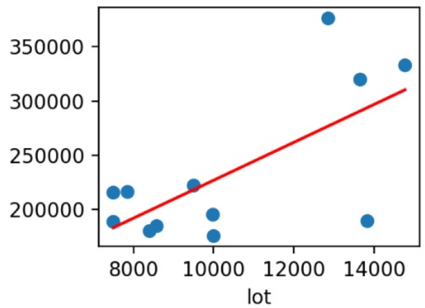
Lecture 4
Supervised Learning: Multiclass Classification

KMA Solaiman
Fall 2023

Partially Adapted from
Chris Re
Stanford ML

Revisiting Linear Regression



Let $h_\theta(x) = \sum_{j=0}^{d} \theta_j x_j$ want to choose θ so that $h_\theta(x) \approx y$. One popular idea called **least squares**

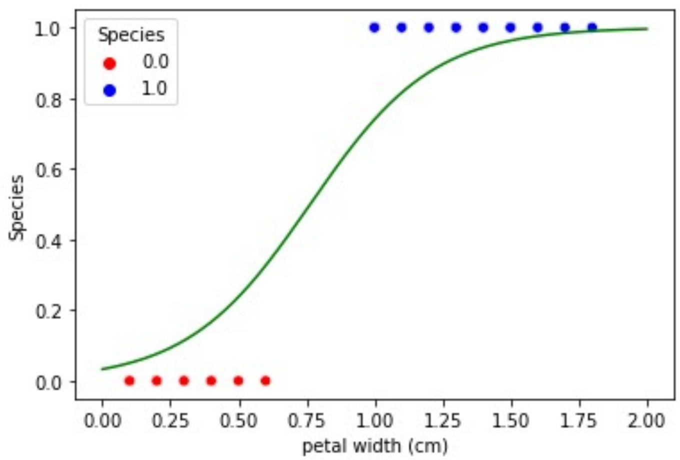
$$J(\theta) = \frac{1}{2} \sum_{i=1}^{n} \left(h_\theta(x^{(i)}) - y^{(i)} \right)^2 .$$

Choose

$$\theta = \underset{\theta}{\operatorname{argmin}}\, J(\theta).$$

Minimize for θ

Revisiting Logistic Regression



Graph of Iris Dataset with logistic regression

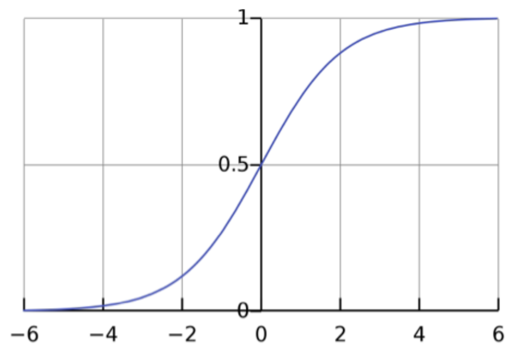
Given a training set $\{(x^{(i)}, y^{(i)})$ for $i = 1, \ldots, n\}$ let $y^{(i)} \in \{0, 1\}$.
Want $h_\theta(x) \in [0, 1]$. Let's pick a smooth function:

$$h_\theta(x) = g(\theta^T x)$$

Here, g is a link function. There are *many*... but we'll pick one!

$$g(z) = \frac{1}{1 + e^{-z}}.$$

SIGMOID



How do we interpret $h_\theta(x)$?

$$P(y = 1 \mid x; \theta) = h_\theta(x)$$
$$P(y = 0 \mid x; \theta) = 1 - h_\theta(x)$$

Logistic Regression: Link Functions

Let's write the Likelihood function. Recall:

$$P(y = 1 \mid x; \theta) = h_\theta(x)$$
$$P(y = 0 \mid x; \theta) = 1 - h_\theta(x)$$

