BROWN
Computer Science

CS1951A: Data Science

Lecture 16: Supervised learning Classification

Lorenzo De Stefani

Spring 2022

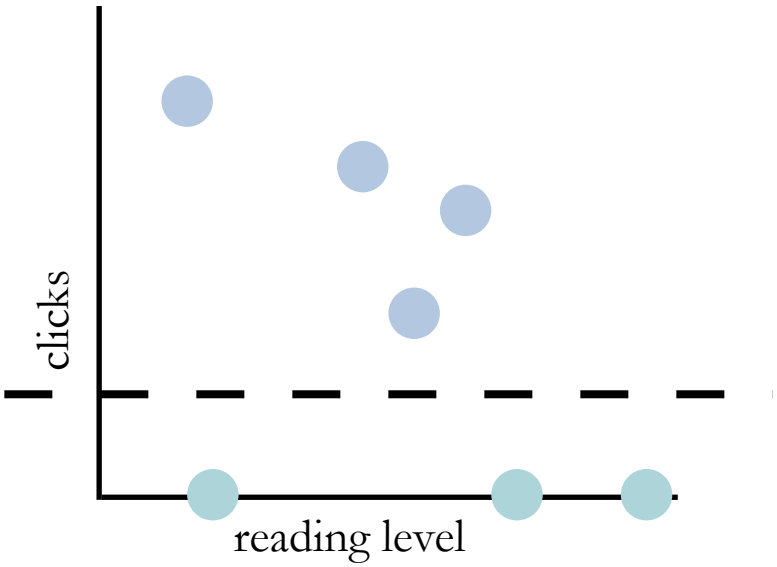
Today

- Supervised learning: classification and regression
- k Nearest Neighbors
- Generative vs. Discriminative Models
- Naïve Bayes
- Linear regression with gradient descent
- Logistic Regression

Supervised vs. Unsupervised Learning

- Supervised: Explicit data labels
  - Sentiment analysis—review text -> star ratings
  - Image tagging—image -> caption
- Unsupervised: No explicit labels
  - Clustering—find groups similar customers
  - Dimensionality Reduction—find features that differentiate individuals

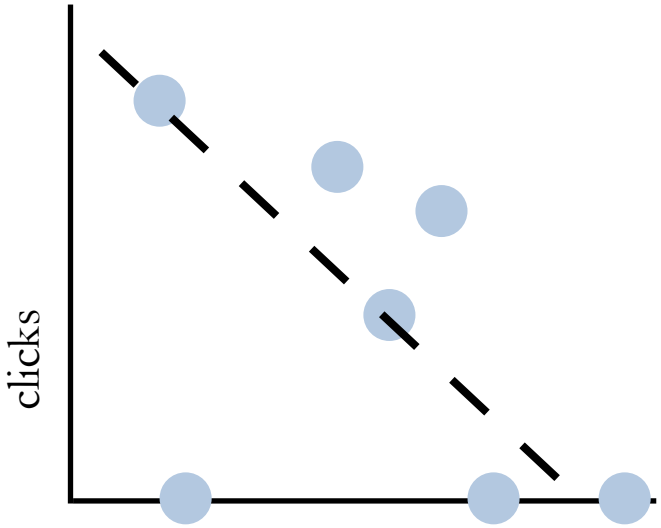
Classification and Regression



The predictor partitions the points in classes

- Assigns a "label" associate with the class
  - Discrete output
- Binary classification with two classes
  - E.g., "clicked, not clicked"

$f(reading\ level) = \{clicked, not\ clicked\}$



The predictor provides an estimate of the value of interest

- Returns real values
- $clicks = m(reading\_level) + b$
- m and b are the parameters of the model to be estimated

Classification: a probabilistic interpretation

We want to learn a predictor for the label Y based on observations of observed values X:

