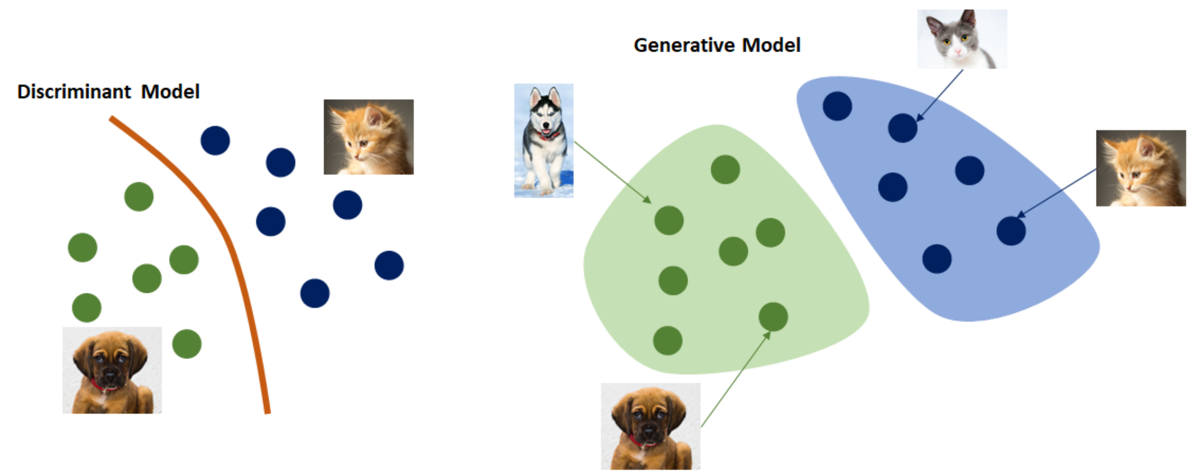
EE 046746 - Technion - Computer Vision

Tutorial 02 - Probabilistic Discriminative Learning

Elias Nehme



- Image source

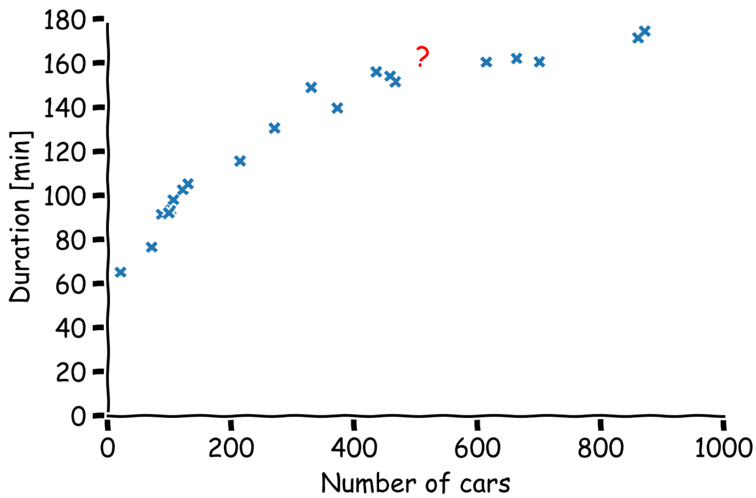
Agenda

Machine Learning Overview

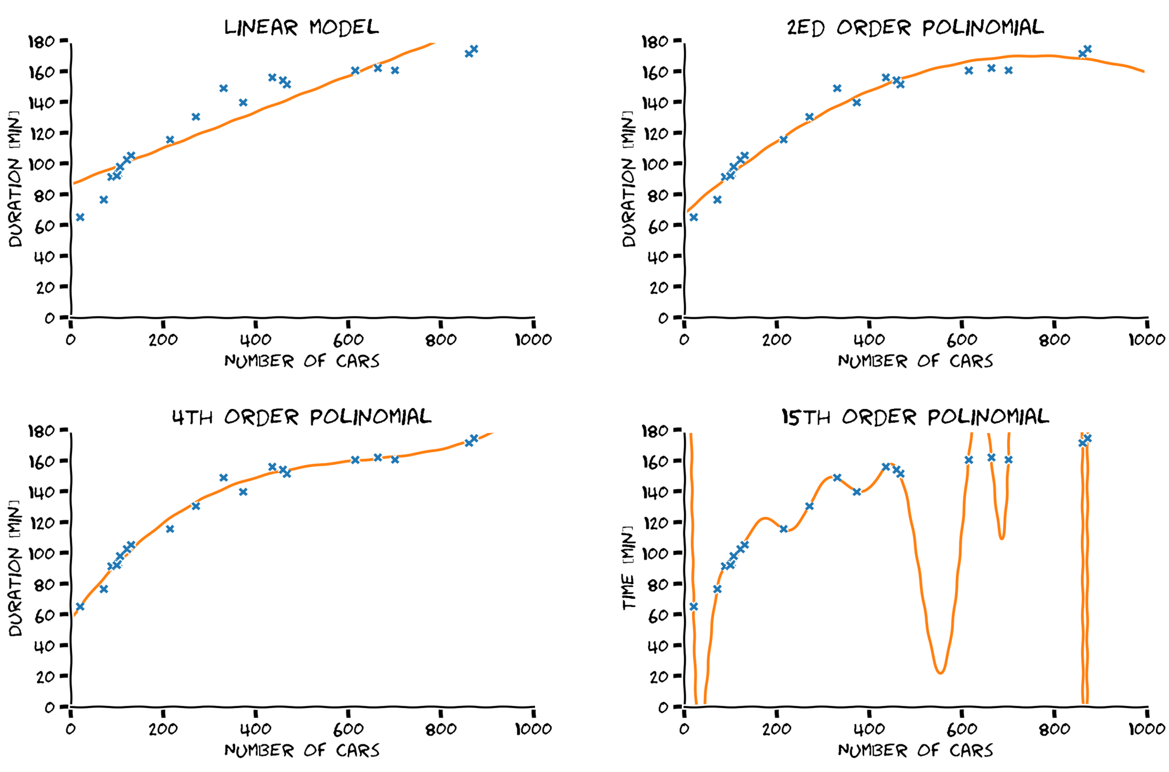
What is Machine Learning?

- Wikipedia definition: "the study of computer algorithms that can improve automatically through experience and by the use of data"
- What does that really mean? Let's see an example.

- **Example 1**: Consider the task of predicting the duration of travelling the road given the current amount of cars.



- Naturally one can choose multiple models to predict the duration.
- For example, we can limit ourselves to parametric models (eg polynomials)
- Which of the following models should we choose?

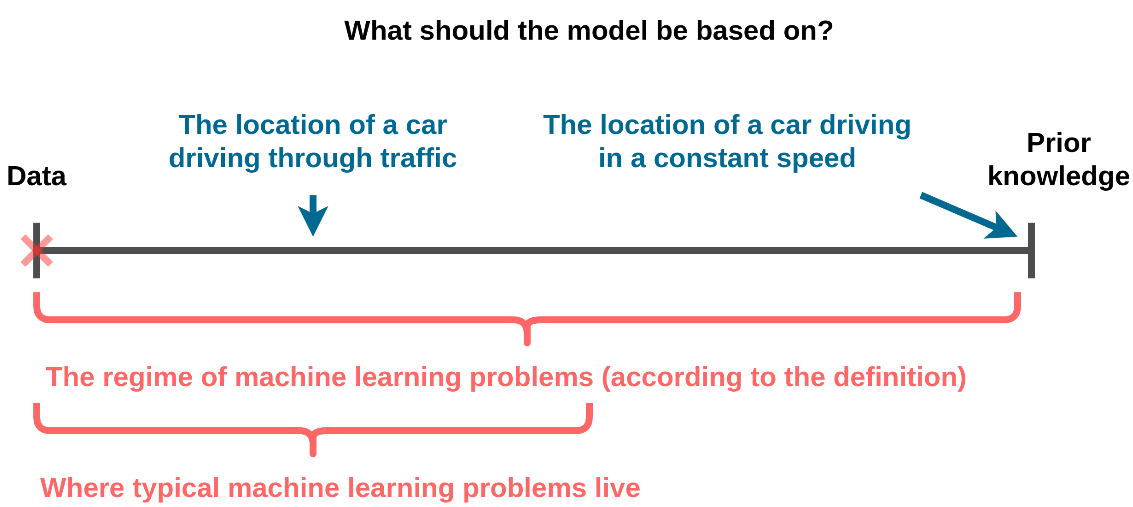


- Higher polynomial degree ensures better fit to data.

