Applied Machine Learning

Logistic and Softmax Regression

Isabeau Prémont-Schwarz

McGill School of Computer Science

Learning objectives

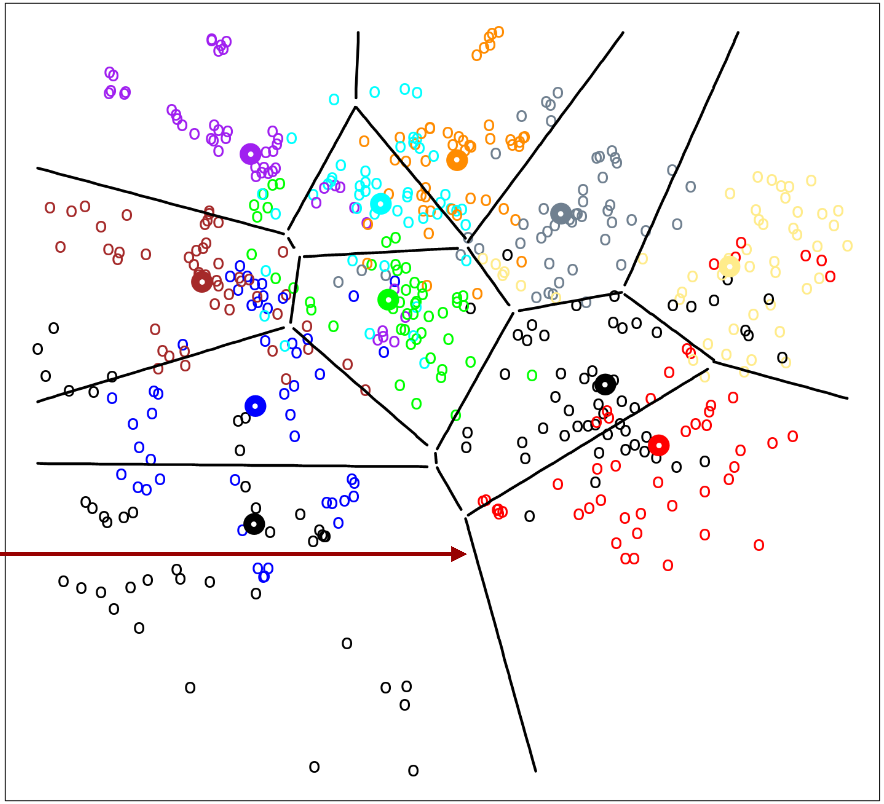
- what are linear classifiers
- logistic regression
    - model
    - loss function
- maximum likelihood view
- multi-class classification

Classification problem

dataset of inputs $x^{(n)} \in \mathbb{R}^D$

and discrete targets $y^{(n)} \in \{1, \ldots, C\}$

binary classification $y^{(n)} \in \{0, 1\}$



linear classification:

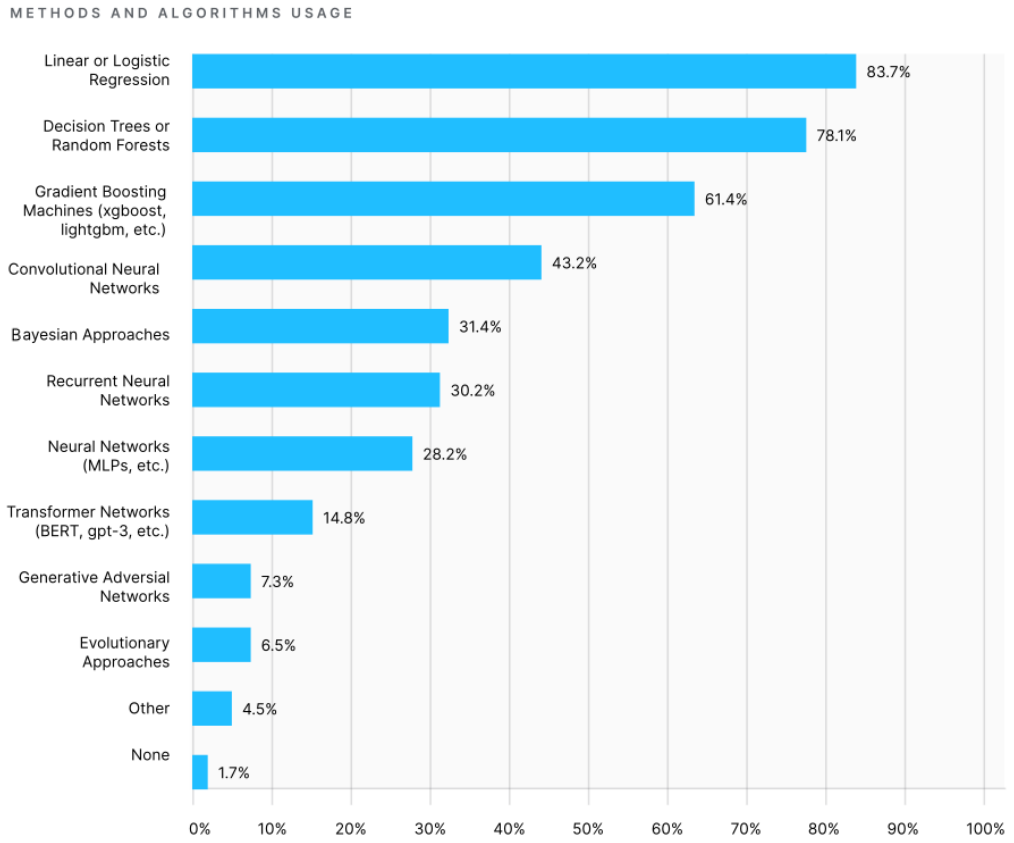
linear decision boundary $w^\top x + b$

how do we find these boundaries?

different approaches give different linear classifiers

Motivation

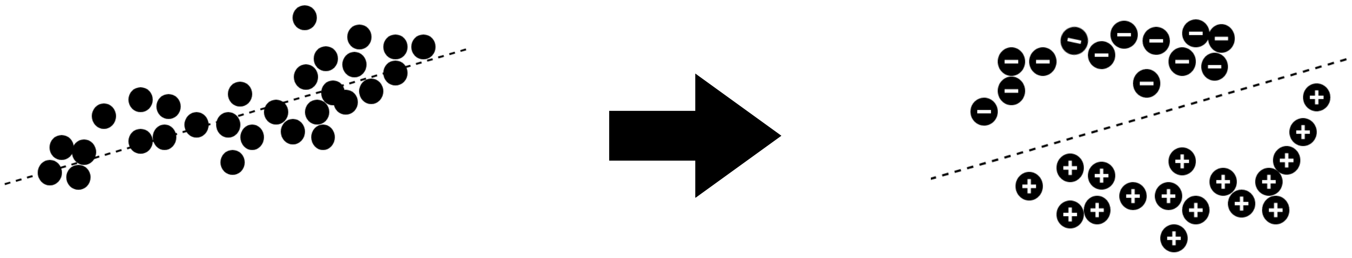
Logistic Regression is **the** most commonly reported data science method used in practice



from 2020 Kaggle's survey on the state of Machine Learning and Data Science,
you can read the full version here

Linear regression for classification?

first idea adapting linear regression to do classification?



