Applied Machine Learning

Logistic and Softmax Regression

Reihaneh Rabbany

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

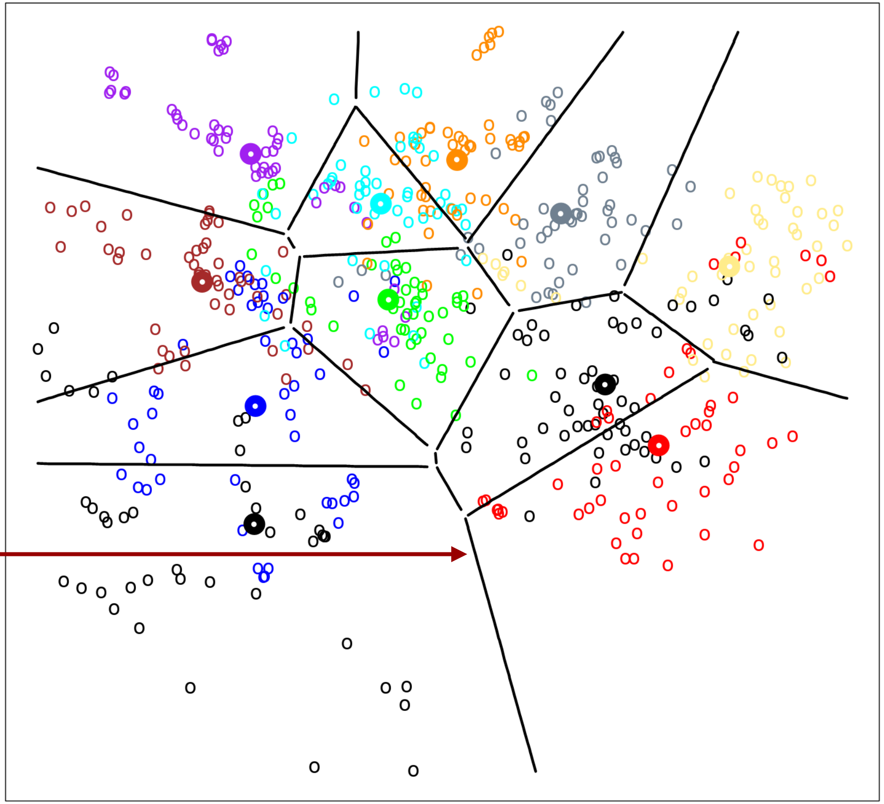
binary classification $y^{(n)} \in \{0, 1\}$

linear classification:

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

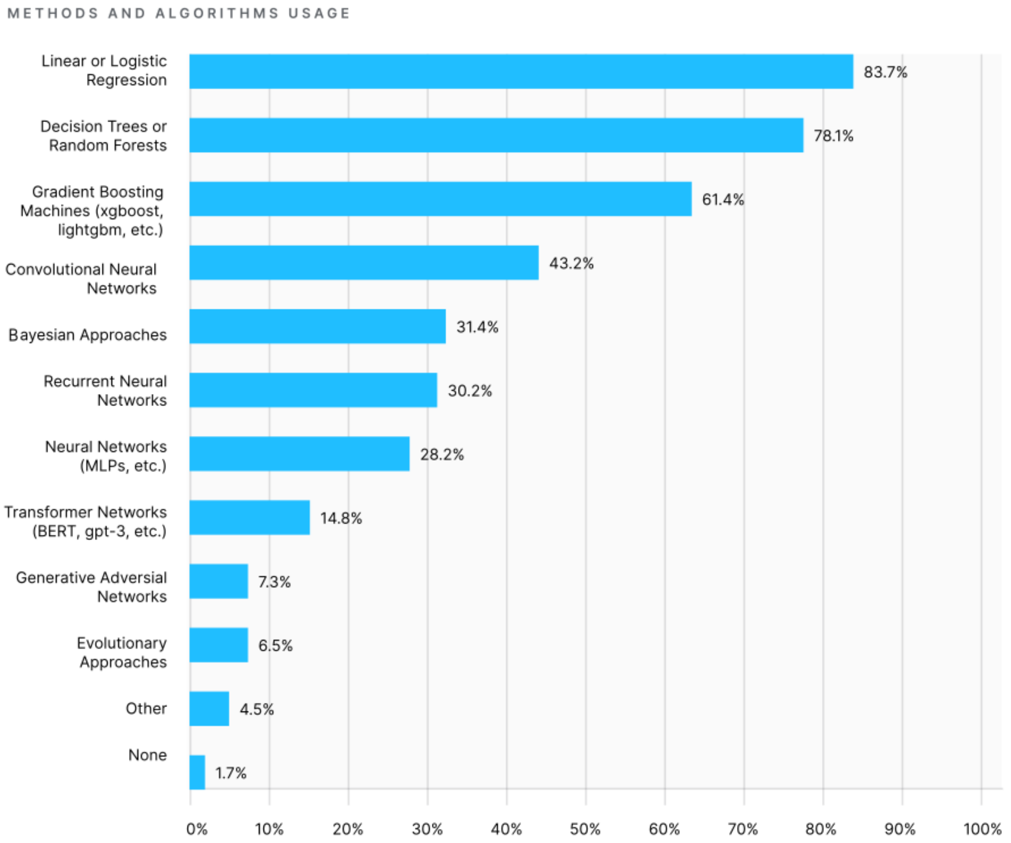
how do we find these boundaries?