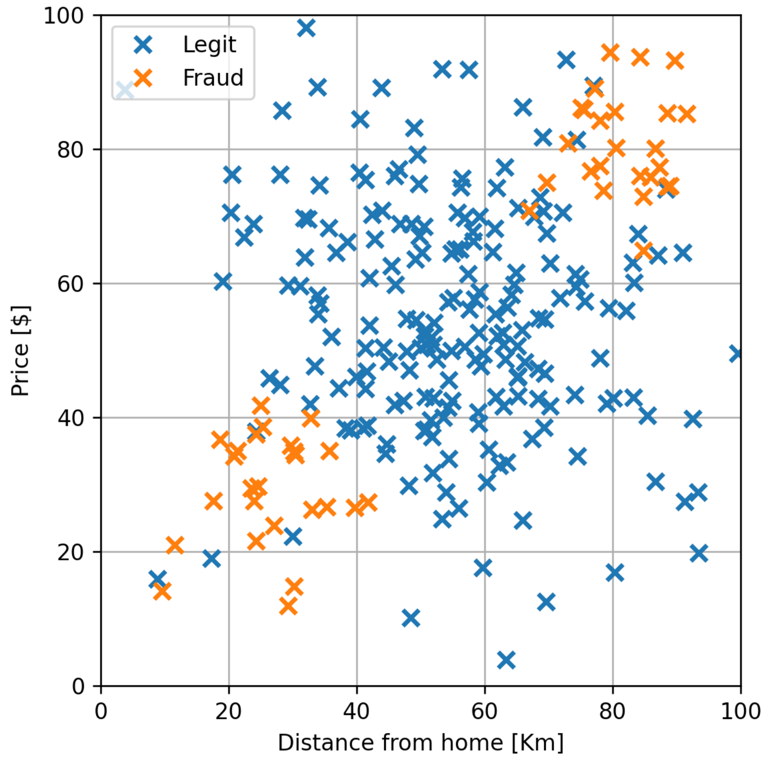
- However, better fit to data does not necessarily ensure better "generalization" error.
- Assuming we have prior knowledge the function is smooth we might choose a 4^{th} order polynomial.

Data vs Prior Knowledge

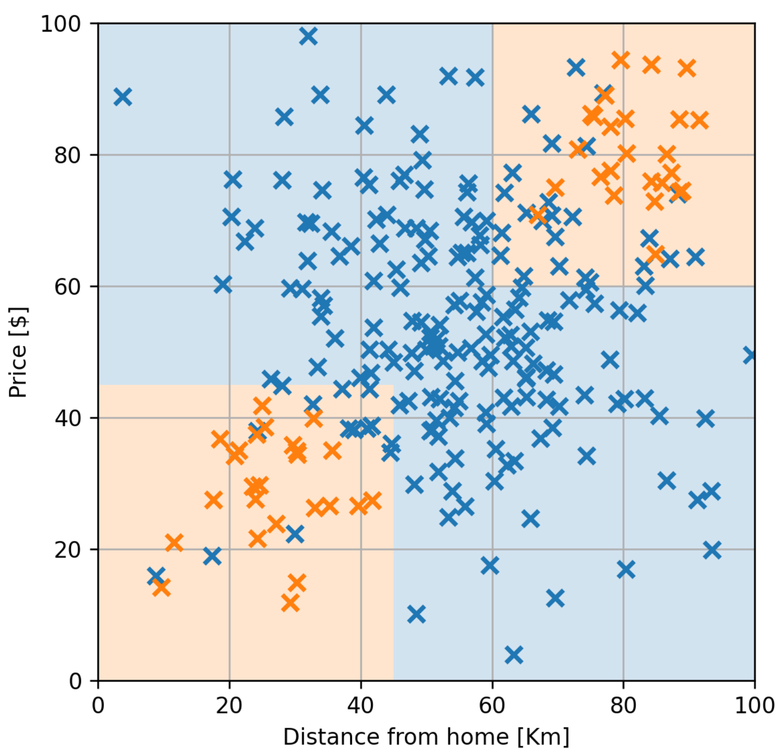
- If we have a solid understanding of the system (geometry, physics, etc.), we rely mainly on prior knowledge.
- However, if we do not have such understanding, we try to infer the system based of collected data.
- For example, coming back to the example from earlier, if the car is driving at constant speed: duration = distance/speed.



- **Example 2**: Consider the task of classifying a credit card deal being legit or fraud based on some charachteristics like the Price, and the physical distance of the store from the card holder's home.



- We would like to create a predction function that tells us absed on these 2 features whether a credit card deal is legit or fraud.
- For example here a naive implemnetation of such a function.

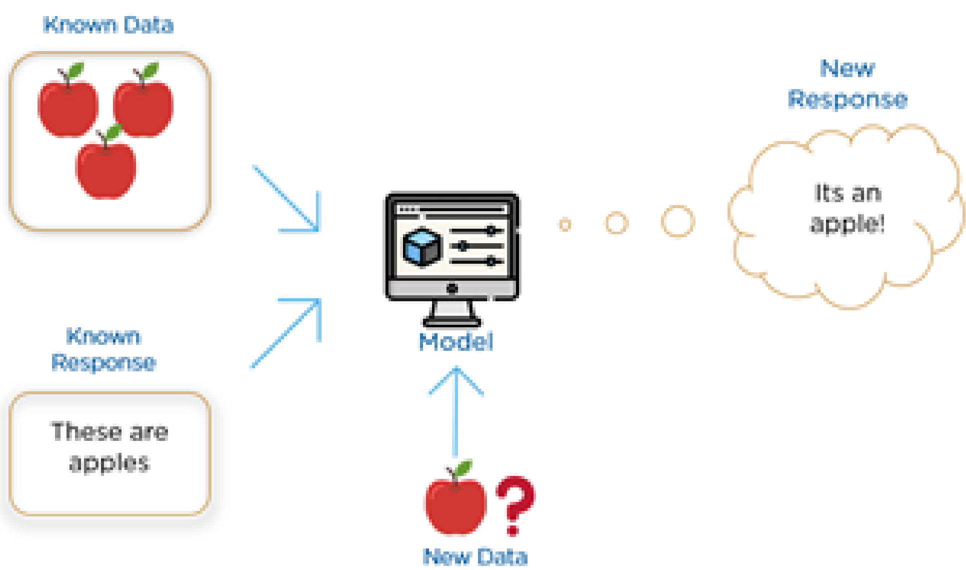


Types of ML Setups

- There are 3 main different types of machine learning setups:
  - **Supervised Learning** (This tutorial)
  - Unsupervised Learning
  - Reinforcement Learning