Then,

$$L(\theta) = P(y \mid X; \theta) = \prod_{i=1}^{n} p(y^{(i)} \mid x^{(i)}; \theta)$$
$$= \prod_{i=1}^{n} h_\theta(x^{(i)})^{y^{(i)}} (1 - h_\theta(x^{(i)}))^{1-y^{(i)}} \quad \text{exponents encode “if-then”}$$

Taking logs to compute the log likelihood $\ell(\theta)$ we have:

Maximize for θ

$$\ell(\theta) = \log L(\theta) = \sum_{i=1}^{n} y^{(i)} \log h_\theta(x^{(i)}) + (1 - y^{(i)}) \log(1 - h_\theta(x^{(i)}))$$

(Log) Likelihoods!

So we've shown that finding a θ to maximize $L(\theta)$ is the same as *maximizing*

$$\ell(\theta) = C(\sigma, n) - \frac{1}{\sigma^2} J(\theta)$$

Or minimizing, $J(\theta)$ directly (why?)

Takeaway: "Under the hood," solving least squares *is* solving a maximum likelihood problem for a particular probabilistic model.

This view shows a path to generalize to new situations!

Solving the optimization problem

- Stochastic *Gradient Descent*

$$\theta^{(0)} = 0$$

$$\theta_j^{(t+1)} = \theta_j^{(t)} - \alpha \frac{\partial}{\partial \theta_j} J(\theta^{(t)}) \qquad \text{for } j = 0, \ldots, d.$$

Thus, our update rule for component j can be written:

After computing derivatives

$$\theta_j^{(t+1)} = \theta_j^{(t)} - \alpha \sum_{i=1}^{n} \left(h_\theta(x^{(i)}) - y^{(i)} \right) x_j^{(i)}.$$

Summary for binary classification/ logistic regression

- Calculate $h_\theta(x) = g(\theta^T x)$
- Get $P(y \mid X; \theta)$ using $h_\theta(x)$, that's likelihood
- Calculate log likelihood from there
- Maximize log likelihood from there –
  use SGD to maximize for θ
  - Start with a guess for θ
  - Keep updating with the rule until convergence