Linear regression $y \in [0, 1]$

Logistic regression $y \in \{0, 1\}$

A linear classifier!

Linear regression for classification?

first idea adapting linear regression to do classification?

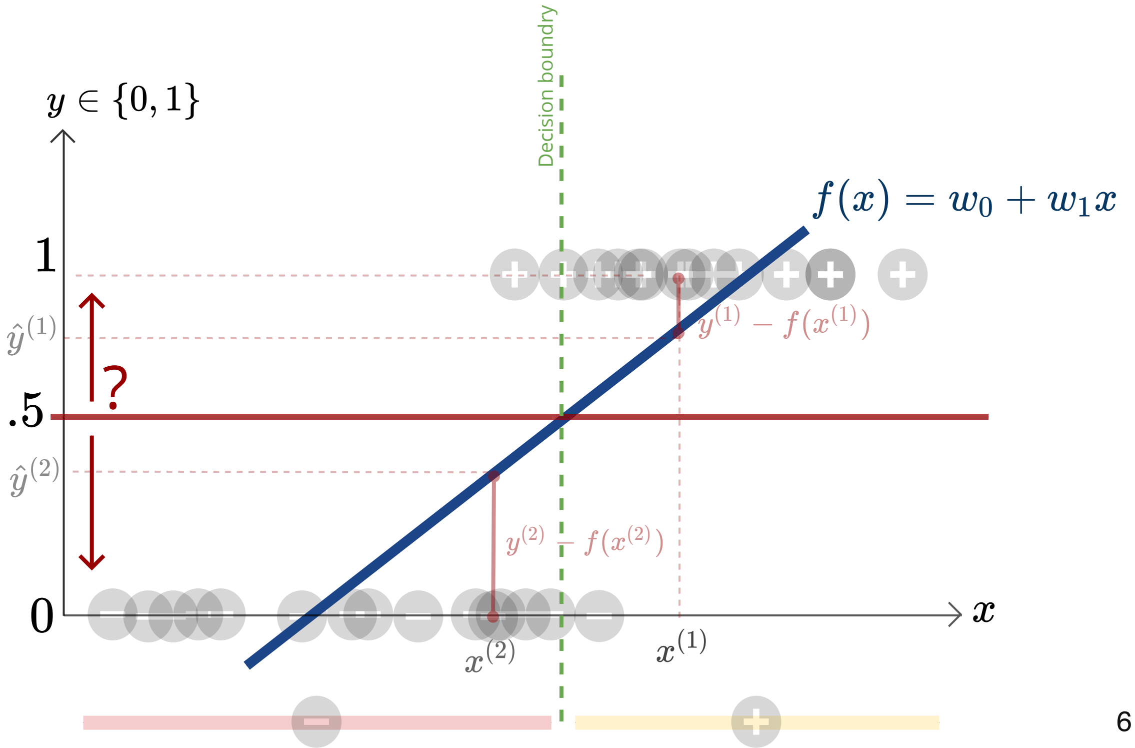
Use 1 and 0 as the target value directly apply linear regression

Using L2 loss:

$$w^* = \arg\min_w \frac{1}{2} \sum_{n=1}^{N} (w^T x^{(n)} - y^{(n)})^2$$

How to get a binary output?

- Threshold $y = \mathbb{I}(f(x) > 0.5)$
- Interpret output as probability

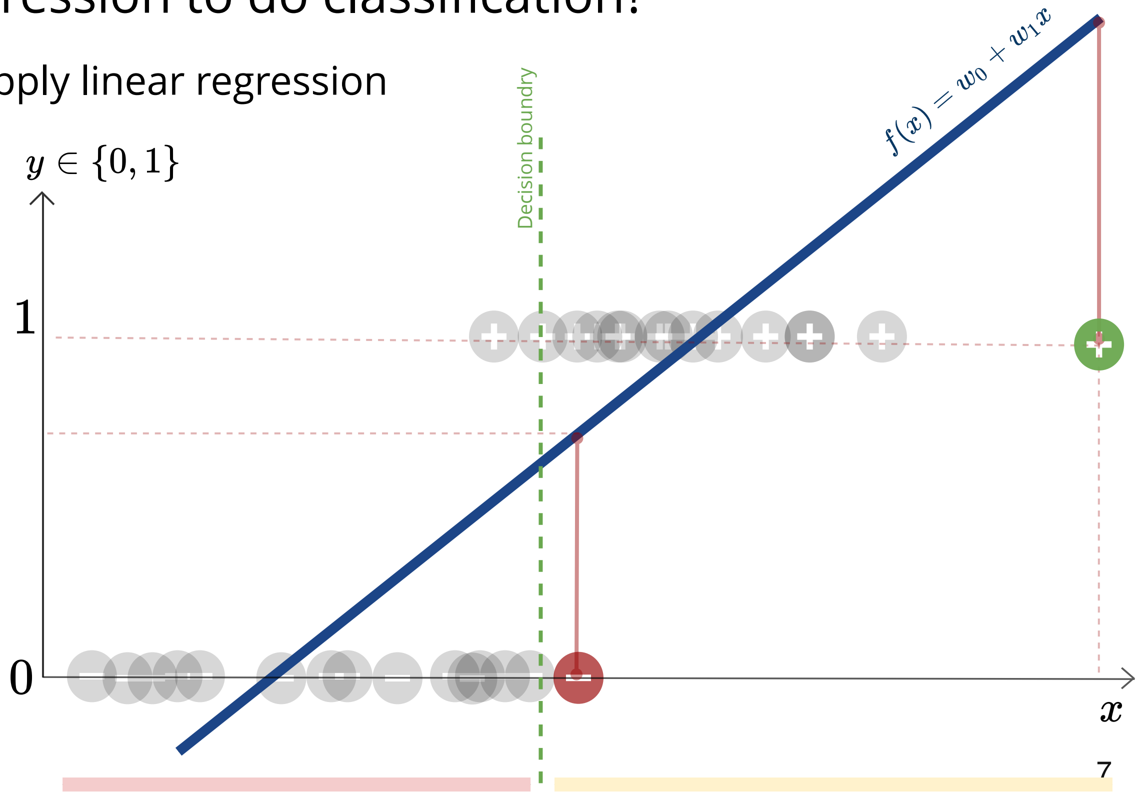


Linear regression for classification?

first idea adapting linear regression to do classification?