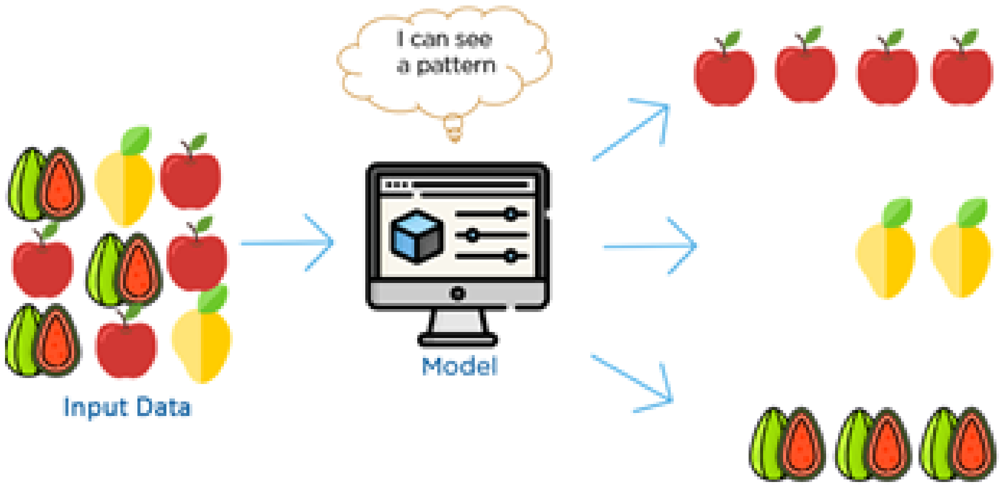
What is Supervised Learning?

- Simplest form of machine learning (easiest to understand). Data is given in the form of examples with labels.
- Algorithm is "trained" to predict the label for each example, while being given feedback for its answers.
- when fully-trained the learning algorithm will be able to observe a new, never-before-seen example and predict a good label for it.
- For example, label emails as spam, classify images as apples, classify cats vs dogs, predict location of tumor in medical images etc.



What is Unsupervised Learning?

- In this case we only have examples (data) with no labels.
- Algorithm is "trained" to understand the properties of the data.
- After training, it can learn to group, cluster, and/or organize the data in a way such that a human (or other intelligent algorithm) can come in and make sense of the newly organized data. (Example in lecture on K-means).



What is Reinforcement Learning?

- Fairly different from the previous two approaches, and beyond the scope of this course.
- Essentially it tries to learn from mistakes with sparse/not-frequent supervision.
- For example, learning a "policy" to navigate a maze with obstacles.

Supervised Learning

- The basis of all other ML problems. Relates to estimation problems from statistical theory.
- The estimation problem is the following: We want to predict the value of unknown random variable (y) based off other known random variables (x).
- Usually in statistics, we assume the distribution of all random variables is known.
- In Supervised learning, we assume we only have a finite sample from this distribution.
- Therefore, our estimator will be built based off the finite sample only.

Supervised Learning - Notation

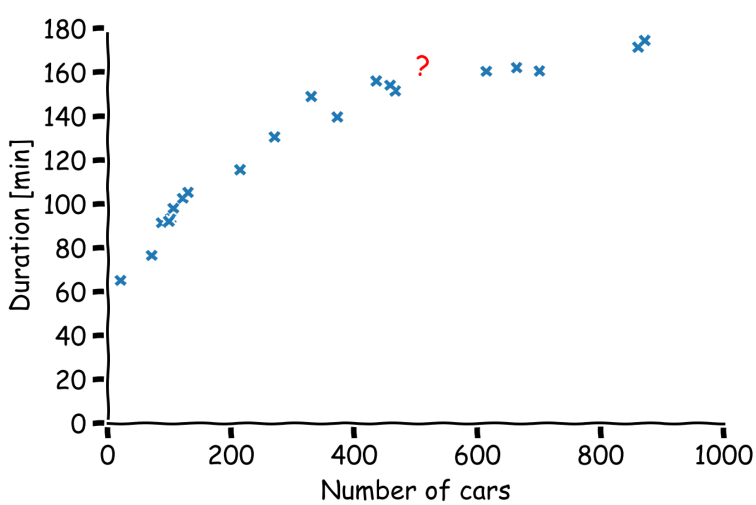
- Labels y - The random variable we are trying to predict
- Observations/Measurements x - the random variables we are basing our predictor on.
- Predictor/Estimator $\hat{y} = h\left(x\right)$ - is the prediction function.
- Dataset $\mathcal{D} = \left\{x^{(i)}, y^{(i)}\right\}_{i=1}^N$ - comprised of N pairs of i.i.d samples from the joint distribution.

Supervised Learning - Problem Types

- Depending on the values that y can take we classify into 2 sub-types:
- Continuous labels y - a regression problem (estimating travel duration example).
- Discrete labels y - a classification problem (predicting legit vs fraud transaction example).

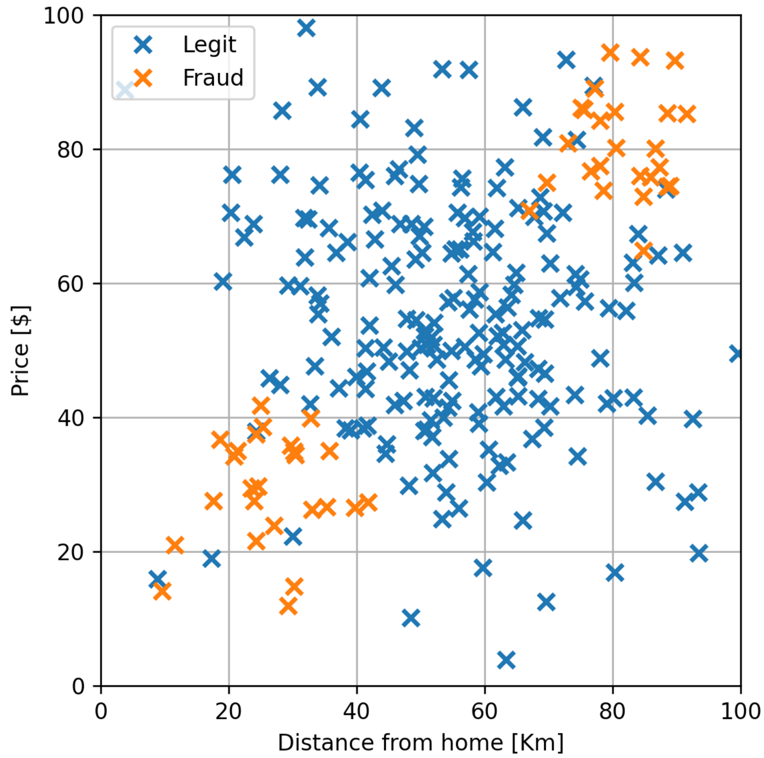
Supervised Learning - Regression

- Regression problem: x = Number of cars, y = Duration time:



Supervised Learning - Classification

- Classification problem: x = $\left[\text{Price}, \text{Distance from home}\right]^T$, $y \in$ {Legit, Fraud}:



Supervised Learning - Optimal Estimator

- Any function mapping $h : x \rightarrow y$ is a valid estimator.
- Optimally we would like our estimator to make no mistakes.
- Due to the labels (y) being a random variable, this is impossible.
- Therefore, we need a way to compare the errors of different estimators to choose the best one.

Supervised Learning - General Solution Paradigm

- Define a mathematical criterion that measures how good an estimator is doing
- Choose a large enough family of models such that one of them will be good enough
- Search all models in the chosen family for the best one.
- (Easier said than done..)

Supervised Learning - Cost Function

- $C(h)$ Gives each estimator a score, lower score = better estimator.
- Optimal estimator $h^*(x)$ is the one with the minimal score:

$$h^* = \underset{h}{argmin}\, C(h)$$

- Usually chosen from a subset of widely used functions.