different approaches give different linear classifiers



Motivation

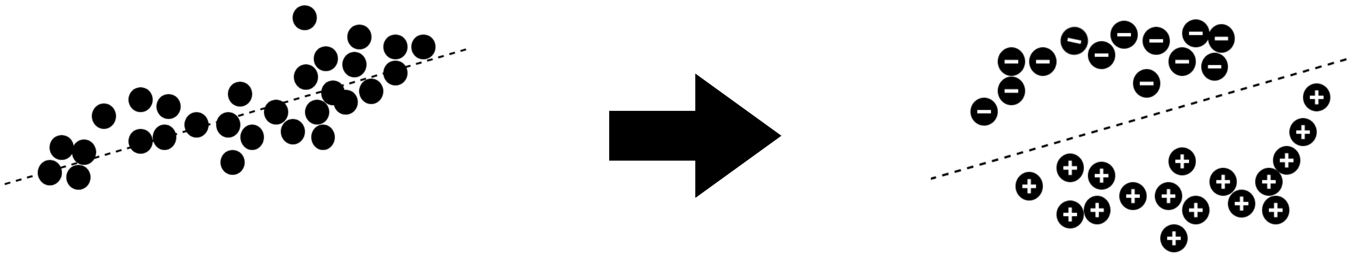
Logistic Regression is **the** most commonly reported data science method used in practice



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?

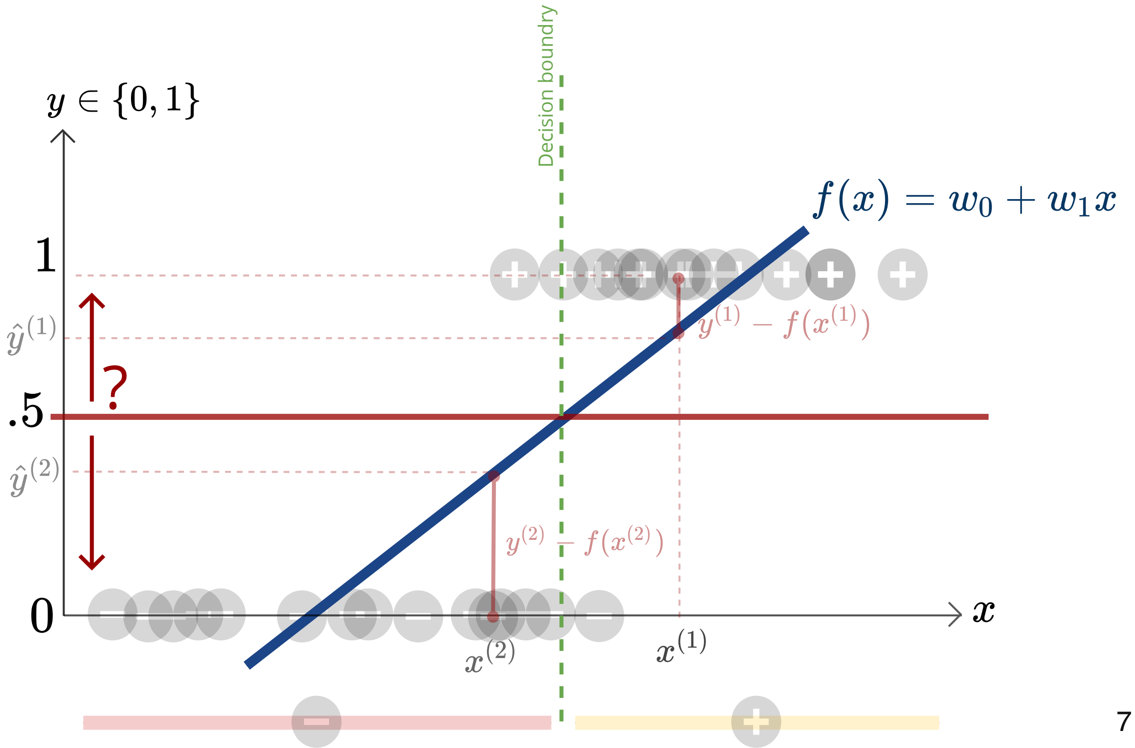
Use 1 and 0 as the target value directly apply linear regression

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?