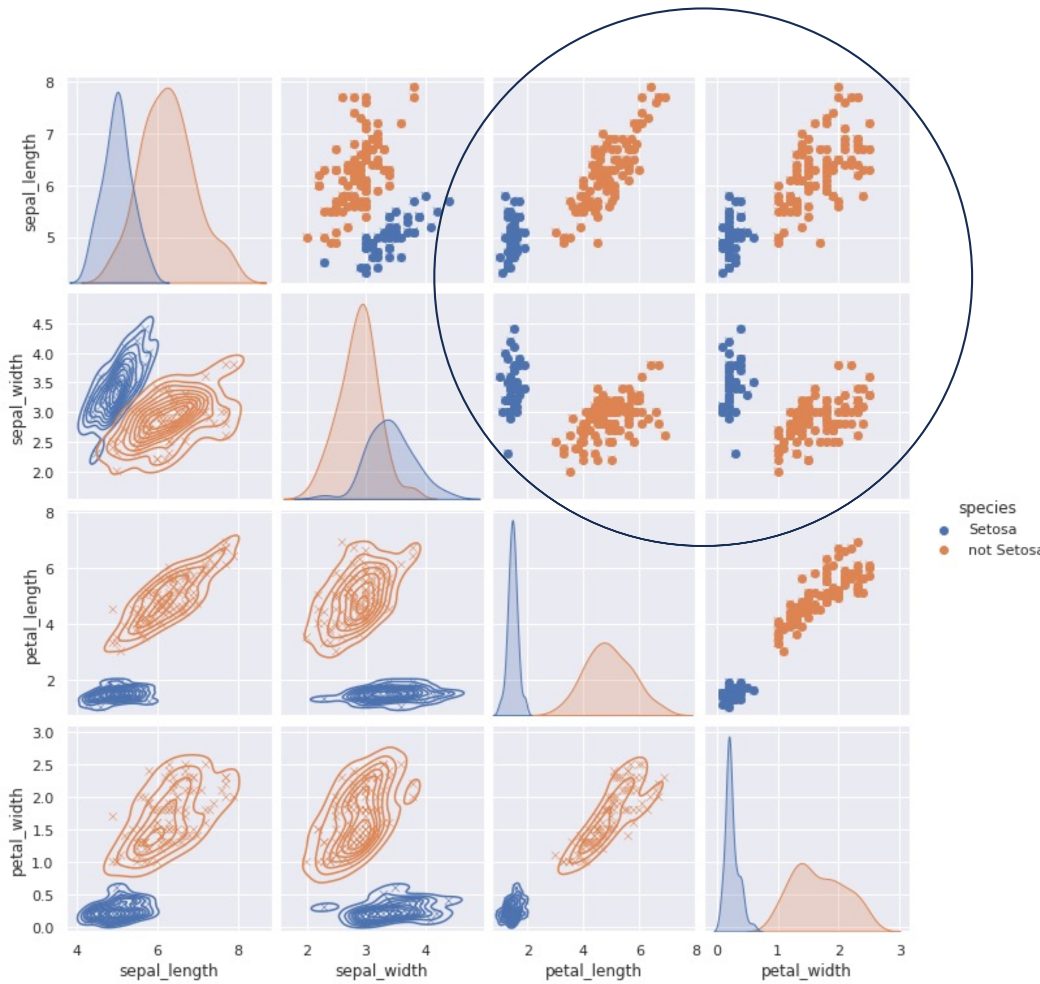
Predicted Output

inference $h_\theta(x)$ is the **output**.

learn $\max_\theta \log p(y \mid x; \theta)$ by maximum likelihood.

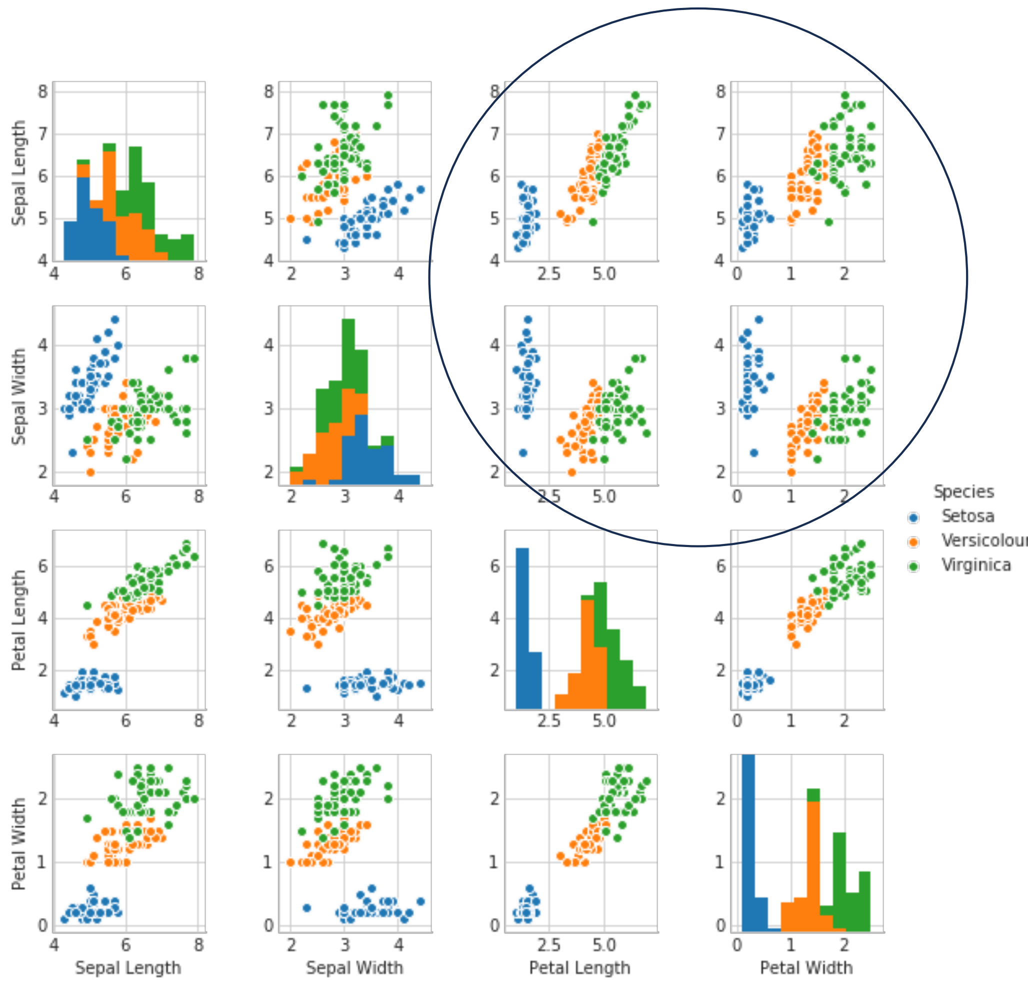
algorithm: SGD $\theta^{(t+1)} = \theta^{(t)} + \alpha \left(y^{(i)} - h_{\theta^{(t)}}(x^{(i)}) \right) x^{(i)}.$