Use 1 and 0 as the target value directly apply linear regression

With L2 loss, correct prediction can have higher loss than the incorrect one!



Linear regression for classification?

first idea adapting linear regression to do classification?

Use 1 and 0 as the target value directly apply linear regression

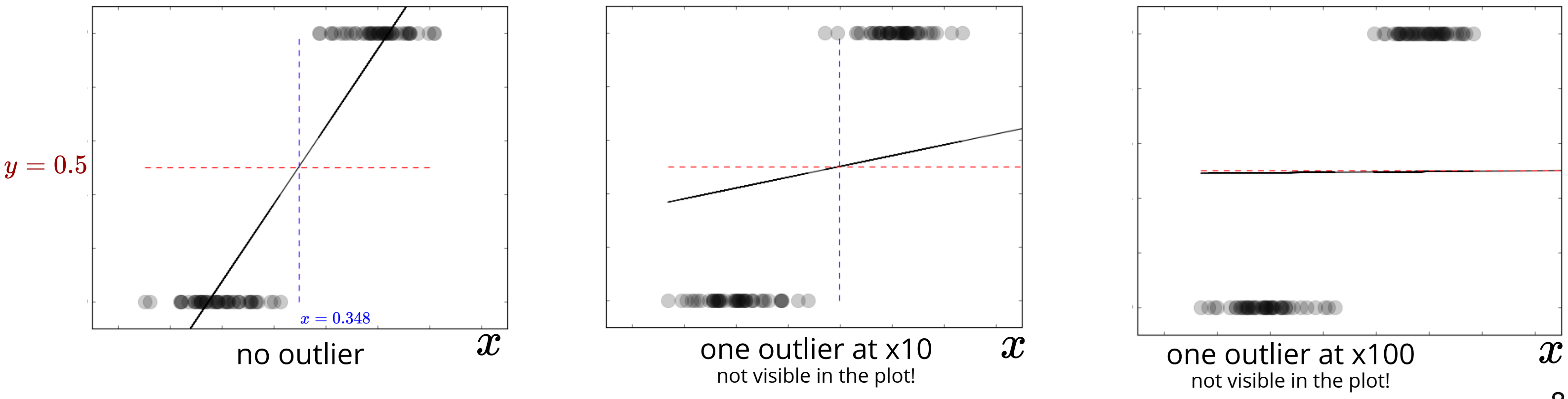
Sensitivity to Outliers, which can dominate the L2 loss (sum of least squares)

Example [D=1]: A single outlier can dominate the L2 loss

Decision boundary is a **D-1** hyperplane,

e.g. here a constant (x=0.348)

Fitted regression model is a D dimentional hyperplane, here a line



Logistic function

Idea 1: output the probability of belonging to class 1: $f(x, w) = p(y=1|x)$

Logistic function

Idea 1: output the probability of belonging to class 1: $f(x, w) = p(y=1|x)$

then (in the binary classification case) : $p(y=0|x) = 1 - p(y=1|x) = 1 - f(x,w)$

Logistic function

Idea 1: output the probability of belonging to class 1: $f(x, w) = p(y=1|x)$

then (in the binary classification case) : $p(y=0|x) = 1 - p(y=1|x) = 1 - f(x,w)$

WHY?

Why don't we just output 0 or 1, why the probability?

Because we need to output continuous values for continuous optimization, and probability is a way to do that which **means something**.

Logistic function

Idea 2: to have a linear boundary between categories we need

$$p(y=1|x,\omega)=0.5 \iff \omega^T x - C = 0$$

$$\text{If we suppose}$$

$$p(y=1|x,\omega)=f(\omega^T x)$$

$$\text{then}$$

$$p(y=1|x,\omega)=0.5 \iff f(\omega^T x)=0.5$$

$$\iff \omega^T x = f^{-1}(0.5) \iff \omega^T x - f^{-1}(0.5) = 0$$

Logistic function

So we have: $p(y=1|x,\omega) = f(\omega^T x)$

How do we choose f?

Logistic function

So we have: $p(y=1|x,\omega) = f(\omega^T x)$

How do we choose f?

f needs to be:

- invertable
- positive
- strictly monotonic

Logistic function

So we have: $p(y=1|x,\omega) = f(\omega^T x)$

How do we choose f?

A simple choice:

f needs to be:

- invertable
- positive
- strictly monotonic

$$p(y=1|x,\omega)/p(y=0|x,\omega) = \exp(\omega^T x)$$

Logistic function

So we have: $p(y=1|x,\omega) = f(\omega^T x)$

How do we choose f?

A simple choice:

$$p(y=1|x,\omega)/p(y=0|x,\omega) = \exp(\omega^T x)$$

f needs to be:

- invertable
- positive
- strictly monotonic

Solving for p(y=1|x, ω) :

$$\exp(\omega^T x) = \frac{p(y=1|x,\omega)}{p(y=0|x,\omega)} = \frac{p(y=1|x,\omega)}{1-p(y=1|x,\omega)}$$

$$\exp(\omega^T x) \cdot (1 - p(y=1|x,\omega)) = p(y=1|x,\omega)$$

$$\exp(\omega^T x) = (1 + \exp(\omega^T x))p(y=1|x,\omega)$$

$$p(y=1|x,\omega) = \frac{\exp(\omega^T x)}{1+\exp(\omega^T x)} = \frac{1}{1+\exp(-\omega^T x)}$$

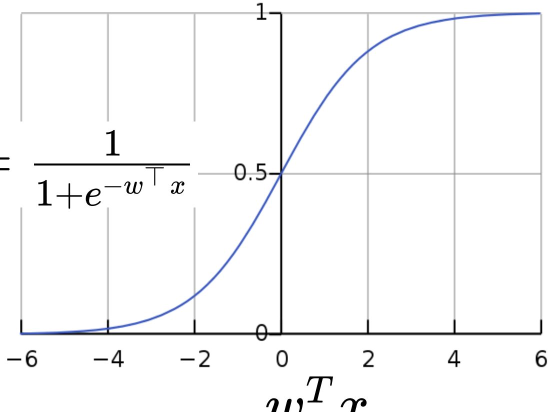
Logistic function

desirable property of $\sigma : \mathbb{R} \to \mathbb{R}, \ \sigma(x) := \frac{1}{1+\exp(-x)}$