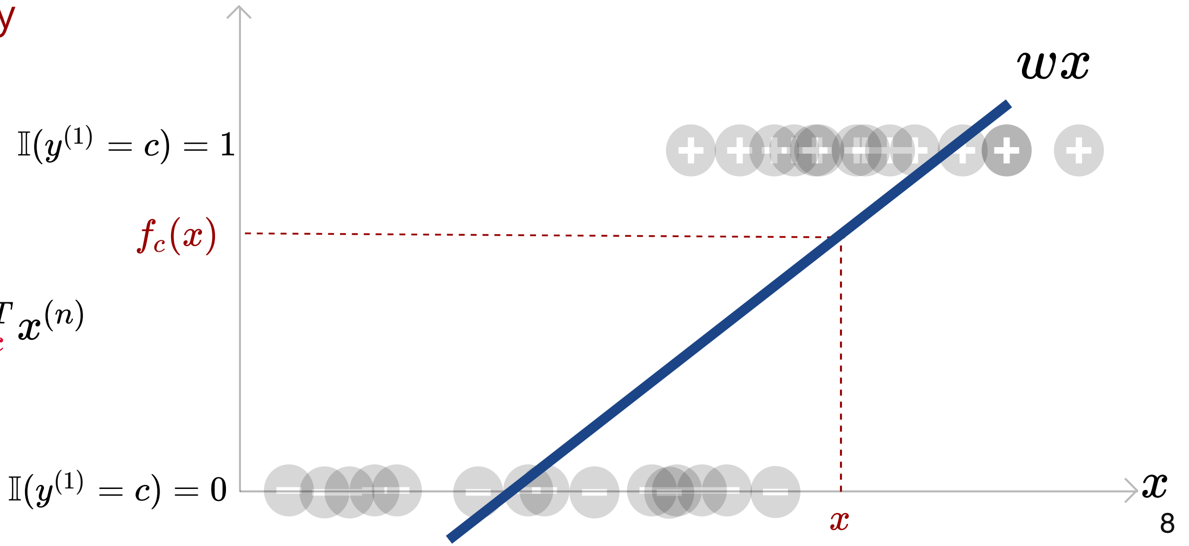
more than one class? $y \in \{0, 1, \ldots, C\}$

fit a linear model to each class: $w_c^* = \arg\min_{w_c} \frac{1}{2} \sum_{n=1}^{N} (w_c^T x^{(n)} - \mathbb{I}(y^{(n)} = c))^2$

Use 1 and 0 as the target value directly
apply linear regression, only one class
maps to one, all other to zero

How to get the output class?

$\hat{y}^{(n)} = \arg\max_c f_c(x) = \arg\max_c w_c^T x^{(n)}$



Linear regression for classification?

first idea adapting linear regression to do classification?

more than one class? $y \in \{0, 1, \dots, C\}$

fit a linear model to each class: $w_c^* = \arg\min_{w_c} \frac{1}{2} \sum_{n=1}^{N} (w_c^T x^{(n)} - \mathbb{I}(y^{(n)} = c))^2$

Use 1 and 0 as the target value directly
apply linear regression, only one class
maps to one, all other to zero

How to get the output class?

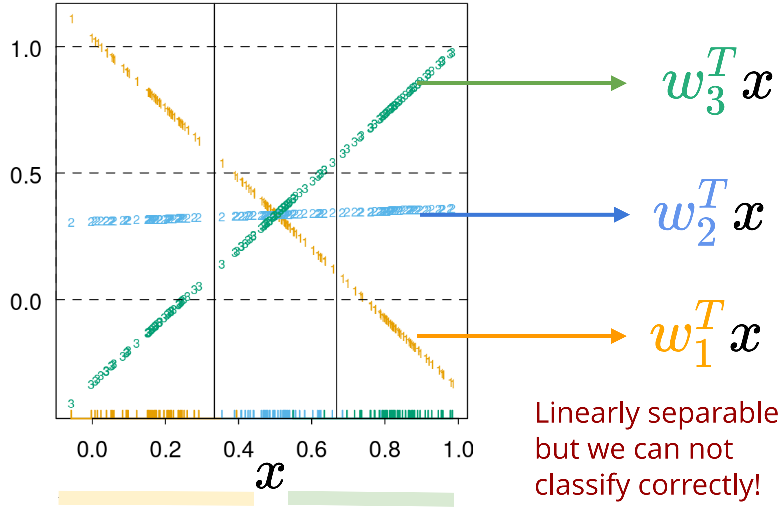
$$\hat{y}^{(n)} = \arg\max_c f_c(x) = \arg\max_c w_c^T x^{(n)}$$

decision boundary between any two classes

$$w_c^T x = w_{c'}^T x$$

$$(w_c - w_{c'})^T x = 0$$

Example:



Linearly separable but we can not classify correctly!

Linear regression for classification?

first idea adapting linear regression to do classification?