Binary Classification

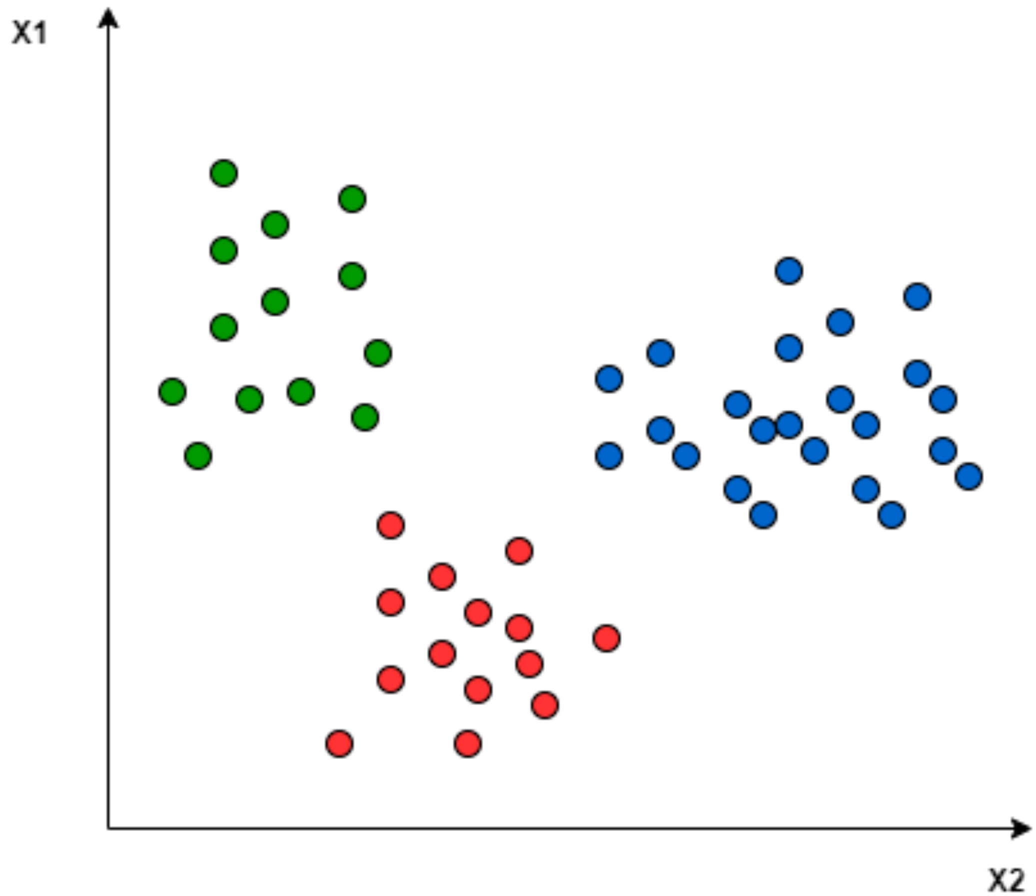


https://www.kaggle.com/code/nicolaspieser/binary-classification-on-the-iris-dataset