- We can study the distribution of the labels given the values X

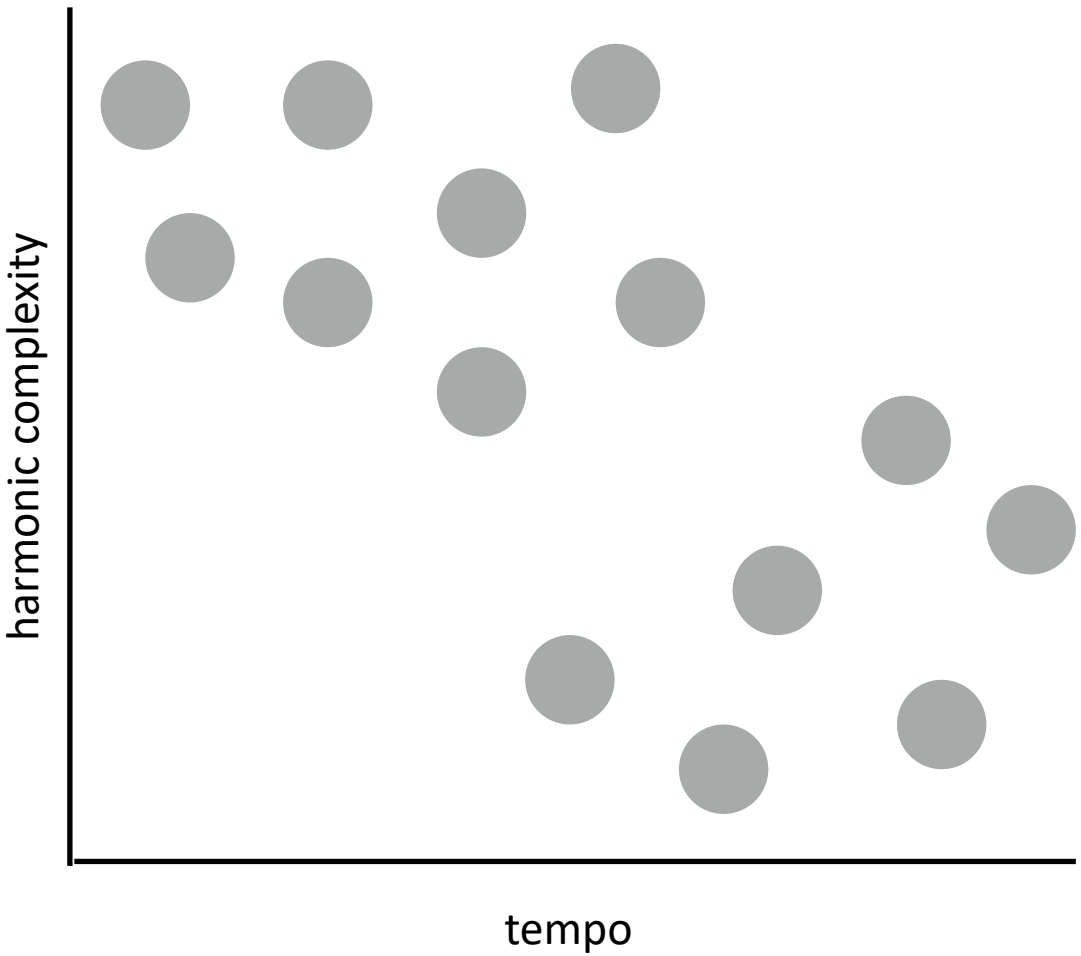
$$P(Y|X)$$

Label

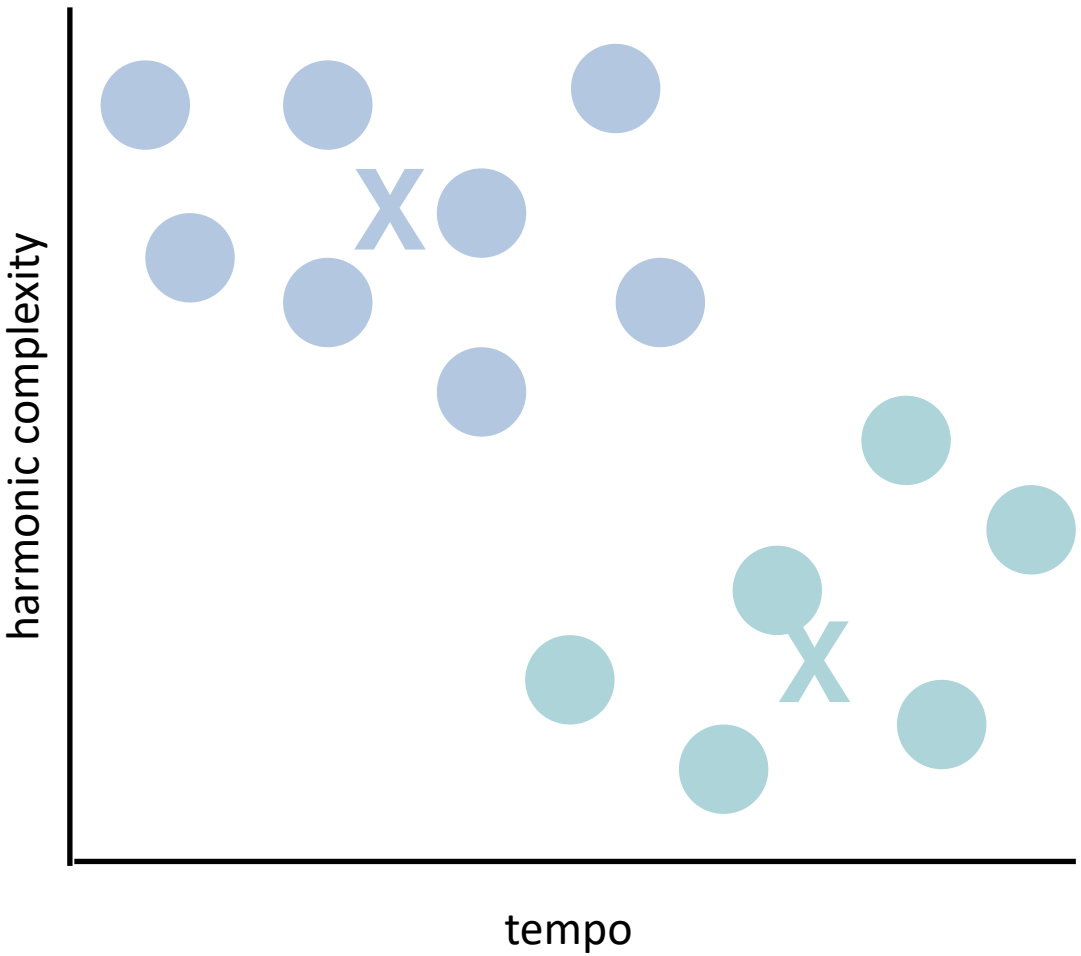
Features

- P(email is spam | words in the message)
- P(genre of song|tempo, harmony, lyrics…)
- P(article clicked | title, font, photo…)