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

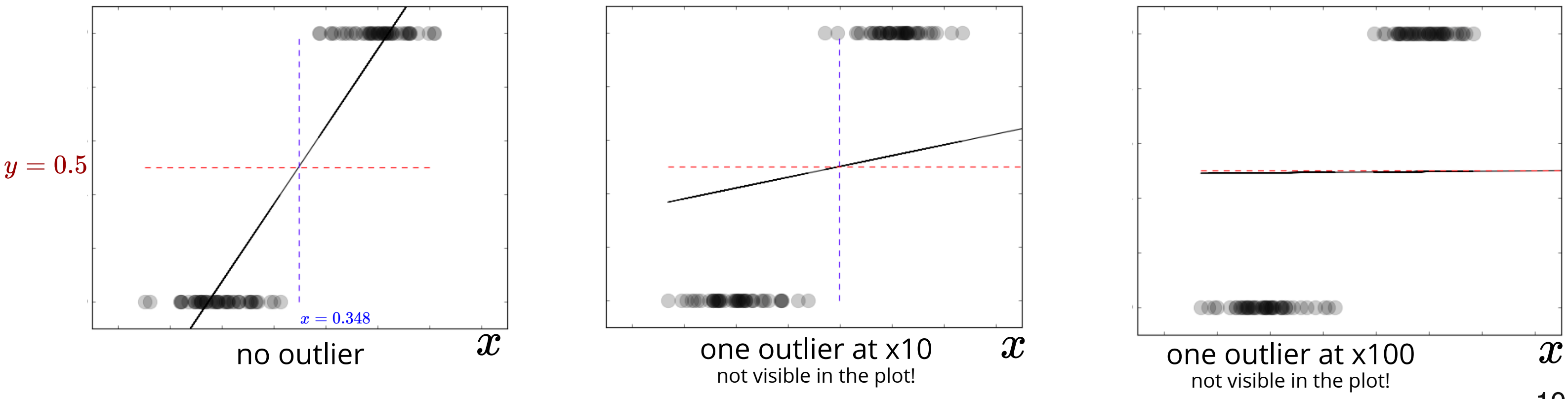
Sensitivity to Outliers, which can dominate the L2 loss (sum of least squares)

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

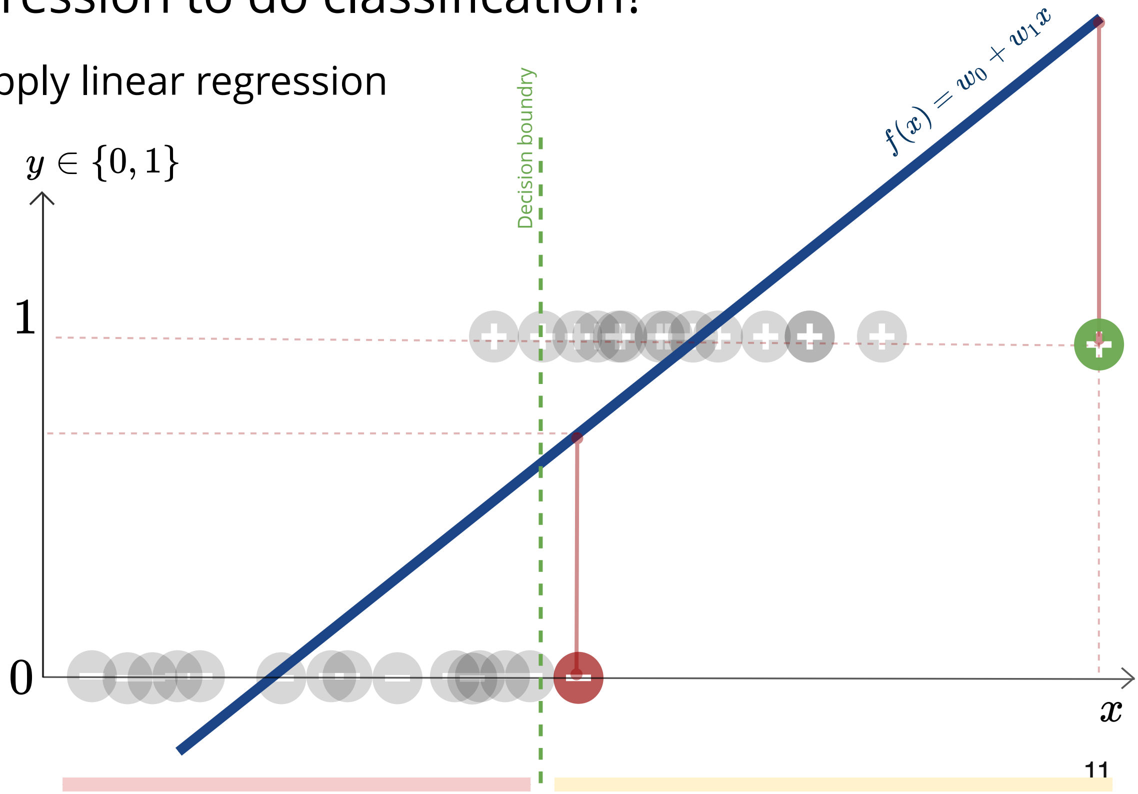


Linear regression for classification?

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!



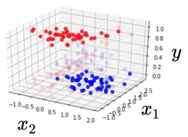
Linear regression for classification?