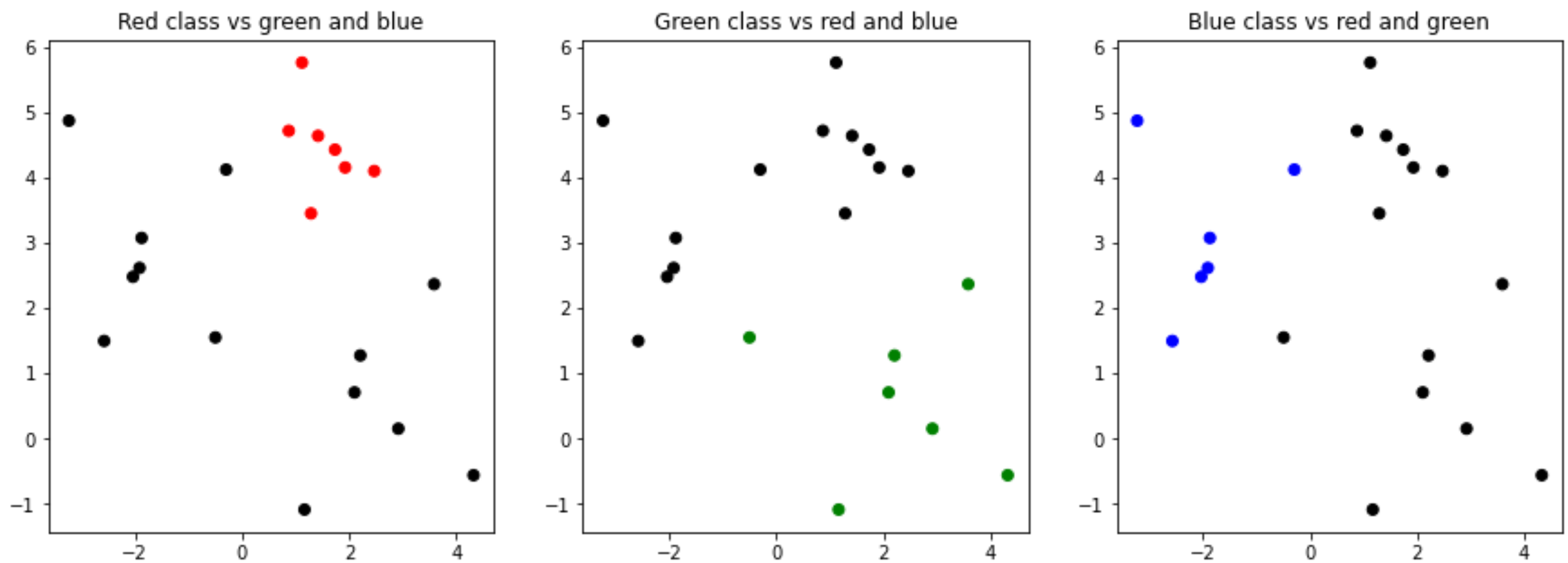
Multiclass
classification
Classification



https://www.kaggle.com/code/mattwills8/multi-class-classification-of-iris-dataset


X1
X2

1 vs All


Red class vs green and blue
Green class vs red and blue
Blue class vs red and green

A Quick and Dirty Intro to Multiclass Classification.
This technique is *the daily workhorse of modern AI/ML*

Multiclass

Suppose we want to choose among k discrete values, e.g., $\{\text{'Cat'}, \text{'Dog'}, \text{'Car'}, \text{'Bus'}\}$ so $k = 4$.