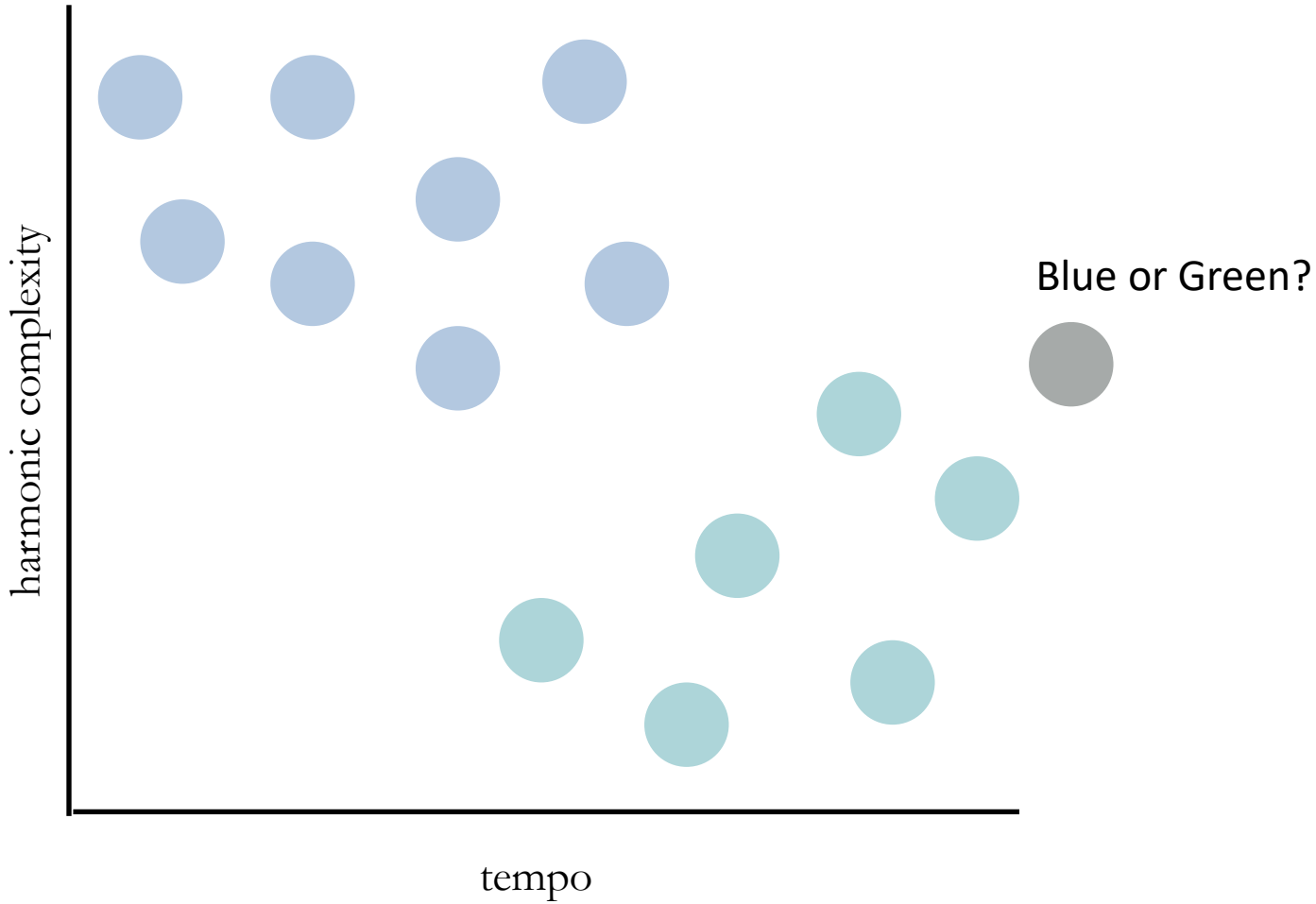
K Means



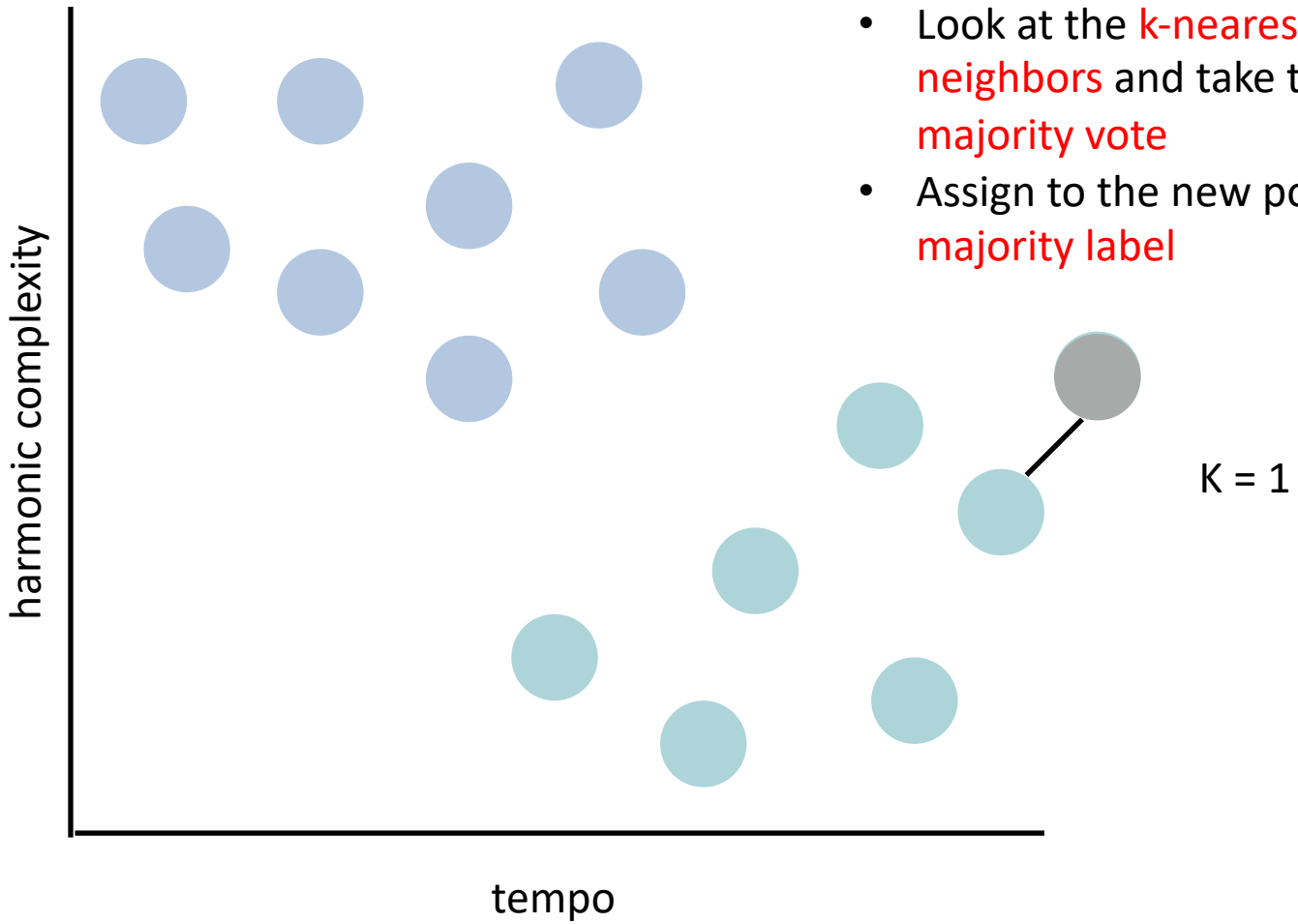
K Means



K Nearest Neighbors

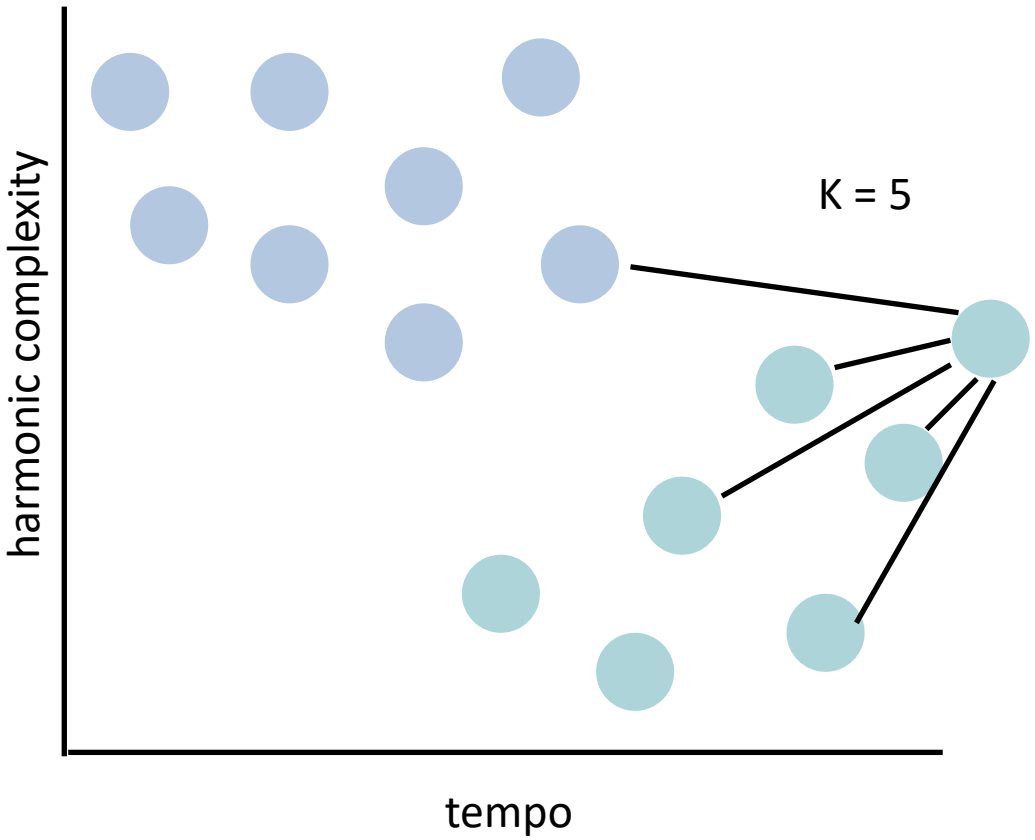


K Nearest Neighbors - Classification



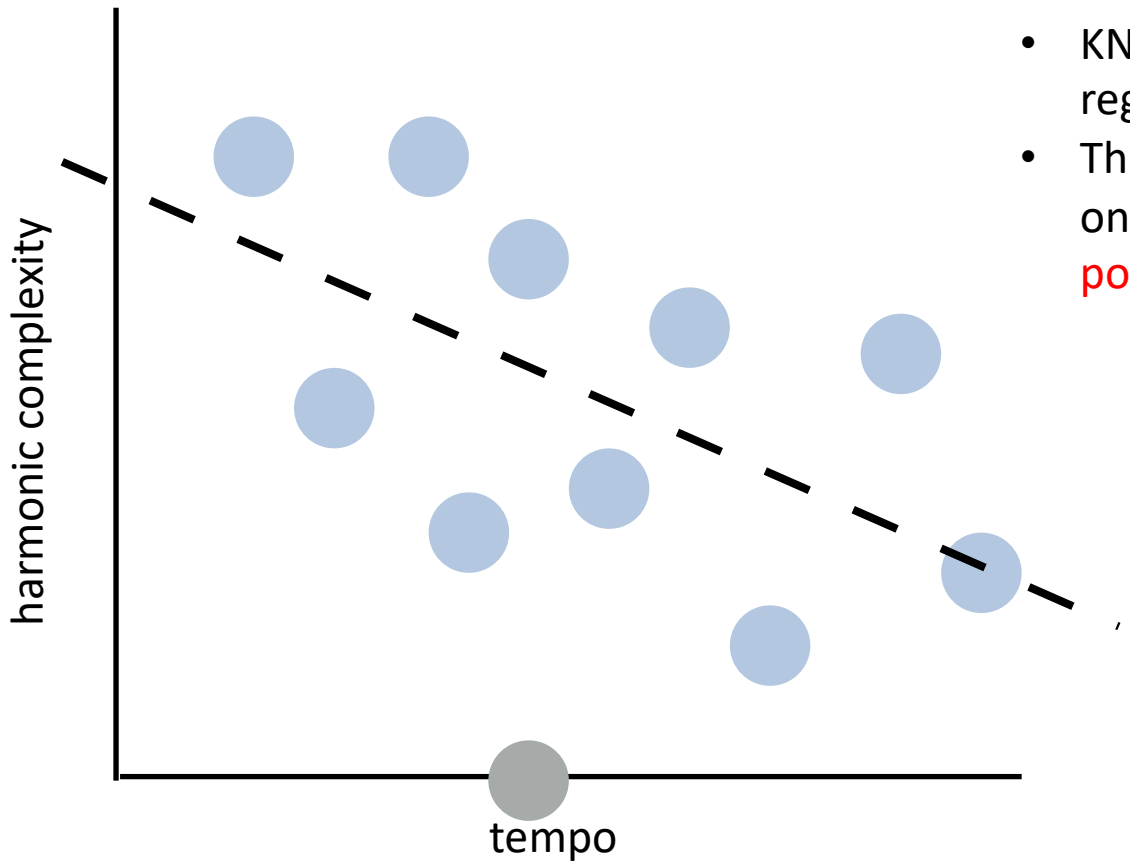
- Look at the k-nearest neighbors and take the majority vote
- Assign to the new point the majority label

K Nearest Neighbors



- K is the parameter that denotes the complexity of the model
- The higher K the more complex assignments we can realize
- How do we decide which are the nearest points/neighbors?
  - Use a measure (e.g., Euclidian Distance)
  - If multiple features, normalize the values and/or use opportune weights