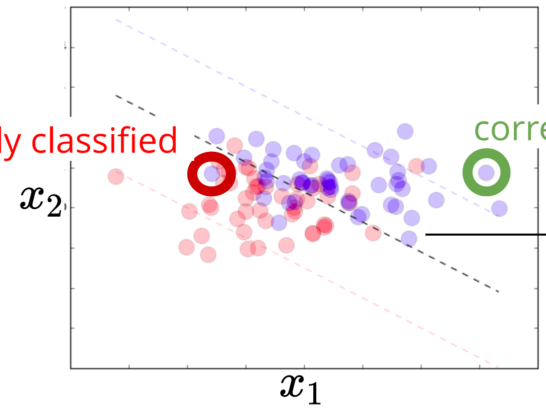
first idea adapting linear regression to do classification?

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

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

Example: D=2
Binary classification





$$w_0{}^T x = w_1{}^T x$$

$$(w_1 - w_0)^T x = 0$$

so one weight vector is enough $w^T x > 0 \Rightarrow y = 1$

$$\begin{cases} w^T x^{(n)} = 100, y^{(n)} = 1 & \Rightarrow (100-1)^2 = 99^2 \\ w^T x^{(n)} = -2, y^{(n)} = 1 & \Rightarrow (-2-1)^2 = 9 \end{cases}$$

correct prediction has higher loss than the incorrect one! ☹

Logistic function

Idea: apply a squashing function to $w^\top x \rightarrow \sigma(w^\top x)$

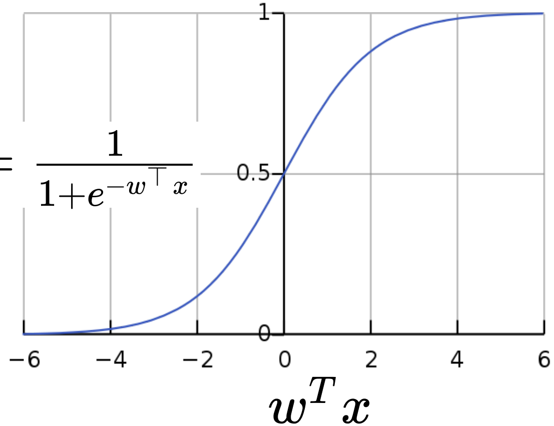
desirable property of $\sigma : \mathbb{R} \rightarrow \mathbb{R}$

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

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

logistic function 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

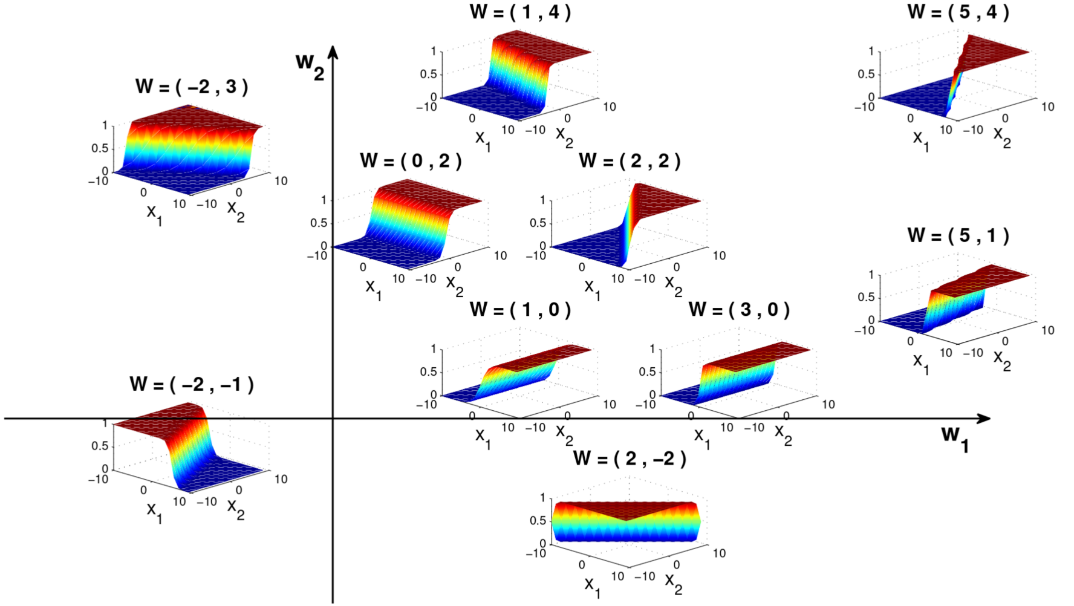
$$f_w(x) = \sigma(w^\top x) = \frac{1}{1+e^{-w^\top x}}$$

logistic function
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)