Supervised Learning - Loss and Risk Functions

- The loss function ℓ gives a score for a single prediction:

$$\ell(h(x), y) = \ell(\hat{y}, y)$$

- The cost is then defined as the expectation over the joint distribution:

$$C(h) = \mathbb{E}\left[\ell(h(x), y)\right]$$

- A familiar synonym for the cost function is the "Risk" function:

$$R(h) \equiv C(h)$$

Supervised Learning - Popular Loss/Risk Functions

- Classification Problems - Misclassification Rate:

$$\ell(\hat{y}, y) = I\left\{\hat{y} \neq y\right\}, \quad R(h) = \mathbb{E}\left[I\left\{h(x) \neq y\right\}\right]$$

- Optimal Classifier given by the Mode:

$$h^*(x) = \underset{y}{argmax}\; p(y|\mathbf{x} = x)$$

- Regression Problems - Mean Squared Error:

$$\ell(\hat{y}, y) = (\hat{y} - y)^2, \quad R(h) = \mathbb{E}\left[(h(x) - y)^2\right]$$

- Optimal Regressor given by the Conditional Mean:

$$h^*(x) = \mathbb{E}\left[y|\mathbf{x} = x\right]$$

Supervised Learning - Unknown Distribution and Empirical Risk

- Problem: We don't actually have access to the posterior distribution $p(y|x)$
- Solution 1 (Generative Models): Estimate the joint distribution based on the dataset $\mathcal{D} = \left\{x^{(i)}, y^{(i)}\right\}_{i=1}^{N}$

- Solution 2 (Discriminative Models): Estimate the expectation empirically based on the dataset $\mathcal{D} = \left\{x^{(i)}, y^{(i)}\right\}_{i=1}^{N}$
- That is, replace the expectation with the empirical mean:

$$R(h) = \mathbb{E}\left[\ell(h(x), y)\right] \approx \hat{R}(h) = \hat{\mathbb{E}}_{\mathcal{D}}\left[\ell(h(x), y)\right] = \frac{1}{N}\sum_{i=1}^{N}\ell(h(x^{(i)}), y^{(i)})$$

- Expected to converge in probability when $N \to \infty$
- Using the empirical risk introduces the problem of overfitting (e.g. polynomial fitting example).

Supervised Learning - Performance Evaluation

- Goal: Learn a model from the given dataset that performs well on **unseen** data.
- For that we would like to evaluate our estimator's performance on data not used in training.
- This is usually achieved by splitting the dataset into two:
  - Training Set - $\mathcal{D}_{\text{train}}$ - used to build our estimator $h^*(x)$
  - Test Set - $\mathcal{D}_{\text{test}}$ - used to evaluate performance on new data
- Performance on the test set can be approximated by the empirical mean:

$$\text{test score} = \frac{1}{N_{\text{test}}}\sum_{x^{(i)}, y^{(i)} \in \mathcal{D}_{\text{test}}}\ell(h^*(x^{(i)}), y^{(i)})$$

Supervised Learning - Parametric Models

- Usually, it is more practical to learn a parametric estimator $h(x;\theta)$ with parameters θ, than searching the entire space of functions for a general $h(x)$.
- This has the benefit of simplifying the optimization and usually reduces overfitting.
- Examples of Parametric Functions:
  - Linear functions: $h(x;\theta) = \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3$
  - Polynomials: $h(x;\theta) = \theta_1 + \theta_2 x_1 + \theta_3 x_1^2 + \theta_4 x_1^3$
  - **Neural networks!** (Next Week)

Supervised Learning - Parametric Models Cont.

- Note that finding the best $h(x;\theta)$ is now broadcasted to finding the optimal set of parameters θ.
- This translates the minimization of the risk functions from earlier to rely on θ:

$$h^* = \underset{h}{argmin}\ R(h) \rightarrow \theta^* = \underset{\theta}{argmin}\ R(h(x;\theta))$$

- For the empirical risk case, this is approximated by:

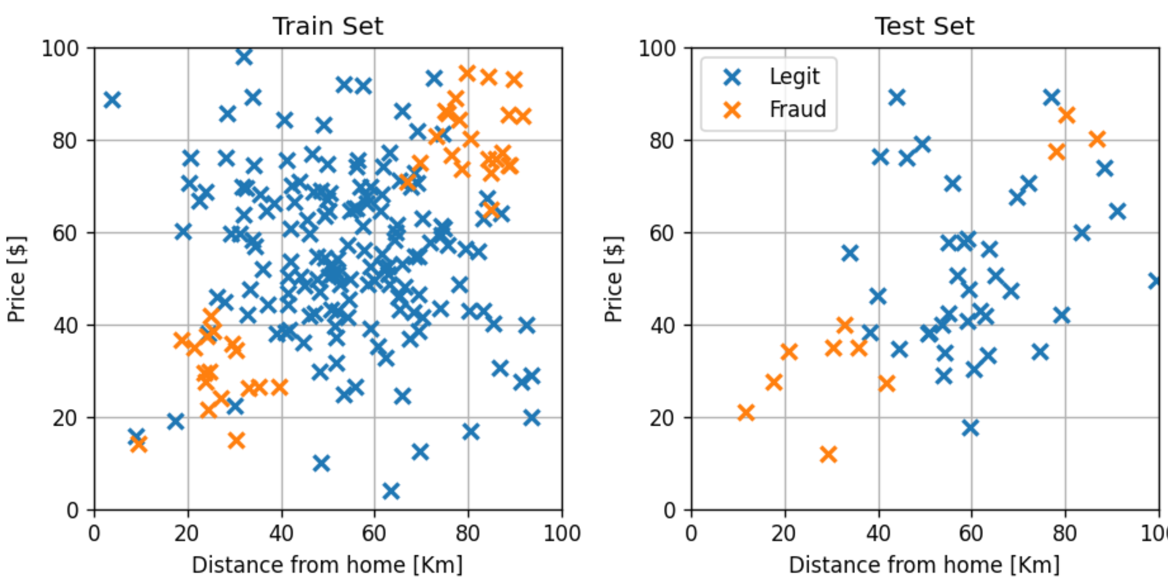
$$\theta^* = \underset{\theta}{argmin}\ \frac{1}{N}\sum_{i=1}^{N} \ell(h(x^{(i)};\theta), y^{(i)})$$

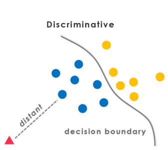
Discriminative vs Generative Models

- Recall: the problem is we don't have the posterior probability distribution $p(y|x)$
- Solution 1 (Generative Models): Estimate the joint distribution based on the dataset $\mathcal{D} = \{x^{(i)}, y^{(i)}\}_{i=1}^{N}$
- Solution 2 (Discriminative Models): Estimate the expectation empirically based on the dataset $\mathcal{D} = \{x^{(i)}, y^{(i)}\}_{i=1}^{N}$

Classification Example

- Let us get back to the classification example, and denote Legit as $y = 0$, and Fraud as $y = 1$.
- First, we split the data into 80% training and 20% test:





Discriminative Modelling

- Main Idea - Build an estimator/classifier/discriminator from the empirical risk directly:

$$\hat{h}_{\mathcal{D}}(x) = \underset{h}{argmin}\ \hat{\mathbb{E}}_{\mathcal{D}}\left[\ell(h(x), y)\right] \rightarrow \hat{h}_{\mathcal{D}}(x) = \underset{h}{argmin}\ \frac{1}{N}\sum_{i=1}^{N}\ell(h(x^{(i)}), y^{(i)})$$

- The estimator can be non-parametric ($h(x)$) or parameteric ($h(x;\theta)$).