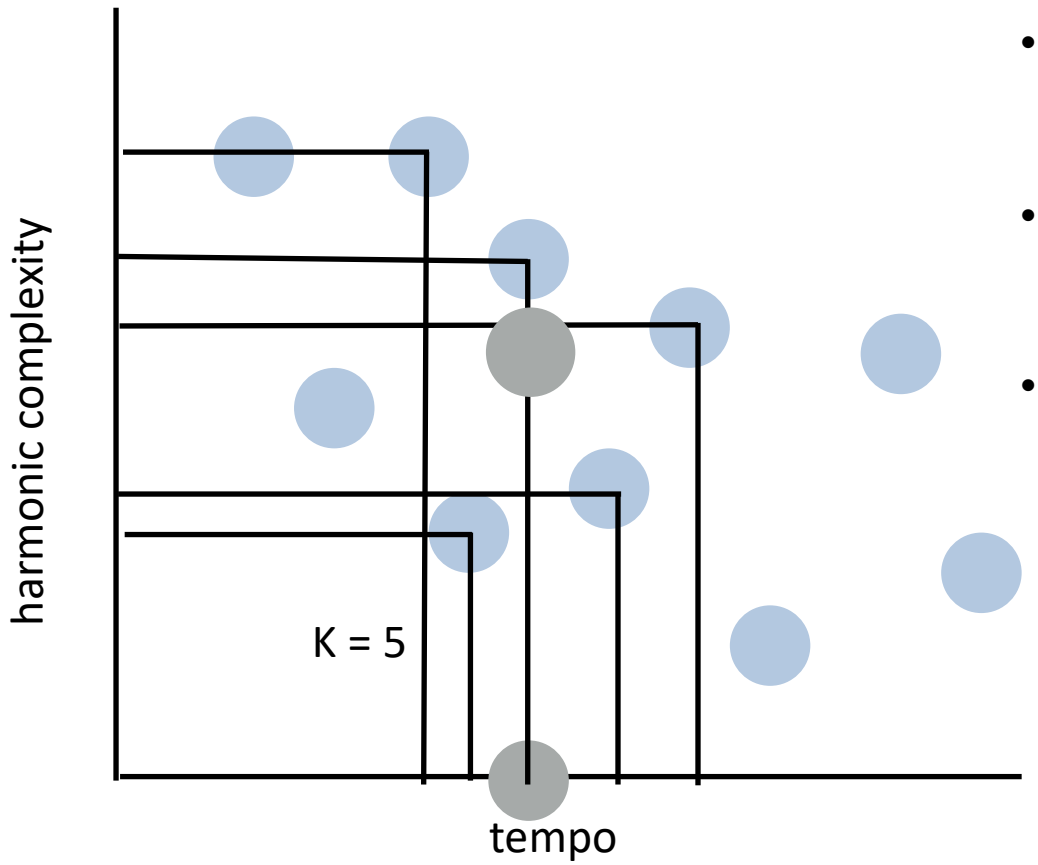
K Nearest Neighbors - Regression



What is its harmonic complexity?

- KNN can also be used for regression
- The prediction of the value is based on the the values of neighboring points

K Nearest Neighbors - Regression



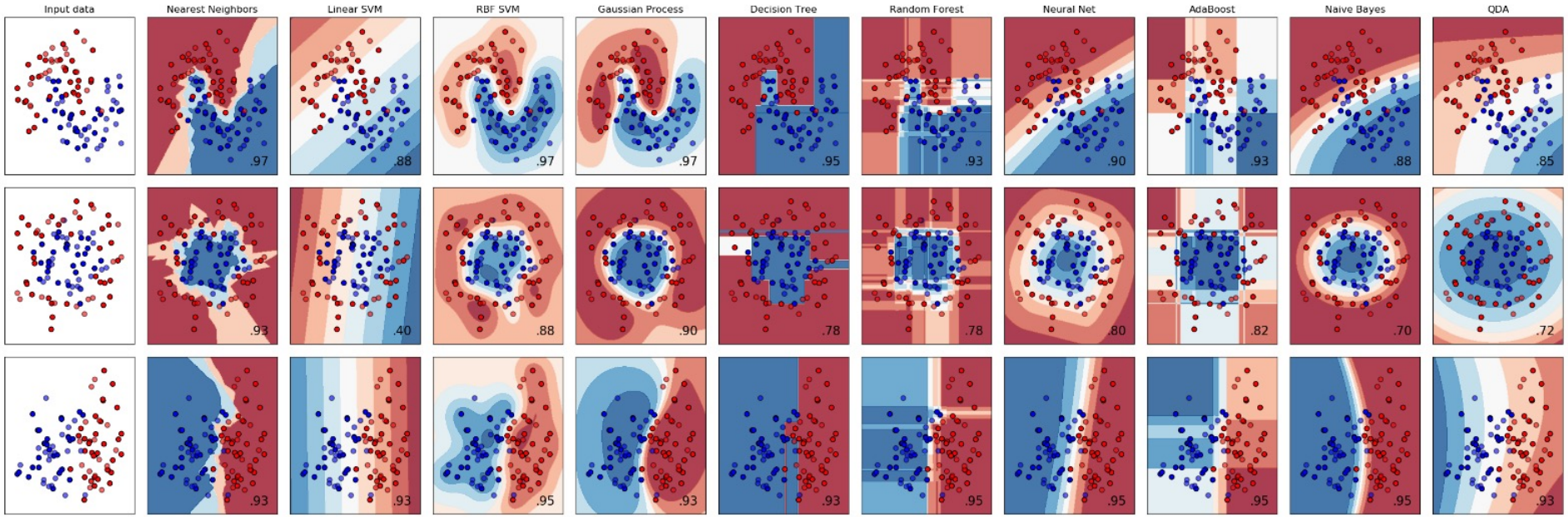
What is its harmonic complexity?

- Example: I want to estimate harmonic complexity (Y) given tempo (X)
- I select the K neighbors whose tempo (X) values are the nearest to that of the point being considered
- The predicted value is is obtained by averaging the values of the Y values of the neighbors

K Nearest Neighbors

- Arguably the simplest ML algorithm
- "Non-Parametric" — no assumptions about the form of the classification model
- No explicit training phase:
  - All the work is done at prediction time
- Works with tiny amounts of training data (single example per class)

Some supervised learning algorithms



Generative Models	Discriminative Models
Estimate P(X, Y) first	Estimate P(Y | X) directly • no explicit probability model
Can assign probability to observations, generate new observations	Only supports classification, less flexible
Often more parameters, but more flexible	Often fewer parameters, better performance on small data
Naive Bayes, Bayes Nets, VAEs, GANs	Logistic Regression, SVMs, Perceptrons, KNN

In the limit, I think these goals *are* the same.
Even if we care about prediction (and we want to do it using as few models as possible), shouldn't we get the best performance by modeling the "true" underlying process?
Isn't it the case that correct explanatory/causal models necessarily make right predictions, but not vice-versa?

Counter argument: You can get perfect* predictive performance with the wrong model. We were extremely good at predicting whether objects would fall or float long before we knew about gravity.

Explanatory/causal models are hard! We might never get there. Maybe empirically accurate predictions should lead, and theory/explanation will follow?