- all $w^\top x > 0$ are squashed close together
- all $w^\top x < 0$ are squashed together

logistic function (aka The Sigmoid) has these properties

$$\sigma(w^\top x) = \frac{1}{1+e^{-w^\top x}}$$



the decision boundary is

$$w^\top x = 0 \Leftrightarrow \sigma(w^\top x) = \frac{1}{2}$$

still a linear decision boundary

Logistic regression: model

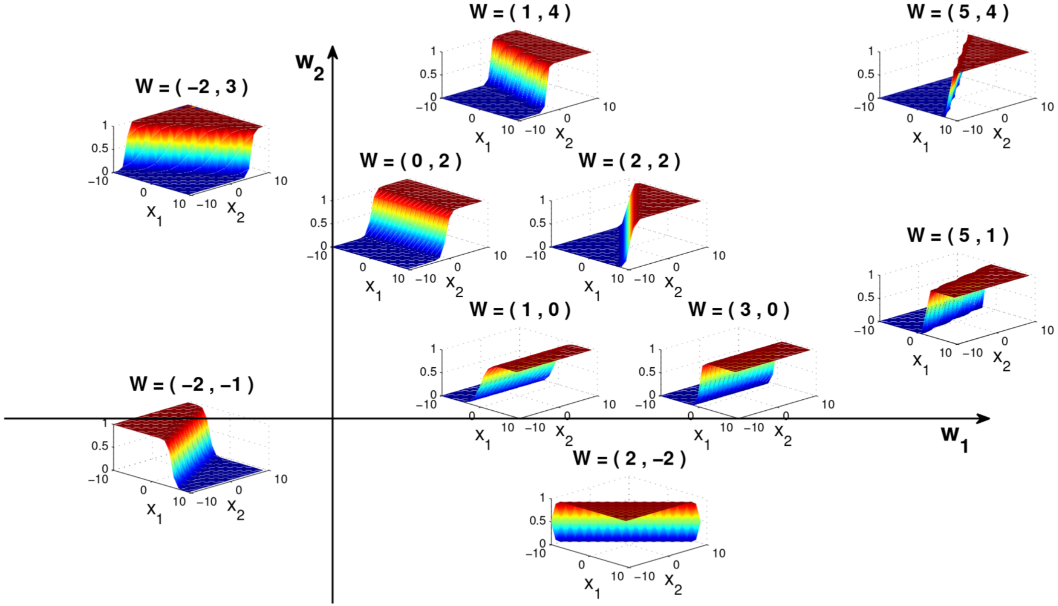
$$f_w(x) = \sigma(w^\top x) = \frac{1}{1+e^{-w^\top x}}$$

Logistic function
(aka Sigmoid fct)
squashing function
activation function

z logit

note the linear decision boundary

Generally, $\sigma(w^T x)$ has a linear decision boundary for any monotonically increasing $\sigma : \mathbb{R} \to \mathbb{R}$



classifiers $\sigma(w_1 x_1 + w_2 x_2)$ for different weights: $w = [w_1, w_2]$

Logistic regression: model

recall the way we included a **bias** parameter $x = [1, x_1]$

the input feature is generated uniformly in [-5,5]
for all the values less than 2 we have y=1 and y=0 otherwise

a good fit to this data is the one shown (green)

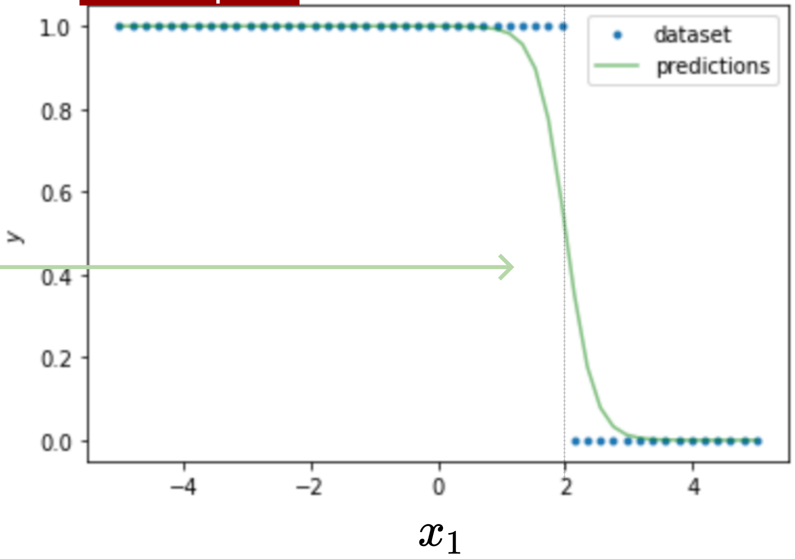
$$f_w(x) = \sigma(w^\top x) = \frac{1}{1+e^{-w^\top x}}$$

in the model shown $w \approx [9.1, -4.5]$

that is $\hat{y} = \sigma(-4.5 x_1 + 9.1)$

what is our model's decision boundary?

example



Logistic regression: The Loss

Maximize MLE <=> Minimize Negative Log Likelihood (NLL):

$$p(y^{(i)}|x^{(i)},\omega) = \begin{cases} \sigma(\omega^T x^{(i)}) \text{ if } y^{(i)} = 1 \\ 1 - \sigma(\omega^T x^{(i)}) \text{ if } y^{(i)} = 0 \end{cases} = \sigma(\omega^T x^{(i)})^{y^{(i)}} \cdot (1 - \sigma(\omega^T x^{(i)}))^{(1-y^{(i)})}$$

Logistic regression: The Loss

Maximize MLE <=> Minimize Negative Log Likelihood (NLL):

$$p(y^{(i)}|x^{(i)},\omega) = \begin{cases} \sigma(\omega^T x^{(i)}) \text{ if } y^{(i)} = 1 \\ 1 - \sigma(\omega^T x^{(i)}) \text{ if } y^{(i)} = 0 \end{cases} = \sigma(\omega^T x^{(i)})^{y^{(i)}} \cdot (1 - \sigma(\omega^T x^{(i)}))^{(1-y^{(i)})}$$

$$L(\mathcal{D},\omega) = p(\mathcal{D}|\omega) = \prod_{x^i,y^i \in \mathcal{D}} \sigma(\omega^T x^{(i)})^{y^{(i)}} \cdot (1 - \sigma(\omega^T x^{(i)}))^{(1-y^{(i)})}$$