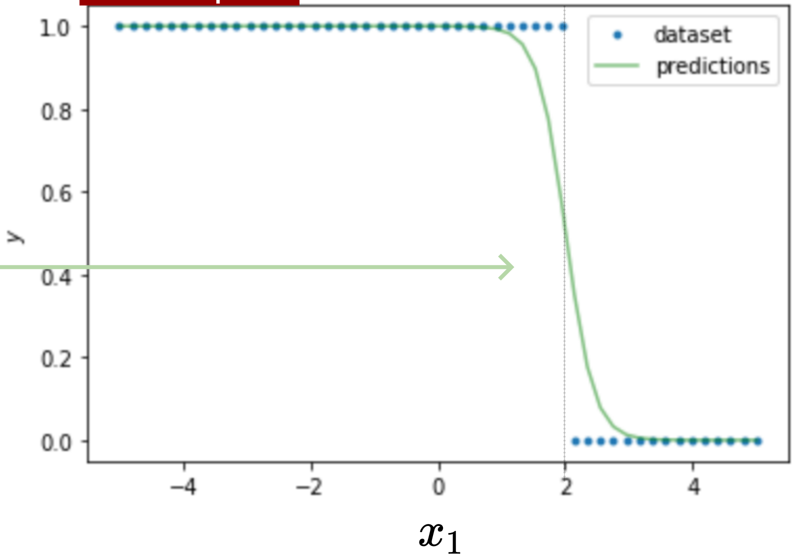
$$f_w(x) = \sigma(w^\top x) = \frac{1}{1+e^{-w^\top x}}$$

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

to find a good model, we need to define the cost (loss) function

the best model is the one with the lowest cost

cost is the some of loss values for individual points

zero-one loss

misclassification error

$$L_{0/1}(\hat{y}, y) = \mathbb{I}(y \neq \hat{y})$$

$\sigma(w^\top x)$

- not a continuous function (in w)
- hard to optimize

L2 loss

least squared error

$$L_2(\hat{y}, y) = \frac{1}{2}(y - \hat{y})^2$$

$\sigma(w^\top x)$

squashing resolves some problems and loss is continuous but

- hard to optimize (non-convex in w)

Logistic regression: the loss

third idea use the cross-entropy loss

$$L_{CE}(\hat{y}, y) = -y\log(\hat{y}) - (1-y)\log(1-\hat{y})$$

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

- it is convex in w
- probabilistic interpretation (soon!) ☺

examples

correctly classified ○ $L_{CE}(y=1, \hat{y}=.9) = -\log(.9) \approx 0.1$ smaller than $L_{CE}(y=1, \hat{y}=.5) = -\log(.5) = 0.69$

incorrectly classified ○ $L_{CE}(y=0, \hat{y}=.9) = -\log(.1) \approx 2.3$ larger than $L_{CE}(y=0, \hat{y}=.5) = -\log(.5) = 0.69$

optional

Cost function

we need to optimize the cost wrt. parameters

first: simplify $L_{CE}(y, \hat{y}) = -y \log(\hat{y}) - (1-y)\log(1-\hat{y})$

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

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

substitute logistic function

substitute logistic function

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

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

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

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

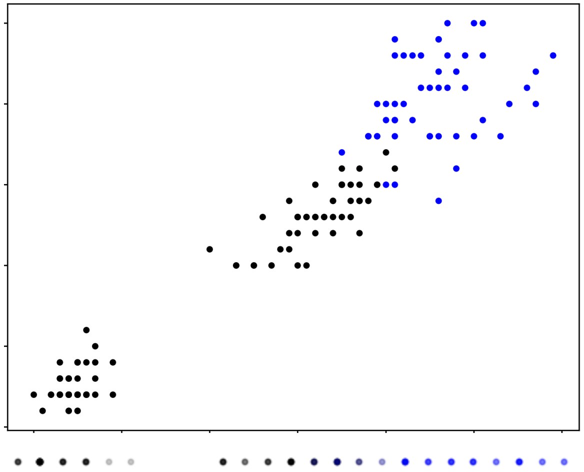
$$\log(1+\epsilon) = \epsilon - \frac{x^2}{2} + \frac{x^3}{3} - \dots$$

Example: binary classification

classification on **Iris flowers dataset**:

(a classic dataset originally used by Fisher)