- Here, we will discuss a non-parametric estimator (Nearest Neighbor)
- In the lecture you will cover also a parametric estimator (Support Vector Machines).

Discriminative Modelling - Nearest Neighbor Classifier

- Algorithm That classifies based on the label of the nearest sample in the dataset:
- We are given a dataset $\mathcal{D} = \left\{x^{(i)}, y^{(i)}\right\}_{i=1}^{N}$, and test sample x_q that we want to predict the label for.
- The algorithm is extremely simple and comprised of two steps:
  - Find the the index of the nearest sample in the dataset:

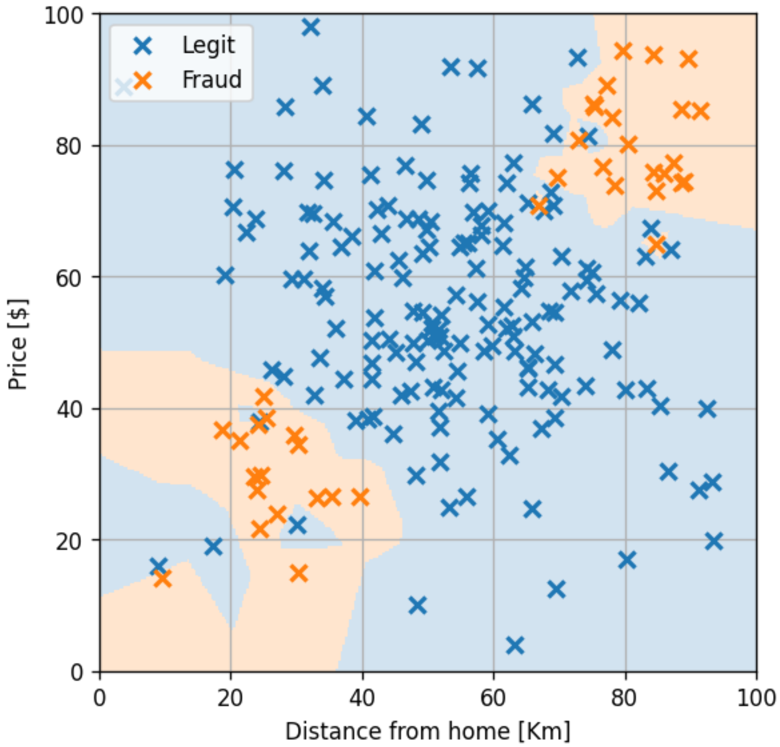
$$i = \underset{i}{argmin}\|x_q - x^{(i)}\|_2$$

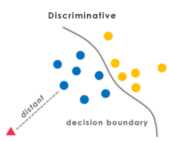
  - Predict the label to be the label at the found index:

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

Discriminative Modelling - Nearest Neighbor Classifier Cont.

- Here are the resulting decision boundaries on the training set for the classification problem:

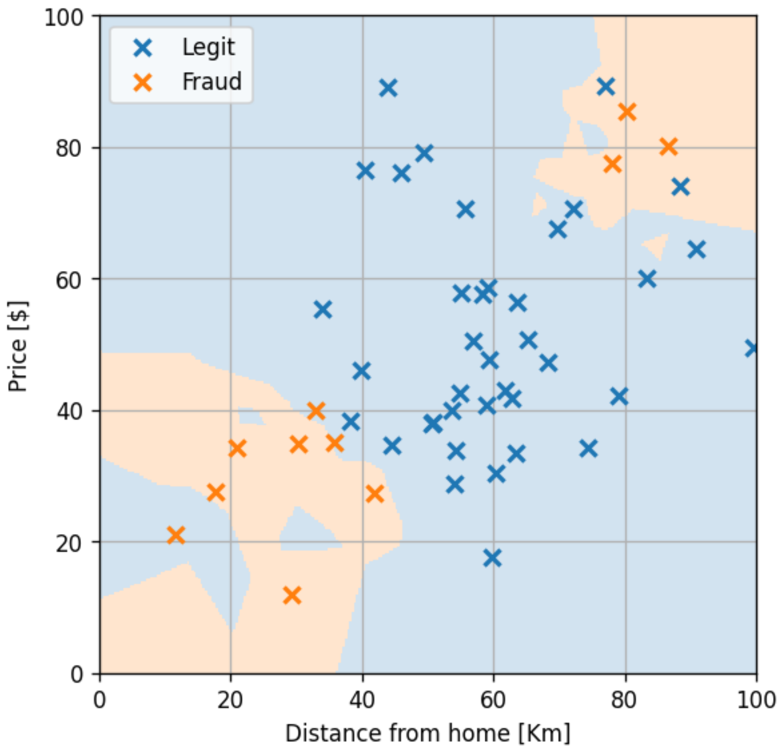


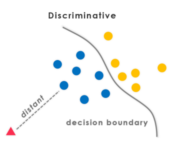


Discriminative Modelling - Nearest Neighbor Classifier Cont.

- The resulting error on the test score is 12%:

$$\text{test score} = \frac{1}{N_{\text{test}}} \sum_{x^{(i)}, y^{(i)} \in \mathcal{D}_{\text{test}}} \ell(h^*(x^{(i)}), y^{(i)}) = 0.12$$

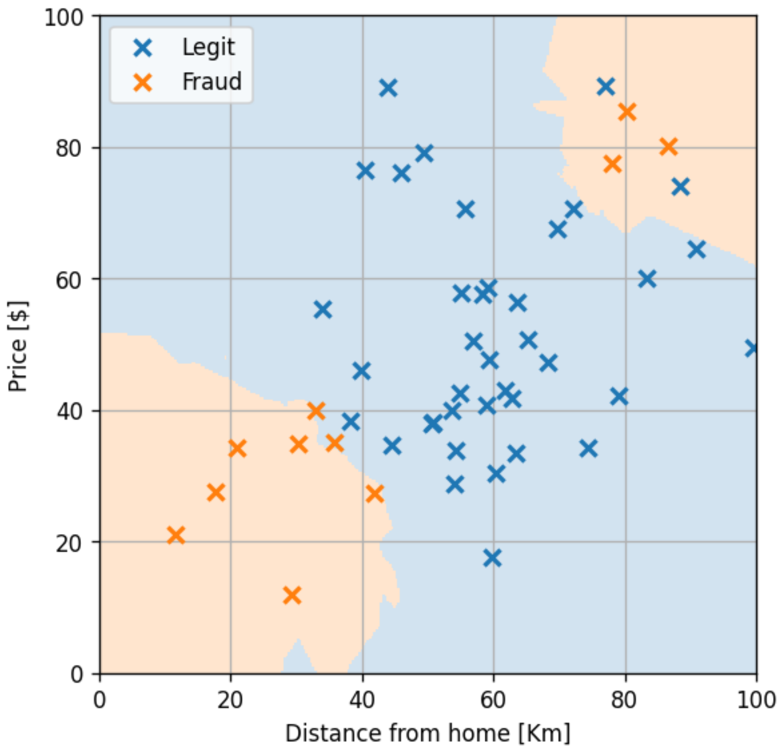


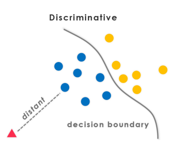


Discriminative Modelling - Nearest Neighbor Classifier Cont.