Logistic regression: The Loss

Maximize MLE <=> Minimize Negative Log Likelihood (NLL):

$$p(y^{(i)}|x^{(i)},\omega) = \begin{cases} \sigma(\omega^T x^{(i)}) \text{ if } y^{(i)} = 1 \\ 1-\sigma(\omega^T x^{(i)}) \text{ if } y^{(i)} = 0 \end{cases} = \sigma(\omega^T x^{(i)})^{y^{(i)}} \cdot (1-\sigma(\omega^T x^{(i)}))^{(1-y^{(i)})}$$

$$L(\mathcal{D},\omega) = p(\mathcal{D}|\omega) = \prod_{x^i,y^i\in\mathcal{D}} \sigma(\omega^T x^{(i)})^{y^{(i)}} \cdot (1-\sigma(\omega^T x^{(i)}))^{(1-y^{(i)})}$$

$$J(\omega) = -\log\left(L(\mathcal{D},\omega)\right) = -\sum_{x^i,y^i\in\mathcal{D}} y^{(i)} \log\left(\sigma(\omega^T x^{(i)})\right) + (1-y^{(i)}) \log\left(1-\sigma(\omega^T x^{(i)})\right)$$

Logistic regression: The Loss

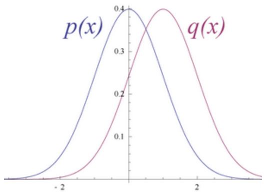
Maximize MLE <=> Minimize Negative Log Likelihood (NLL):

$$p(y^{(i)}|x^{(i)},\omega) = \begin{cases} \sigma(\omega^T x^{(i)}) \text{ if } y^{(i)} = 1 \\ 1-\sigma(\omega^T x^{(i)}) \text{ if } y^{(i)} = 0 \end{cases} = \sigma(\omega^T x^{(i)})^{y^{(i)}} \cdot (1-\sigma(\omega^T x^{(i)}))^{(1-y^{(i)})}$$

$$L(\mathcal{D},\omega) = p(\mathcal{D}|\omega) = \prod_{x^i,y^i \in \mathcal{D}} \sigma(\omega^T x^{(i)})^{y^{(i)}} \cdot (1-\sigma(\omega^T x^{(i)}))^{(1-y^{(i)})}$$

$$J(\omega) = -\log\left(L(\mathcal{D},\omega)\right) = -\sum_{x^i,y^i \in \mathcal{D}} y^{(i)} \log\left(\sigma(\omega^T x^{(i)})\right) + (1-y^{(i)}) \log\left(1-\sigma(\omega^T x^{(i)})\right)$$

Cross-Entropy: $\sum_k -p_k \log(q_k)$



Logistic regression: The Loss

Maximize MLE <=> Minimize Negative Log Likelihood (NLL):

$$J(w) = \sum_{n=1}^{N} -y^{(n)} \log(\sigma(w^\top x^{(n)})) - (1 - y^{(n)}) \log(1 - \sigma(w^\top x^{(n)}))$$

substitute logistic function

$$\log\left(\frac{1}{1+e^{-w^\top x}}\right) = -\log\left(1 + e^{-w^\top x}\right)$$

substitute logistic function

$$\log\left(1 - \frac{1}{1+e^{-w^\top x}}\right) = \log\left(\frac{1}{1+e^{w^\top x}}\right) = -\log\left(1 + e^{w^\top x}\right)$$

$\sigma(x) = 1 - \sigma(-x)$

simplified cost $J(w) = \sum_{n=1}^{N} y^{(n)} \log\left(1 + e^{-w^\top x}\right) + (1 - y^{(n)}) \log\left(1 + e^{w^\top x}\right)$

Logistic regression: The Loss

Maximize MLE <=> Minimize Negative Log Likelihood (NLL):

$$J(w) = \sum_{n=1}^{N} -y^{(n)} \log(\sigma(w^\top x^{(n)})) - (1 - y^{(n)}) \log(1 - \sigma(w^\top x^{(n)}))$$

substitute logistic function

$$\log\left(\frac{1}{1+e^{-w^\top x}}\right) = -\log\left(1 + e^{-w^\top x}\right)$$

substitute logistic function

$$\log\left(1 - \frac{1}{1+e^{-w^\top x}}\right) = \log\left(\frac{1}{1+e^{w^\top x}}\right) = -\log\left(1 + e^{w^\top x}\right)$$

$\sigma(x) = 1 - \sigma(-x)$

simplified cost

$$J(w) = \sum_{n=1}^{N} y^{(n)} \log\left(1 + e^{-w^\top x}\right) + (1 - y^{(n)}) \log\left(1 + e^{w^\top x}\right)$$

$$J(w) = \frac{1}{N} \sum_{n=1}^{N} y^{(n)} \log\left(1 + e^{-w^\top x}\right) + (1 - y^{(n)}) \log\left(1 + e^{w^\top x}\right)$$

Better behaved during optimization because the scale of loss is independent of dataset size optional

Cost function implementation

simplified cost: $J(w) = \sum_{n=1}^{N} y^{(n)} \log\left(1 + e^{-w^\top x}\right) + (1 - y^{(n)}) \log\left(1 + e^{w^\top x}\right)$

```
def cost(w, # D
         x, # N x D
         y  # N
        ):
    z = np.dot(x,w) #N x 1
    J = np.mean( y * np.log1p(np.exp(-z)) + (1-y) * np.log1p(np.exp(z)) )
    return J
```

why not `np.log(1 + np.exp(-z)) ?`

```
In [3]: np.log(1+1e-100)
Out[3]: 0.0
In [4]: np.log1p(1e-100)
Out[4]: 1e-100
```

for small ϵ, $\log(1+\epsilon)$ suffers from floating point inaccuracies

$$\log(1+\epsilon) = \epsilon - \frac{\epsilon^2}{2} + \frac{\epsilon^3}{3} - \dots$$

Example: binary classification

classification on **Iris flowers dataset**:

(a classic dataset originally used by Fisher)

classifying Iris flowers

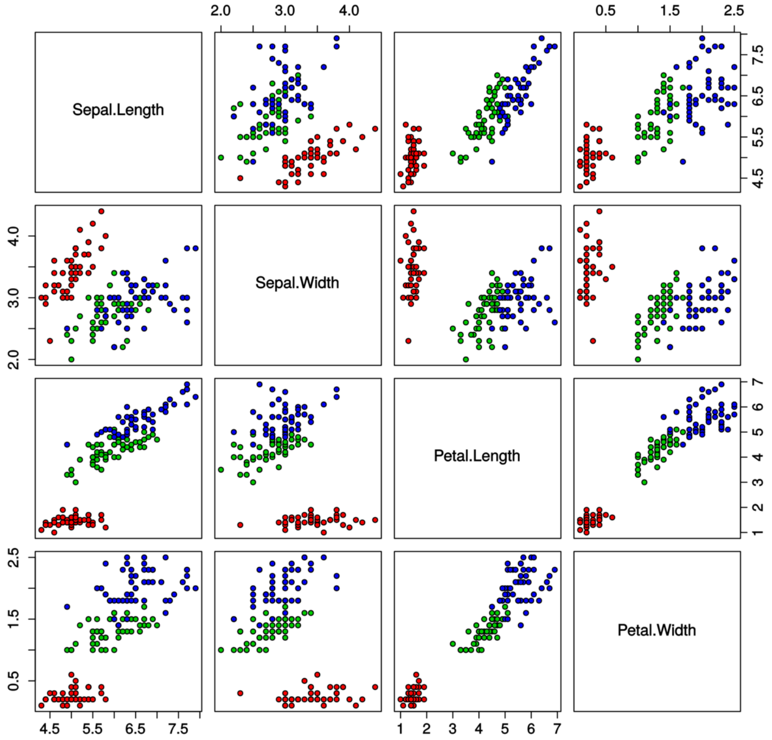
N = 150 instances of flowers

D=4 features {the length and the width of the sepals and petals}

C=3 classes {setosa, versicolor, virginica} : 50 samples of each

index	sl	sw	pl	pw	label
0	5.1	3.5	1.4	0.2	Setosa
1	4.9	3.0	1.4	0.2	Setosa
	...				
50	7.0	3.2	4.7	1.4	Versicolor
	...				
149	5.9	3.0	5.1	1.8	Virginica

Iris Data (red=setosa,green=versicolor,blue=virginica)



Example: binary classification

classification on **Iris flowers dataset**:

(a classic dataset originally used by Fisher)

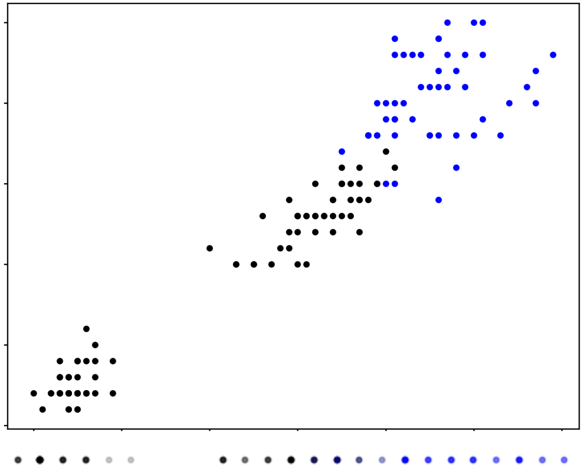
N = 150 instances of flowers

D=4 features {the length and the width of the sepals and petals}

C=3 classes {setosa, versicolor, virginica} : 50 samples of each

$N_c = 50$ samples with D=4 features, for each of C=3 species of Iris flower

$N_1 = 50$ samples with D=1features for one class and $N_0 = 100$ samples with D=1 features for the other class



our setting

2 classes

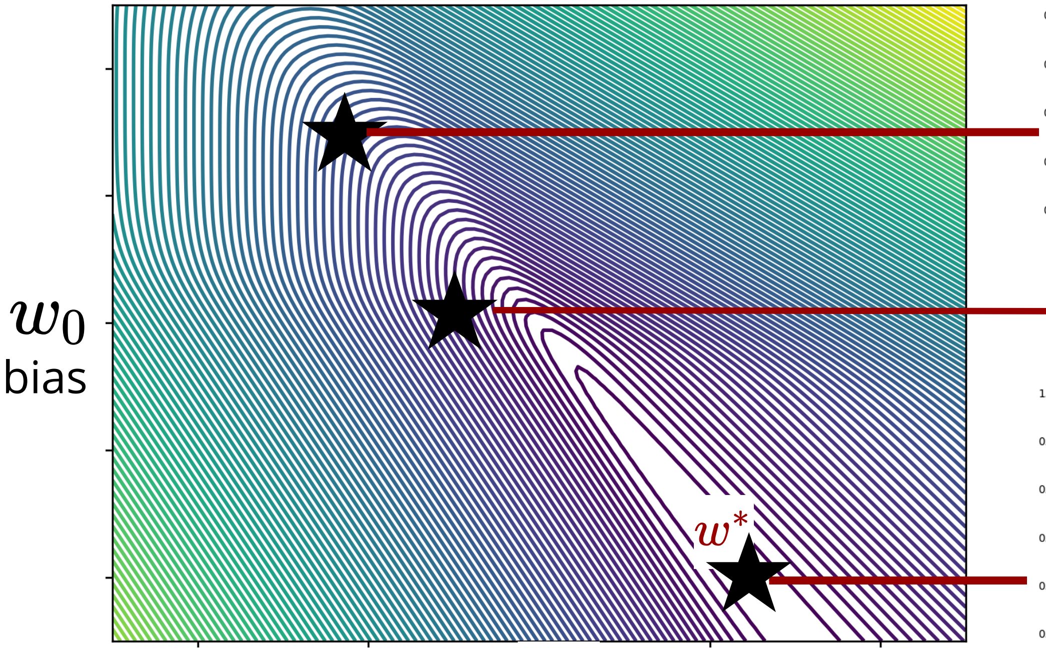
(blue vs others)