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



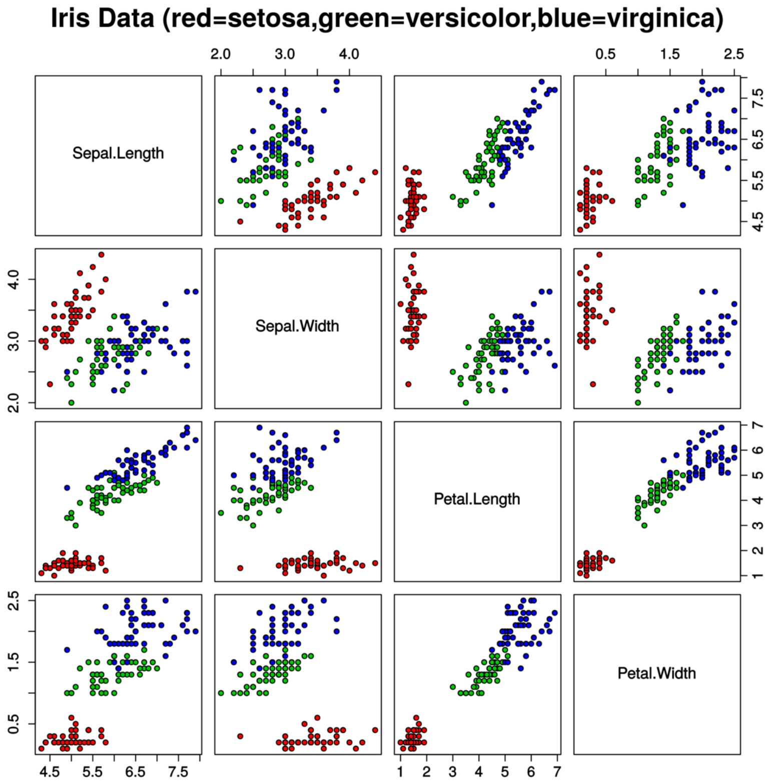
our setting

2 classes

(blue vs others)

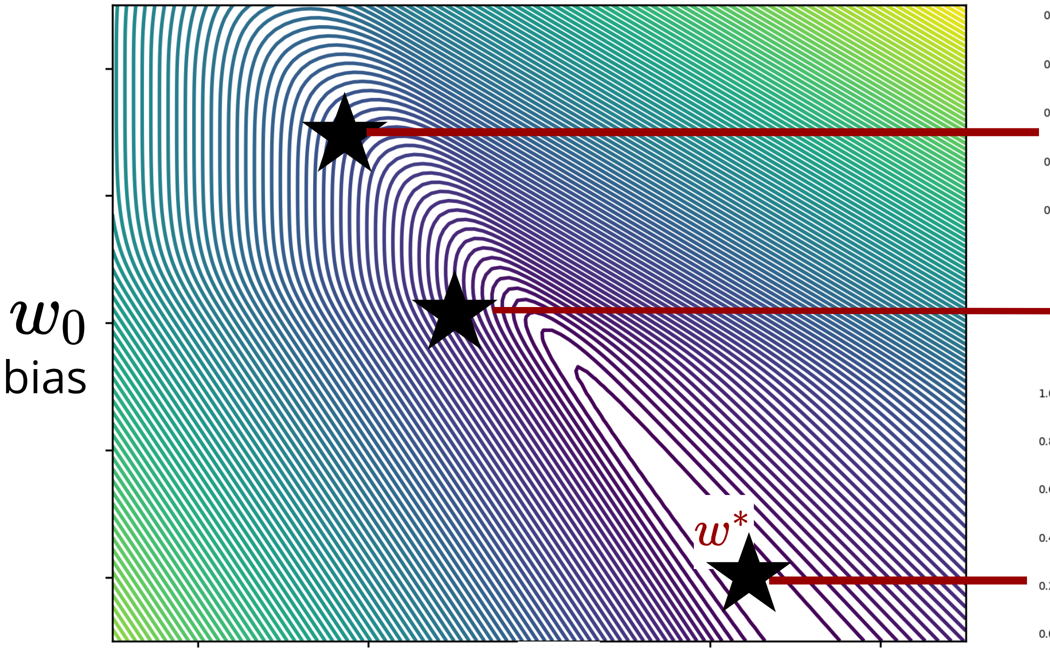
1 features

(petal width + bias)