- The resulting test score can be reduced to 10% using a modified algorithm with $K = 5$ neighbors (Details in the lecture):

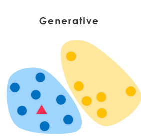
$$\text{test score} = \frac{1}{N_{\text{test}}} \sum_{x^{(i)}, y^{(i)} \in \mathcal{D}_{\text{test}}} \ell(h^*(x^{(i)}), y^{(i)}) = 0.1$$





Discriminative Modelling - Popular Algorithms

- K Nearest Neighbors
- Decision Trees
- Support Vector Machines
- Linear Least Squares Regression



Generative Modelling

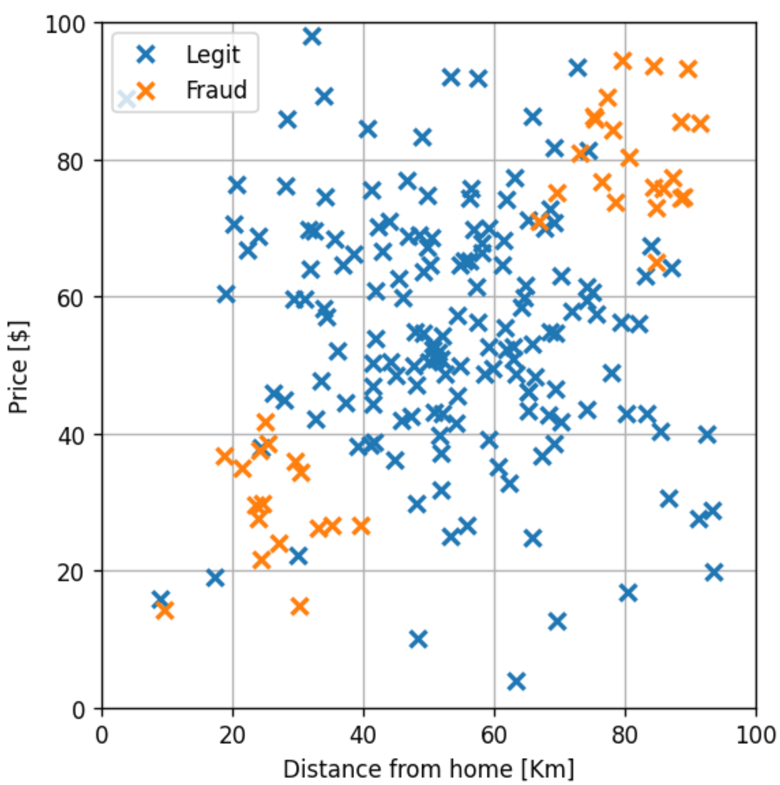
- First: Estimate the unknown joint distribution $\hat{p}(x,y)$: Conviniently separated to $\hat{p}(x|y)$ and $\hat{p}(y)$
- Second: Employ Bayes rule to get the approximated posterior (up to normalization): $\hat{p}(y|x) = \frac{\hat{p}(x|y)\hat{p}(y)}{p(x)} = \frac{\hat{p}(x|y)\hat{p}(y)}{\int_y \hat{p}(x|y)\hat{p}(y)}$
- Third: Use the approximated posterior to get the final classifier: $h^*(x) = \underset{y}{argmax}\ \hat{p}(y|\mathbf{x}=x)$
- The estimation of $p(x|y)$ can be non-parametric (KDE) or parameteric (eg Gaussians).
- Here, we will discuss a parametric estimator for $p(x|y;\theta)$ (Quadratic Discriminant Analysis)
- In the lecture you will cover a non-parametric estimator $p(x|y)$ (Naive Bayes Classifier).

Generative Modelling - Quadratic Discriminant Analysis

- Assume we have a mixed joint probability distribution $p(x, y)$, where some of the variables are continuous x, and some are discrete y.
- It is convinient in this case to write down $p(x, y) = p(x|y)p(y)$ and estimate each part separately:
  - $p(y)$ can be simply estimated using label proportions in the dataset
  - $p(x|y)$ can estimated separately for each value of y
- If we further assume a parametric form $p(x|y;\theta)$, we can turn the problem into estimating θ per y

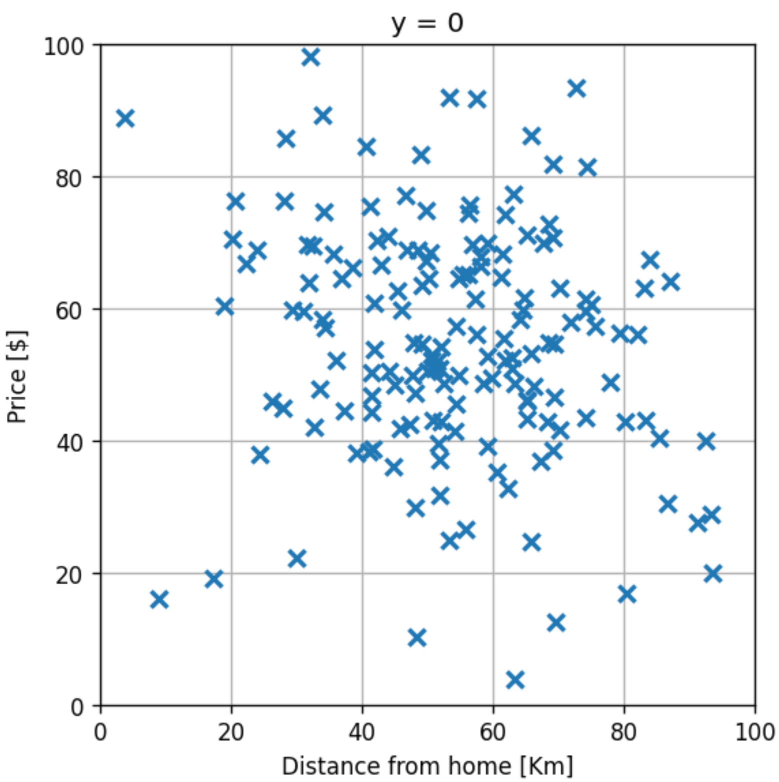
Generative Modelling - Quadratic Discriminant Analysis Cont.

- Back to the example of the credit card transactions:
- Training set have 200 data points with 160 Legit ($y = 0$) and 40 Fraud ($y = 1$).
- Therefore $p(y)$ can be estimated using the proportions: $p(y = 0) = \frac{160}{200} = 0.8$ and $p(y = 1) = \frac{40}{200} = 0.2$



Generative Modelling - Quadratic Discriminant Analysis Cont.

- Next, we estimate a conditional density $p(x|y)$ for each y separately: $p(x|y=0)$ and $p(x|y=1)$.
- This can done with a non-parametric method (eg KDE) or a parametric one (eg Gaussian density).
- For the parametric case, we can derive the optimal parameters using Maximum Likelihood Estimation (MLE).



Generative Modelling - Quadratic Discriminant Analysis Cont.

- Assuming a parametric Gaussian density, then for each $p(x|y)$ we will estimate an expectation vector $\mu_{x|y}$ and a covariance matrix $\Sigma_x|y$.
- The resulting estimated posterior probabilities are given by:

$$\hat{p}(y=0|x) = \frac{\hat{p}(x|y=0)\hat{p}(y=0)}{p(x)}, \ \hat{p}(y=1|x) = \frac{\hat{p}(x|y=1)\hat{p}(y=1)}{p(x)}$$

- Finally the resulting estimator is given by the mode of the resulting posterior using Bayes rule:

$$h^*(x) = \underset{y}{argmax}\ \hat{p}(y|\mathbf{x}=x) = \underset{y}{argmax}\ \{\hat{p}(y=0|x), \hat{p}(y=1|x)\}$$

- This translates to the criterion:

$$h^*(x) = \begin{cases} 0 & \text{if } \hat{p}(x|y=0)\hat{p}(y=0) > \hat{p}(x|y=1)\hat{p}(y=1) \\ 1 & \text{otherwise.} \end{cases}$$