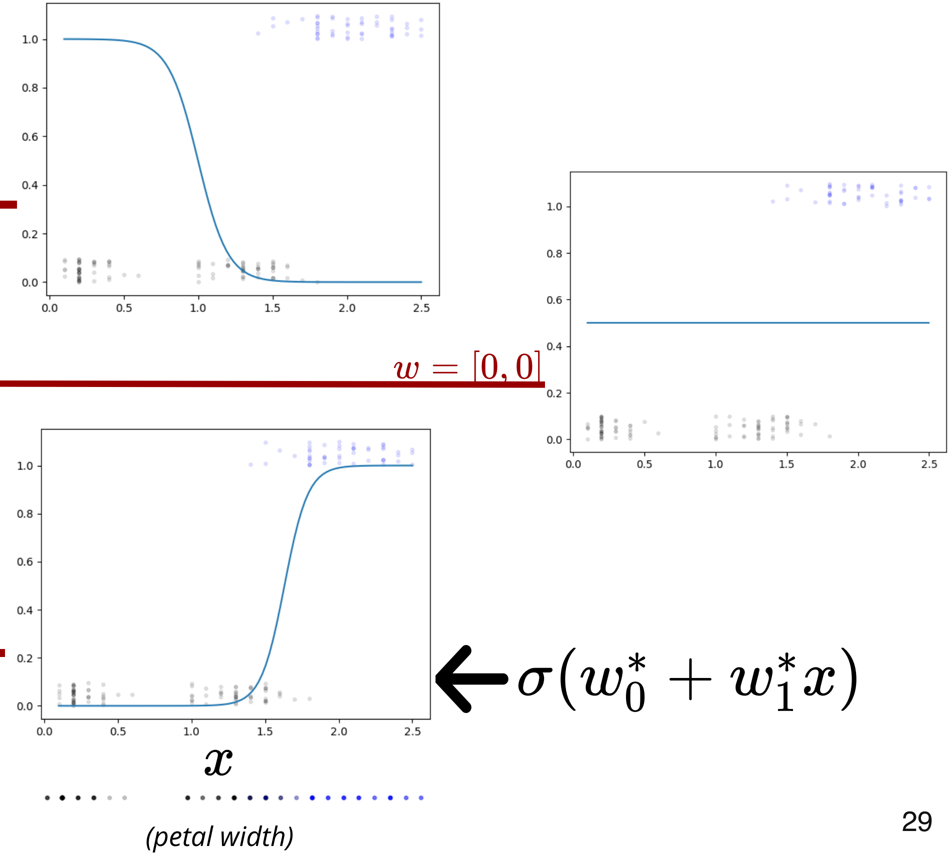
1 features

(petal width + bias)

Example: binary classification

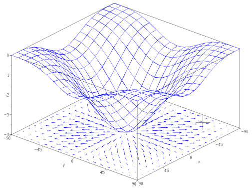
$J(w)$ as a function of these weights





we have two weights associated
with bias + petal width

Gradient



how did we find the optimal weights?

(in contrast to linear regression, no closed form solution)

cost: $J(w) = \sum_{n=1}^{N} y^{(n)} \log\left(1 + e^{-w^\top x^{(n)}}\right) + (1 - y^{(n)}) \log\left(1 + e^{w^\top x^{(n)}}\right)$

taking partial derivative $\frac{\partial}{\partial w_d} J(w) = \sum_n -y^{(n)} x_d^{(n)} \frac{e^{-w^\top x^{(n)}}}{1+e^{-w^\top x^{(n)}}} + x_d^{(n)} (1 - y^{(n)}) \frac{e^{w^\top x^{(n)}}}{1+e^{w^\top x^{(n)}}}$

$= \sum_n -x_d^{(n)} y^{(n)} (1 - \hat{y}^{(n)}) + x_d^{(n)} (1 - y^{(n)}) \hat{y}^{(n)} = \sum_n x_d^{(n)} (\hat{y}^{(n)} - y^{(n)})$

gradient $\nabla J(w) = \sum_n x^{(n)} (\hat{y}^{(n)} - y^{(n)})$

$\hat{y}^{(n)} = \sigma(w^\top x^{(n)})$

compare to gradient for linear regression $\nabla J(w) = \sum_n x^{(n)} (\hat{y}^{(n)} - y^{(n)})$

$\hat{y}^{(n)} = w^\top x^{(n)}$

Multiclass classification

using this probabilistic view we extend logistic regression to multiclass setting

binary classification: Bernoulli likelihood:

$$\text{Bernoulli}(y \mid \hat{y}) = \hat{y}^y (1-\hat{y})^{1-y} \qquad \text{subject to} \qquad \hat{y} \in [0,1]$$

$$\begin{cases} \hat{y} & y=1 \\ 1-\hat{y} & y=0 \end{cases}$$

↓

using logistic function to ensure this $\hat{y} = \sigma(z) = \sigma(w^T x)$

C classes: categorical likelihood

$$\text{Categorical}(y \mid \hat{y}) = \prod_{c=1}^{C} \hat{y}_c^{\mathbb{I}(y=c)} \qquad \text{subject to} \qquad \sum_c \hat{y}_c = 1$$

$$\begin{cases} \hat{y}_1 & y=1 \\ \hat{y}_2 & y=2 \\ \dots & \\ \hat{y}_C & y=C \end{cases}$$

↓

achieved using softmax function

Softmax

generalization of logistic to > 2 classes:

- **logistic**: $\sigma : \mathbb{R} \to (0,1)$ produces a single probability
  - probability of the second class is $(1 - \sigma(z))$
- **softmax:** $\mathbb{R}^C \to \Delta_C$ recall: probability simplex $p \in \Delta_c \to \sum_{c=1}^{C} p_c = 1$

$$\hat{y}_c = \text{softmax}(z)_c = \frac{e^{z_c}}{\sum_{c'=1}^{C} e^{z_{c'}}} \quad \text{so} \quad \sum_c \hat{y} = 1$$

example $\text{softmax}([1,1,2,0]) = [\frac{e}{2e+e^2+1}, \frac{e}{2e+e^2+1}, \frac{e^2}{2e+e^2+1}, \frac{1}{2e+e^2+1}]$

$$\text{softmax}([10,100,-1]) \approx [0,1,0]$$

if input values are large, softmax becomes similar to argmax

similar to logistic this is also a squashing function

Multiclass classification

C classes: categorical likelihood

$\mathrm{Categorical}(y \mid \hat{y}) = \prod_{c=1}^{C} \hat{y}_c^{\mathbb{I}(y=c)}$ using softmax to enforce sum-to-one constraint

$$\hat{y}_c = \mathrm{softmax}([{w_1}^\top x, \ldots, {w_C}^\top x])_c = \frac{e^{{w_c}^\top x}}{\sum_{c'} e^{{w_{c'}}^\top x}}$$

so we have on parameter **vector** for each class

$$w_1 = [w_{1,1}, w_{1,2}, \ldots w_{1,D}]$$

to simplify equations we write $z_c = {w_c}^\top x$

$$\hat{y}_c = \mathrm{softmax}([z_1, \ldots, z_C])_c = \frac{e^{z_c}}{\sum_{c'} e^{z_{c'}}}$$