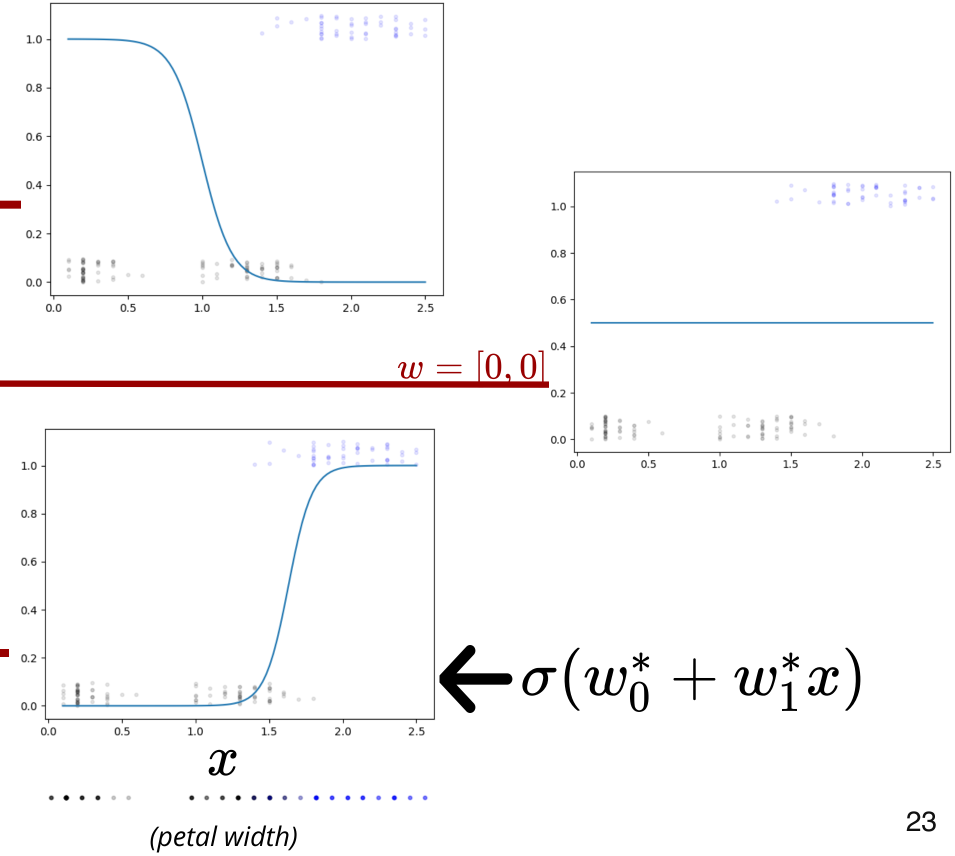
Example: binary classification

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

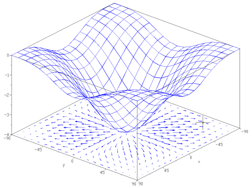


w_1

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 $\nabla J(w) = \sum_n x^{(n)} (\underbrace{\hat{y}^{(n)}}_{\sigma(w^\top x^{(n)})} - y^{(n)})$

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

Probabilistic view

Interpret the prediction as class probability $\hat{y} = p_w(y = 1 \mid x) = \sigma(w^\top x)$

the log-ratio of class probabilities is linear

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

logit function
is the inverse of logistic

so we have a Bernoulli likelihood

$$p(y^{(n)} \mid x^{(n)}; w) = \mathrm{Bernoulli}(y^{(n)}; \sigma(w^\top x^{(n)})) = {\hat{y}^{(n)}}^{y^{(n)}} (1 - \hat{y}^{(n)})^{1-y^{(n)}}$$

conditional likelihood of the labels given the inputs

$$L(w) = \prod_{n=1}^{N} p(y^{(n)} \mid x^{(n)}; w) = \prod_{n=1}^{N} {\hat{y}^{(n)}}^{y^{(n)}} (1 - \hat{y}^{(n)})^{1-y^{(n)}}$$

Maximum likelihood & logistic regression

likelihood $L(w) = \prod_{n=1}^{N} p(y^{(n)} \mid x^{(n)}; w) = \prod_{n=1}^{N} \hat{y}^{(n)^{y^{(n)}}} (1 - \hat{y}^{(n)})^{1-y^{(n)}}$

find w that maximizes log likelihood

$$
\begin{aligned}
w^* &= \max \sum_{n=1}^{N} \log p_w(y^{(n)} \mid x^{(n)}; w) \\
&= \max_w \sum_{n=1}^{N} y^{(n)} \log(\hat{y}^{(n)}) + (1 - y^{(n)}) \log(1 - \hat{y}^{(n)}) \\
&= \min_w J(w)
\end{aligned}
$$

the cross entropy cost function!

so using cross-entropy loss in logistic regression is maximizing conditional likelihood

we saw a similar interpretation for linear regression (L2 loss maximizes the conditional Gaussian likelihood)

Multiclass classification

using this probabilistic view we extend logistic regression to multiclass setting

binary classification: Bernoulli likelihood:

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

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

↓

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

C classes: categorical likelihood

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

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

↓

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:** $\mathbb{R}^C \to \boxed{\Delta_C}$ 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

so similar to logistic this is also a squashing function

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

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

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 $z_c = w_c{}^\top x$

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

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

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

using this encoding from now on

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

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

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

Implementing the cost function

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

similar to the binary case

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

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

prevent this using this one trick!

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

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

Optimization

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

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

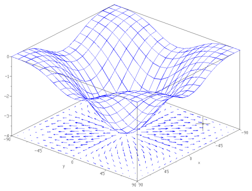
by minimizing the cost (maximizing the likelihood of $\mathcal{D}$)

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

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

$$\nabla J(w) = \underbrace{\left[\frac{\partial}{\partial w_{1,1}} J, \ldots \frac{\partial}{\partial w_{1,D}} J, \ldots, \frac{\partial}{\partial w_{C,D}} J\right]^\top}_{\text{length } C \times D}$$

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

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

this looks familiar!

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

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

so the derivative of log-sum-exp is softmax

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

Discreminative vs. generative classification

naive Bayes

learns the **joint** distribution

$$p(y, x) = p(y)p(x \mid y)$$

the max-likelihood estimate of prior and likelihood has closed-form solution

(using empirical frequencies)

makes stronger assumptions

usually works better with smaller datasets

linear decision boundary for Gaussian naive Bayes only **if** the variance is fixed

logistic regression

learns the **conditional** distribution

$$p(y \mid x)$$

no closed-form solution

(use numerical optimization)

weaker assumptions, since it doesn't model the distribution of input (x)

usually works better with larger datasets

linear decision boundary

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!)