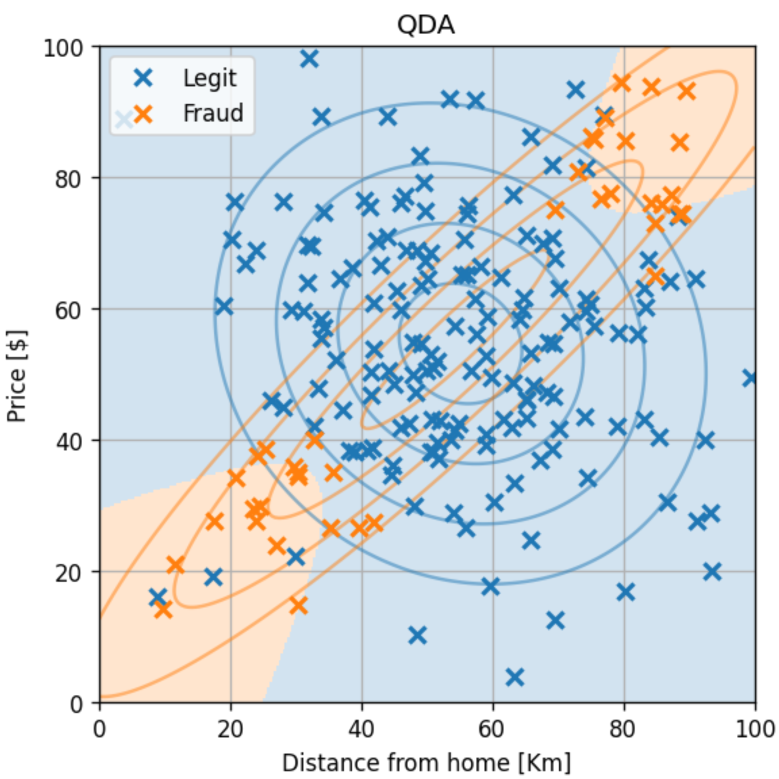
- Substituting the Gaussian distribution in $p(x|y=0/1)$ results in quadratic boundaries, hence the name.

Generative

Generative Modelling - Quadratic Discriminant Analysis Cont.

- Note that the orange Gaussian is probably a poor model choice and a mixture model of two Gaussians would have worked much better.
- The resulting test score is 8%:

$$\text{test score} = \frac{1}{N_{\text{test}}} \sum_{x^{(i)},y^{(i)} \in \mathcal{D}_{\text{test}}} \ell(h^*(x^{(i)}), y^{(i)}) = 0.08$$

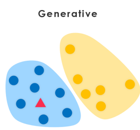


Generative

Generative Modelling - Popular Algorithms

- Linear Discriminant Analysis
- Quadratic Discriminant analysis
- Naive Bayes

- Markov Random Fields



Generative Modelling - Limitations

- Suffers from the curse of dimensionality:
  - Space coverage becomes increasly more difficult for higher dimensions of x
  - Number of samples n needed to estimate the conditional density $p(x|y)$ is exponential in the dimension: $\approx n^d$
  - Naive solution via assuming independence (Naive Bayes Classifier)
- Models we can work with are very limited, because we need to satisfy:
  - $p(x, y; \theta) \geq 0, \ \forall x, y, \theta$
  - $\int \int p(x, y; \theta) = 1, \ \forall \theta$

Probabilistic Discriminative Models

- Recall: the problem is we don't have the posterior probability distribution $p(y|x)$
- Solution 1 (Generative Models): Estimate the **joint** distribution based on the dataset $\mathcal{D} = \left\{x^{(i)}, y^{(i)}\right\}_{i=1}^{N}$
- Solution 2 (Discriminative Models): Estimate the expectation empirically based on the dataset $\mathcal{D} = \left\{x^{(i)}, y^{(i)}\right\}_{i=1}^{N}$
- Solution 3 (Prob. Discriminative Models): Estimate the **posterior** distribution **directly** based on the dataset $\mathcal{D} = \left\{x^{(i)}, y^{(i)}\right\}_{i=1}^{N}$

Probabilistic Discriminative Models -General Idea

- Usually these methods in most books will be called simply discriminative without the "probabilistic".
- However, most good books will still differentiate between different types of discriminative models although without special names.
- Here, we will estimate directly $p(y|x)$, usually by parameterizing it to $p(y|x; \theta)$.
- Parameter estimation as usual will be done with maximum likelihood.

Probabilistic Discriminative Models -General Idea Cont.

- For classification problems, the function $p(y|x; \theta)$ should satisfy:
  - $p(y|x; \theta) \geq 0, \ \forall x, y, \theta$
  - $\sum_{y=1}^{C} p(y|x; \theta) = 1, \ \forall x, \theta$

- The second here is much simpler than the condition for generative models which was: $\int\int p(x,y;\theta)=1,\ \forall\theta$
- Such models can be easily constructed, for example for $C=2$ classes (binary classification), we only demand:

$$p(y=0|x;\theta)+p(y=1|x;\theta)=1\ \forall x,\theta$$

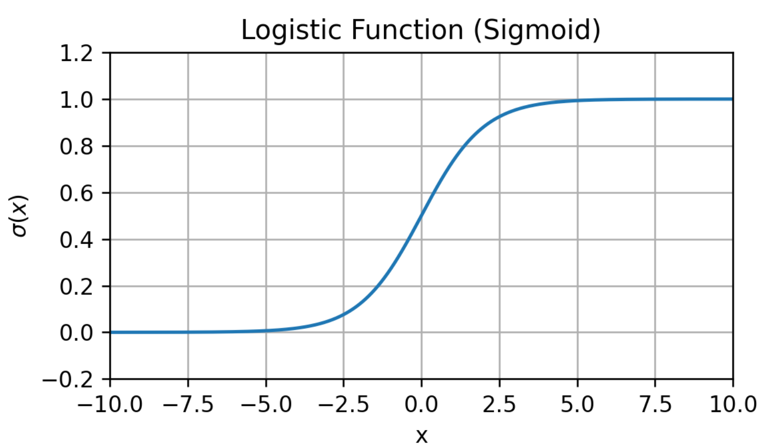
- Any parametric function $f(x;\theta)$ that returns values between 0 and 1 can define a valid model like so:

$$p(y=1|x)=f(x;\theta)$$
$$p(y=0|x)=1-f(x;\theta)$$

Probabilistic Discriminative Models - Binary Classification Example

- Assume we are dealing with a binary classification problem $y\in\{0,1\}$.
- Using the sigmoid function we can even relax the range condition of $f(x;\theta)$.
- Known as Logistic Regression in the literature.
- Any parametric model of the following form is valid:

$$p(y=1|x)=\sigma(f(x;\theta))$$
$$p(y=0|x)=1-\sigma(f(x;\theta))$$



Probabilistic Discriminative Models - Binary Classification Example

- Estimating the parameters θ with MLE we get the following objective function:

$$\theta^* = \underset{\theta}{\operatorname{argmin}} - \sum_{i=1}^{N} \log(p(y^{(i)}|x^{(i)};\theta))$$

$$= \underset{\theta}{\operatorname{argmin}} - \sum_{i=1}^{N} I\left\{y^{(i)} = 1\right\} \log(\sigma(f(x^{(i)};\theta)) + I\left\{y^{(i)} = 0\right\} \log(1 - \sigma(f(x^{(i)};\theta))$$