Likelihood for multiclass classification

C classes: categorical likelihood

$$\text{Categorical}(y \mid \hat{y}) = \prod_{c=1}^{C} \hat{y}_c^{\mathbb{I}(y=c)}$$ using softmax to enforce sum-to-one constraint

$$\hat{y}_c = \text{softmax}([z_1, \ldots, z_C])_c = \frac{e^{z_c}}{\sum_{c'} e^{z_{c'}}}$$ where $z_c = {w_c}^\top x$

substituting softmax in Categorical likelihood:

likelihood

$$L(\{w_c\}) = \prod_{n=1}^{N} \prod_{c=1}^{C} \text{softmax}([z_1^{(n)}, \ldots, z_C^{(n)}])_c^{\mathbb{I}(y^{(n)}=c)}$$

$$= \prod_{n=1}^{N} \prod_{c=1}^{C} \left(\frac{e^{z_c^{(n)}}}{\sum_{c'} e^{z_{c'}^{(n)}}} \right)^{\mathbb{I}(y^{(n)}=c)}$$

One-hot encoding

likelihood

$$L(\{w_c\}) = \prod_{n=1}^{N} \prod_{c=1}^{C} \left(\frac{e^{z_c^{(n)}}}{\sum_{c'} e^{z_{c'}^{(n)}}} \right)^{\mathbb{I}(y^{(n)}=c)}$$

log-likelihood

$$\ell(\{w_c\}) = \sum_{n=1}^{N} \sum_{c=1}^{C} \mathbb{I}(y^{(n)} = c)(z_c^{(n)} - \log \sum_{c'} e^{z_{c'}^{(n)}})$$

one-hot encoding for labels $y^{(n)} \rightarrow [\mathbb{I}(y^{(n)} = 1), \ldots, \mathbb{I}(y^{(n)} = C)]$

convert that feature into C binary features

Example: $y^{(n)} \in \{1, 2, 3\} \Rightarrow y^{(n)} \in \{[1, 0, 0], [0, 1, 0], [0, 0, 1]\}$

side note

we can also use this encoding for categorical **features**

$$x_d^{(n)} \rightarrow [\mathbb{I}(x_d^{(n)} = 1), \ldots, \mathbb{I}(x_d^{(n)} = C)]$$

Food Name	Categorical #	Calories
Apple	1	95
Chicken	2	231
Broccoli	3	50

Apple	Chicken	Broccoli	Calories
1	0	0	95
0	1	0	231
0	0	1	50

one-hot example from here

One-hot encoding

likelihood $L(\{w_c\}) = \prod_{n=1}^{N} \prod_{c=1}^{C} \left(\frac{e^{z_c^{(n)}}}{\sum_{c'} e^{z_{c'}^{(n)}}} \right)^{\mathbb{I}(y^{(n)}=c)}$

log-likelihood $\ell(\{w_c\}) = \sum_{n=1}^{N} \sum_{c=1}^{C} \mathbb{I}(y^{(n)} = c)(z_c^{(n)} - \log \sum_{c'} e^{z_{c'}^{(n)}})$

one-hot encoding for labels $y^{(n)} \rightarrow [\mathbb{I}(y^{(n)} = 1), \ldots, \mathbb{I}(y^{(n)} = C)]$

$z^{(n)} = [z_1^{(n)}, z_2^{(n)}, \ldots z_C^{(n)}]$, $z_c^{(n)} = w_c^\top x^{(n)}$

using this encoding from now on

log-likelihood $\ell(\{w_c\}) = \sum_{n=1}^{N} \left({y^{(n)}}^\top z^{(n)} - \log \sum_{c'} e^{z_{c'}^{(n)}} \right)$

Implementing the cost function

softmax cross entropy cost function is the negative of the log-likelihood

similar to the binary case

$$J(\{w_c\}) = -\left(\sum_{n=1}^{N} ({y^{(n)}}^\top z^{(n)} - \log \sum_{c'} e^{z_{c'}^{(n)}})\right) \quad \text{where } z_c = {w_c}^\top x$$

naive implementation of log-sum-exp causes over/underflow

we could run into very large or small numbers inside the exponential

prevent this using this one trick!

$$\log \sum_c e^{z_c} = \bar{z} + \log \sum_c e^{z_c - \bar{z}}$$

where $\bar{z} \leftarrow \max_c z_c$

this bring the numbers in exponent close to zero and makes the log-sum-exp numerically stable

Optimization

given the training data $\mathcal{D} = \{(x^{(n)}, y^{(n)})\}_n$

find the best model parameters $\{w_c\}_c$

by minimizing the cost (maximizing the likelihood of $\mathcal{D}$)

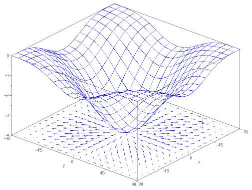
$$J(\{w_c\}) = -\sum_{n=1}^{N} (y^{(n)^\top} z^{(n)} - \log \sum_{c'} e^{z_{c'}^{(n)}}) \text{ where } z_c = w_c{}^\top x$$

need to use gradient descent (for now calculate the gradient)

$$\nabla J(w) = [\frac{\partial}{\partial w_{1,1}} J, \ldots \frac{\partial}{\partial w_{1,D}} J, \ldots, \frac{\partial}{\partial w_{C,D}} J]^\top$$

length $C \times D$

Gradient



need to use gradient descent (for now calculate the gradient)

$$J(\{w_c\}) = -\sum_{n=1}^{N} \left({y^{(n)}}^\top z^{(n)} - \log \sum_{c'} e^{z_{c'}^{(n)}} \right) \quad \text{where } z_c = {w_c}^\top x$$

using chain rule

$$\frac{\partial}{\partial w_{c,d}} J = \sum_{n=1}^{N} \frac{\partial J}{\partial z_c^{(n)}} \frac{\partial z_c^{(n)}}{\partial w_{c,d}} = \sum_n (\hat{y}_c^{(n)} - y_c^{(n)}) x_d^{(n)}$$

this looks familiar!

$$\frac{\partial z_c^{(n)}}{\partial w_{c,d}} \rightarrow x_d^{(n)}$$

$$\frac{\partial J}{\partial z_c^{(n)}} \rightarrow -y_c^{(n)} + \frac{e^{z_c^{(n)}}}{\sum_{c'} e^{{z^{(n)}}_{c'}}}$$

so the derivative of log-sum-exp is softmax

$$\frac{e^{z_c^{(n)}}}{\sum_{c'} e^{{z^{(n)}}_{c'}}} \rightarrow \hat{y}_c^{(n)}$$

Summary

- logistic regression: logistic activation function + cross-entropy loss
  - cost function
  - probabilistic interpretation
    - using maximum likelihood to derive the cost function

Gaussian likelihood	$\Leftrightarrow$	L2 loss
Bernoulli likelihood		cross-entropy loss

- multi-class classification: softmax + cross-entropy
  - cost function
  - one-hot encoding
  - gradient calculation (will use later!)