Supervised Classification

Good if not dramatic fizz. ***

Rubbery - rather oxidised. *

Gamy, succulent tannins. Lovely. ****

Provence herbs, creamy, lovely. ****

Lovely mushroomy nose and good length. *****

Quite raw finish. A bit rubbery. **

Supervised Classification

Lovely mushroomy nose and good length. 1

Gamy, succulent tannins. Lovely. 1

Provence herbs, creamy, lovely. 1

Good if not dramatic fizz. 0

Quite raw finish. A bit rubbery. 0

Rubbery - rather oxidised. 0

Supervised Classification

y	X							
Label	lovely	good	raw	rubbery	rather	mushroomy	gamy	…
1	1	1	0	0	0	0	0	…
1	1	0	0	0	0	0	1	…
1	1	0	0	0	0	0	0	…
0	0	0	1	1	0	0	0	…

Supervised Classification

y	X							
Label	lovely	good	raw	rubbery	rather	mushroomy	gamy	…
1	1	1	0	0	0	0	0	…
1	1	0	0	0	0	0	1	…
1	1	0	0	0	0	0	0	…
0	0	0	1	1	0	0	0	…
???	1	0	1	0	1	0	1	…

Bayes Rule

$$P(Y|X) = \frac{P(X|Y)P(Y)}{P(X)}$$

Bayes Rule

$$P(Y|X) = \frac{P(X|Y)P(Y)}{P(X)}$$

Posterior

Bayes Rule

$$P(Y|X) = \frac{P(X|Y)P(Y)}{P(X)}$$

Posterior

Marginal / "Evidence"

- Generally, very hard to estimate!
- Unsupervised learning techniques can be useful
- In Naïve Bayes, we will use some assumption about distribution of the features (e.g. multinomial, Gaussian)
- Since it is the same for all considered labels we can ignore it

Bayes Rule

Likelihood Prior

$$P(Y|X) = \frac{P(X|Y)P(Y)}{P(X)}$$

Posterior

Marginal / "Evidence"

- $P(X|Y)P(Y) = P(X,Y)$
- Equivalent to estimating joint distribution of features and label
- We estimate it form the labeled examples!

Naïve Bayes

Label	lovely	good	raw	rubbery	rather	mushroomy	gamy	…
1	1	1	0	0	0	0	0	…

$$P(Y=1|lovely, good,\ldots)$$
$$=P(lovely, good,\ldots|Y=1)P(Y=1)$$
$$=P(Y=1, lovely, good,\ldots)$$
$$=P(lovely|Y=1, good,\ldots)P(Y=1, good,\ldots)$$

Naïve Bayes

Label	lovely	good	raw	rubbery	rather	mushroomy	gamy	…
1	1	1	0	0	0	0	0	…

$$P(C|x_1, x_2, \ldots, x_k)$$
$$=P(x_1|x_2, \ldots, x_k, C)P(x_2|x_3, \ldots, x_k, C)\ldots P(x_k|C)P(C)$$
$$=P(x_1|C)P(x_2|C)\ldots P(x_k|C)P(C)$$

Assume naively that the features are independent!

Naïve Bayes

Lovely mushroomy nose and good length.	1
Gamy, succulent tannins. Lovely.	1
Provence herbs, creamy, lovely.	1
Quite raw finish. A bit rubbery.	0
Good if not dramatic fizz.	0
Rubbery - rather oxidised.	0

x	$P(x\|Y=1)$	$P(x\|Y=0)$
lovely	??	??
good	??	??
raw	??	??
rubbery	??	??

Question time

Lovely mushroomy nose and good length.	1	Quite raw finish. A bit rubbery.	0
Gamy, succulent tannins. Lovely.	1	Good if not dramatic fizz.	0
Provence herbs, creamy, lovely.	1	Rubbery - rather oxidised.	0

x	$P(x\|Y=1)$	$P(x\|Y=0)$
good	??	??

(a)1.0, 0.0
(b)1/2, 1/2
(c)1/3, 1/3

Question time

Lovely mushroomy nose and good length.	1
Gamy, succulent tannins. Lovely.	1
Provence herbs, creamy, lovely.	1
Quite raw finish. A bit rubbery.	0
Good if not dramatic fizz.	0
Rubbery - rather oxidised.	0

x	$P(x\|Y=1)$	$P(x\|Y=0)$
good	??	??

(a)1.0, 0.0

(b)1/2, 1/2

(c)1/3, 1/3

Naïve Bayes

Given a review "Quite mushroomy, a bit dramatic" what label should we predict?

x	$P(x\|Y=1)$	$P(x\|Y=0)$
a	0.9	0.9
bit	0.2	0.4
dramatic	0.6	0.4
gamy	0.1	0.0
good	0.2	0.2
lovely	0.5	0.1
mushroomy	0.2	0.2
quite	0.7	0.8

$$P(Y|X) = P(X|Y)P(Y)/P(X)$$

Naïve Bayes

Given a review "Quite mushroomy, a bit dramatic" what label should we predict?

x	$P(x\|Y=1)$	$P(x\|Y=0)$
a	0.9	0.9
bit	0.2	0.4
dramatic	0.6	0.4
gamy	0.1	0.0
good	0.2	0.2
lovely	0.5	0.1
mushroomy	0.2	0.2
quite	0.7	0.8

$$P(Y|X) = \frac{P(X|Y)P(Y)}{P(X)}$$

Generally, the distribution over the labels is unknown!

- Domain knowledge
- Estimate from data
- Naïve uniform assumption

Naïve Bayes

Given a review "Quite mushroomy, a bit dramatic" what label should we predict?

x	$P(x\|Y=1)$	$P(x\|Y=0)$
a	0.9	0.9
bit	0.2	0.4
dramatic	0.6	0.4
gamy	0.1	0.0
good	0.2	0.2
lovely	0.5	0.1
mushroomy	0.2	0.2
quite	0.7	0.8

$$P(Y|X) = \frac{P(X|Y)P(Y)}{P(X)}$$

Prior

$P(Y=1)$	$P(Y=0)$
0.3	0.7

We pick the label y

$$argmax_y\, P(Y=y|X)$$

For these values

a) Y=1

b) Y=0

x	$P(x\|Y=0)$	$P(x\|Y=1)$
a	0.9	0.9
bit	0.2	0.4
dramatic	0.6	0.4
gamy	0.1	0.0
good	0.2	0.2
lovely	0.5	0.1
mushroomy	0.2	0.2
quite	0.7	0.8

Prior

P(Y=0)	P(Y=1)
0.3	0.7

P(Y=0 , X=x) = 0.9 x 0.2 x 0.6 x 0.2 x 0.7 x <u>0.3</u> = 0.005

P(Y=1 , X=x) 0.9 x 0.4 x 0.4 x 0.2 x 0.8 x <u>0.7</u> = 0.016

Naïve Bayes– Generative model

x	P(x|Y=1)	P(x|Y=0)
a	0.9	0.9
bit	0.2	0.4
dramatic	0.6	0.4
gamy	0.1	0.0
good	0.2	0.2
lovely	0.5	0.1
mushroomy	0.2	0.2
quite	0.7	0.8

