(\mathbf{i})} = L(\ h(\mathbf{x^{(i)}}; \Theta), \mathbf{y^{(i)}}\ ) = L(\mathbf{\hat{y}^{(i)}}, \mathbf{y^{(i)}})$$

where $L(a, b)$ is a function that is 0 when $a = b$ and increasing as a increasingly differs from b.

Two common forms of L are Mean Squared Error (for Regression) and Cross Entropy Loss (for classification).

Optimal Θ

The Loss for the entire training set is simply the average (across examples) of the Loss for the example

$$\mathcal{L}_\Theta = \frac{1}{m} \sum_{i=1}^{m} \mathcal{L}_\Theta^{(\mathbf{i})}$$

The best (optimal) Θ is the one that minimizes the Average (across training examples) Loss

$$\Theta^* = \underset{\Theta}{\operatorname{argmin}} \, \mathcal{L}_\Theta$$

Pattern matching

The "dot product" (special case of inner product) is one function that often appears in template matching

- It measures the similarity of two vectors

$$\mathbf{v} \cdot \mathbf{v}' = \sum_{i=1}^{n} \mathbf{v}_i \mathbf{v}'_i$$

- As a similarity measure (rather than as a distance) high dot product means "more similar".

In Machine Learning it is *often* (but not always) the case

- we match a feature vector $\mathbf{x}^{(\mathbf{i})}$
- to all/some of the parameters Θ

KNN: a simple model for the Classification task

Parameters Θ **are** the training examples

- training examples are discarded after training/fitting
$$\langle \Theta_{\mathbf{x}}, \Theta_{\mathbf{y}} \rangle = \langle \mathbf{X}, \mathbf{y} \rangle$$

KNN

- measures *similarity* out of sample feature vector $\mathbf{x}$ against the feature vector of each example i
- **dot product** matches example against a row of $\Theta_{\mathbf{x}}$
$$\text{similarity}(\mathbf{x}, \Theta_{\mathbf{x}}^{(\mathbf{i})}) = \mathbf{x} \cdot \Theta_{\mathbf{x}}^{(\mathbf{i})} = \mathbf{x} \cdot \mathbf{X}^{(\mathbf{i})}$$

KNN uses *lots* of parameters

$$
\begin{array}{lll}
\|\Theta\| & = & \|\Theta_{\mathbf{x}}\| & + & \|\Theta_{\mathbf{y}}\| \\
 & & m * n & + & m
\end{array}
$$

Perhaps *exact matching* against a large set of examples is not necessary ?

- Digit classification
    - A "generic" pattern for each digit
        - pattern for a "1" is a vertical column of dark pixels in the center
        - pattern for a "8" is two "donut holes" stacked atop one another, with a "pinched waist"
    - Parameter size: $10 * n$
        - 10 patterns * n pixel intensities per pattern

We will learn *other* models for Classification that essentially learn these *per-digit* patterns

In [4]:

```
print("Done")
```

Done