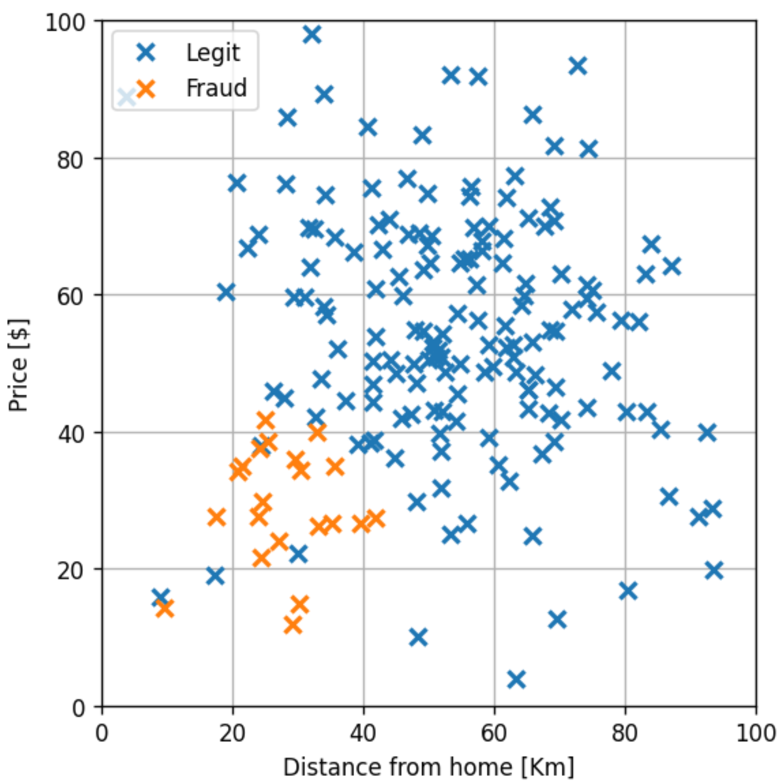
$$= \underset{\theta}{\operatorname{argmin}} - \sum_{i=1}^{N} y^{(i)} \log(\sigma(f(x^{(i)};\theta)) + (1 - y^{(i)}) \log(1 - \sigma(f(x^{(i)};\theta))$$

- Resulting estimator after optimization:

$$h(x) = \underset{y}{\operatorname{argmax}} p(y|x;\theta) = \begin{cases} 1 & \text{if } \sigma(f(x;\theta)) > 0.5 \\ 0 & \text{otherwise.} \end{cases} = \begin{cases} 1 & \text{if } f(x;\theta) > 0 \\ 0 & \text{otherwise.} \end{cases}$$

Probabilistic Discriminative Models - Binary Classification Example

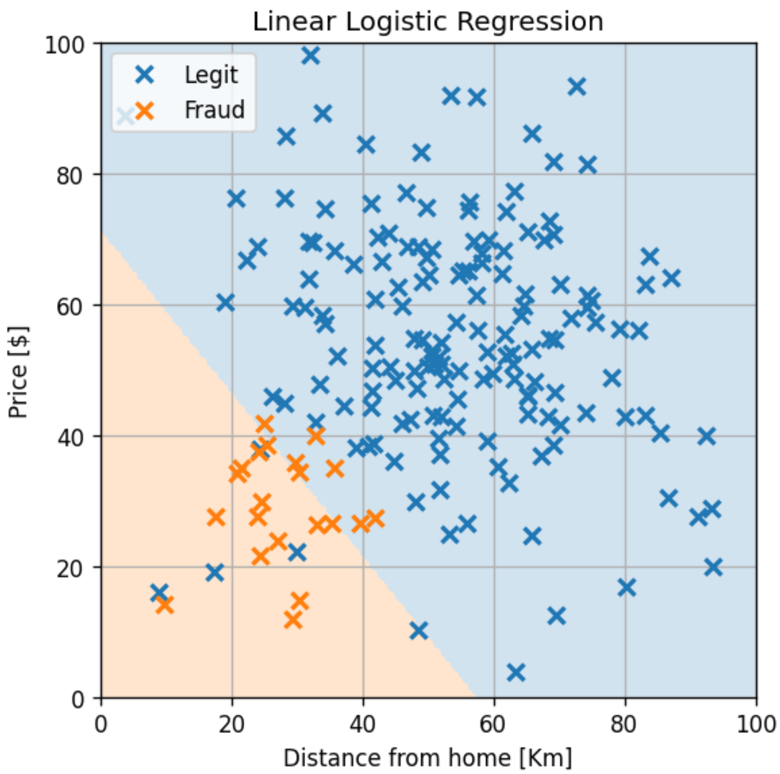
- Coming back to the fraud example from earlier (taking only one part of the space for simplicity):



Probabilistic Discriminative Models - Binary Classification Example

- Fitting a linear parametric function: $f(x;\theta) = \theta^T x$
- The resulting test score is 2%:

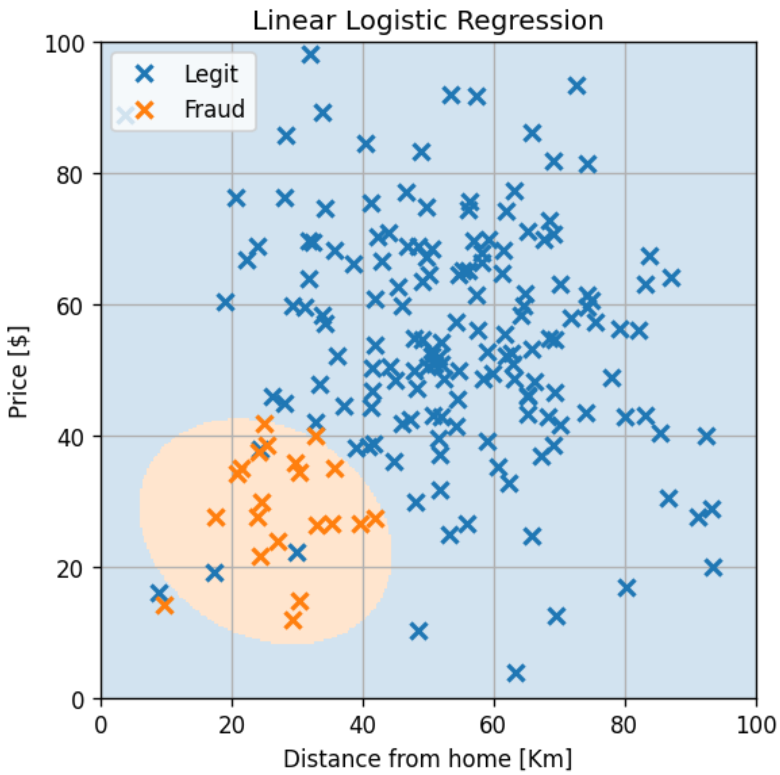
$$\text{test score} = \frac{1}{N_{\text{test}}} \sum_{x^{(i)}, y^{(i)} \in \mathcal{D}_{\text{test}}} \ell(h^*(x^{(i)}), y^{(i)}) = 0.02$$



Probabilistic Discriminative Models - Binary Classification Example

- For $f(x;\theta)$ taken as a second order polynomial, the resulting test score is 0% (no mistakes!):

$$\text{test score} = \frac{1}{N_{\text{test}}} \sum_{x^{(i)}, y^{(i)} \in \mathcal{D}_{\text{test}}} \ell(h^*(x^{(i)}), y^{(i)}) = 0$$



Recommended Videos

Warning!

- These videos do not replace the lectures and tutorials.
- Please use these to get a better understanding of the material, and not as an alternative to the written material.

Video By Subject

- Machine Learning Course by Andrew Ng - Coursera
- Lectures 1-7, EE 046195 by Omer Yair - Technion

Credits

- EE 046746 Spring 2021 - Elias Nehme
- Lectures 1-7, EE 046195 Spring 2021 - Omer Yair
- What are the Types of Machine Learning? - Hunter Heidenreich

- Icons from Icon8.com - https://icons8.com