We encode with **one-hot** vectors i.e. $y \in \{0,1\}^k$ and $\sum_{j=1}^{k} y_j = 1$.

$$\underset{\text{`Cat'}}{\begin{pmatrix}1\\0\\0\\0\end{pmatrix}} \quad \underset{\text{`Dog'}}{\begin{pmatrix}0\\1\\0\\0\end{pmatrix}} \quad \underset{\text{`Car'}}{\begin{pmatrix}0\\0\\1\\0\end{pmatrix}} \quad \underset{\text{`Bus'}}{\begin{pmatrix}0\\0\\0\\1\end{pmatrix}}$$

A prediction here is actually a *distribution* over the k classes. This leads to the SoftMax function described below (derivation in the notes!). That is our hypothesis is a vector of k values:

$$P(y = j | x; \bar{\theta}) = \frac{\exp(\theta_j^T x)}{\sum_{i=1}^{k} \exp(\theta_i^T x)}.$$

Here each θ_j has the *same dimension* as x, i.e., $x, \theta_j \in R^{d+1}$ for $j = 1, \ldots, k$.

Quick Comments on Presentation

- *Check for home:* does $k = 2$ case agree with logistic regression?

$$P(y = j|x;\theta) = \frac{e^{\theta_j^T x}}{e^{\theta_1^T x} + e^{\theta_2^T x}}$$

Hint: Given (θ_1, θ_2) for a two class model, compare with logistic regression with the model $\theta_1 - \theta_2$.

- For general k, a probability estimate for any $k - 1$ classes determines the other class (since estimates must sum to 1).

How do you train multiclass? (Picture Version)

$$P(y = j|x; \theta) = \frac{\exp(\theta_j^T x)}{\sum_{i=1}^{k} \exp(\theta_i^T x)}.$$

Intuitively, we maximize the probability of the given class.

How do you train multiclass?

Fixing x and θ, our output is a vector $\hat{p} \in \mathbb{R}_+^k$ s.t. $\sum_{j=1}^k \hat{p}_j = 1$.

$$\hat{p}_j = P(y = j | x; \theta) = \frac{\exp(\theta_j^T x)}{\sum_{i=1}^k \exp(\theta_i^T x)}.$$

Formally, we maximize the probability of the given class!

How do you train multiclass?

Fixing x and θ, our output is a vector $\hat{p} \in \mathbb{R}_+^k$ s.t. $\sum_{j=1}^k \hat{p}_j = 1$.

$$\hat{p}_j = P(y = j|x; \theta) = \frac{\exp(\theta_j^T x)}{\sum_{i=1}^k \exp(\theta_i^T x)}.$$

Formally, we maximize the probability of the given class!
We can view as $\textsc{CrossEntropy}$:

$$\textsc{CrossEntropy}(p, \hat{p}) = -\sum_j p(x = j) \log \hat{p}(x = j).$$

Here, p is the label, which is a one-hot vector.

How do you train multiclass?

Fixing x and θ, our output is a vector $\hat{p} \in \mathbb{R}_{+}^{k}$ s.t. $\sum_{j=1}^{k} \hat{p}_j = 1$.

$$\hat{p}_j = P(y = j | x; \theta) = \frac{\exp(\theta_j^T x)}{\sum_{i=1}^{k} \exp(\theta_i^T x)}.$$

Formally, we maximize the probability of the given class!
We can view as $\textsc{CrossEntropy}$:

$$\textsc{CrossEntropy}(p, \hat{p}) = -\sum_{j} p(x = j) \log \hat{p}(x = j).$$

Here, p is the label, which is a one-hot vector.Thus, if the label is i, this formula reduces to:

$$-\log \hat{p}(x = i) = -\log \frac{\exp(\theta_i^T x)}{\sum_{j=1}^{k} \exp(\theta_j^T x)}.$$

We minimize this–and you've seen the movie, it works the same as the others!