Naïve Bayes– Generative model

x	P(x|Y=1)	P(x|Y=0)
a	**0.9**	0.9
bit	0.2	0.4
dramatic	0.6	0.4
gamy	0.1	0.0
good	0.2	0.2
lovely	0.5	0.1
mushroomy	0.2	0.2
quite	0.7	0.8

A … 0.9

Naïve Bayes– Generative model

x	P(x|Y=1)	P(x|Y=0)
a	0.9	0.9
bit	0.2	0.4
dramatic	0.6	0.4
gamy	0.1	0.0
good	0.2	0.2
lovely	0.5	0.1
mushroomy	0.2	0.2
quite	**0.7**	0.8

A quite … 0.63

Naïve Bayes – Generative model

x	P(x|Y=1)	P(x|Y=0)
a	0.9	0.9
bit	0.2	0.4
dramatic	**0.6**	0.4
gamy	0.1	0.0
good	0.2	0.2
lovely	0.5	0.1
mushroomy	0.2	0.2
quite	0.7	0.8

A quite dramatic … 0.38

Naïve Bayes – Generative model

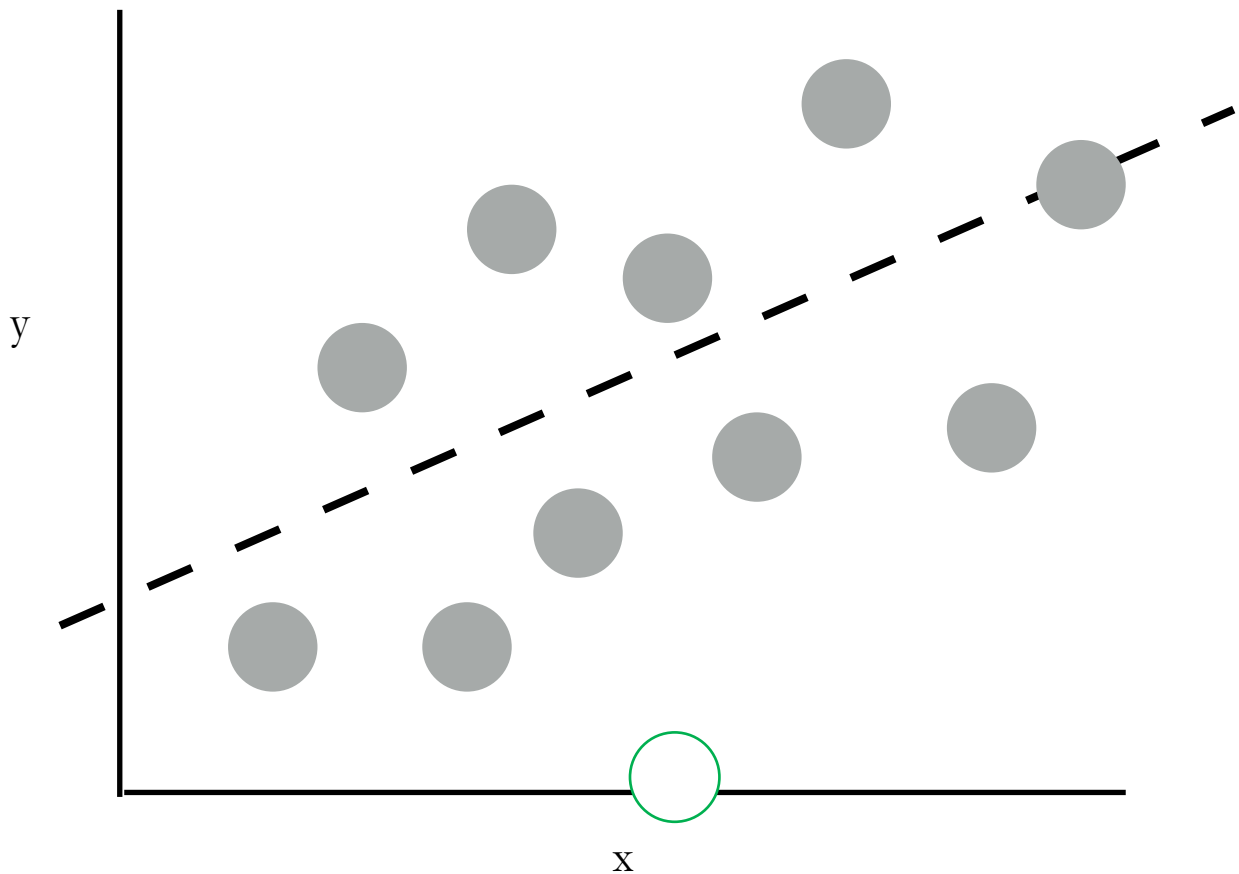
x	P(x|Y=1)	P(x|Y=0)
a	0.9	0.9
bit	0.2	0.4
dramatic	0.6	0.4
gamy	**0.1**	0.0
good	0.2	0.2
lovely	0.5	0.1
mushroomy	0.2	0.2
quite	0.7	0.8

A quite dramatic gamy … 0.04

Predictions with Linear Regression

$$y = w_1 x_1 + w_2 x_2 + \cdots + w_k x_k$$

$$y = \vec{w} \cdot \vec{x}$$



Regression Analysis in Stats	Regression in ML
Make claims about whether there is a statistically significant relationship between X and Y	Given X, predict Y Obtain a model to make predictions for new inputs
(Often) interested in correlation; focus on controlling false positives and removing colinearity	Focused on prediction accuracy; exploiting correlation is totally fine
A "result" is typically in the form of a decision on significant relationship and/or p-value	A "result" is typically in the form of an improvement in prediction performance on a (held out) test set
Avoid overfitting by preferring simple models; avoid overclaiming by accounting for "degrees of freedom" when computing p values	Avoid overfitting through regularization; avoid overclaiming by maintaining train/test splits and reporting test performance

Regression in Stats/ML

- Still, these are, fundamentally, the same model
- These differences are “by convention”
- Different scientific communities with different goals
- The two methods yield different guarantees

Training with Gradient Descent

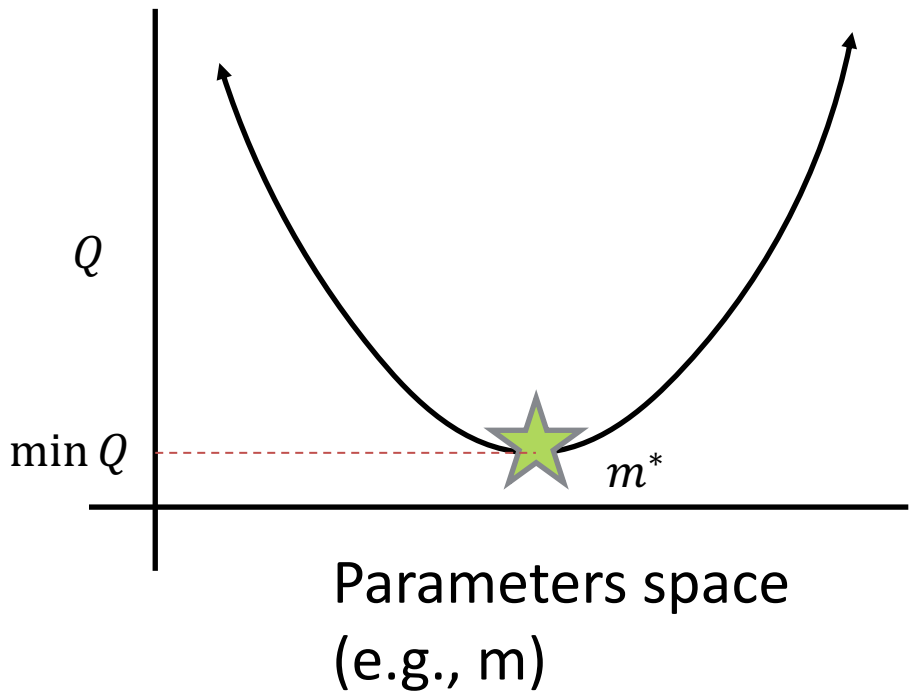
We want to find the model $f(x) = mx + b$ which minimizes the sum of squared predictions error (least squares-SLQ)

$$Q = \sum_{i=1}^{n} \left(\bar{Y} - f(x)\right)^2 = \sum_{i=1}^{n} (\bar{Y} - (mx + b))^2$$

Q minimized by

$$m^* = \frac{Cov(X,Y)}{Var(X)}$$

$$b^* = \bar{Y} - m\bar{X}$$



Training with Gradient Descent

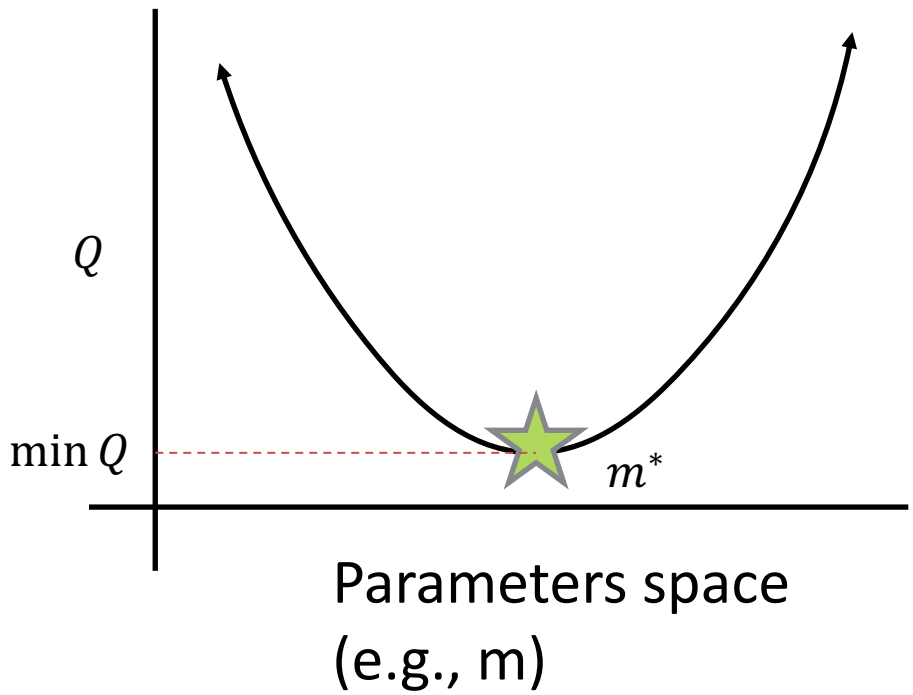
We want to find the model $f(x) = mx + b$ which minimizes the sum of squared predictions error (least squares-SLQ)

$$Q = \sum_{i=1}^{n} \left(\bar{Y} - f(x)\right)^2 = \sum_{i=1}^{n} (\bar{Y} - (mx + b))^2$$

Q minimized by

$$m^* = \frac{Cov(X, Y)}{Var(X)}$$

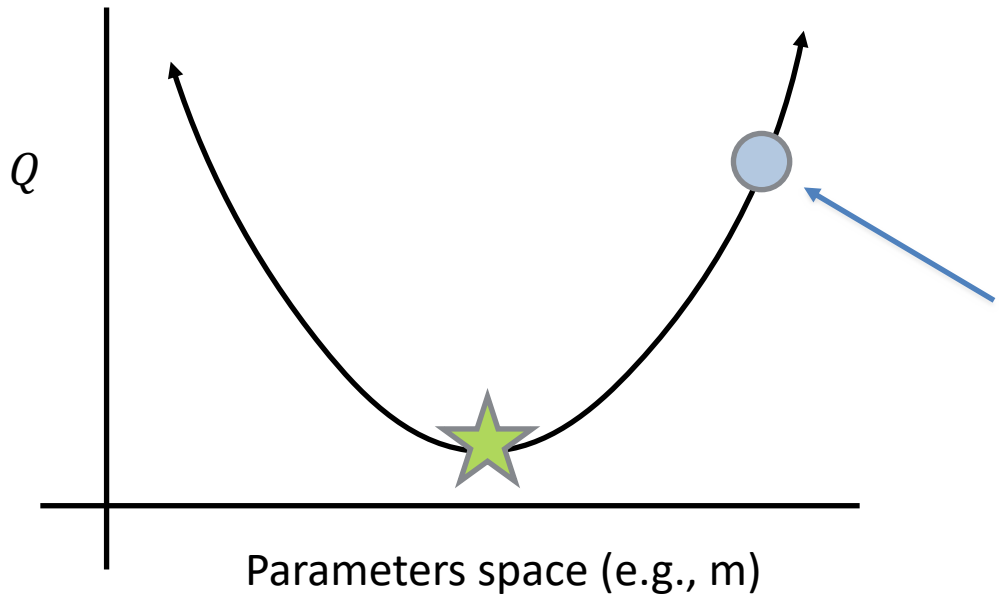
$$b^* = \bar{Y} - m\bar{X}$$



Training with Gradient Descent

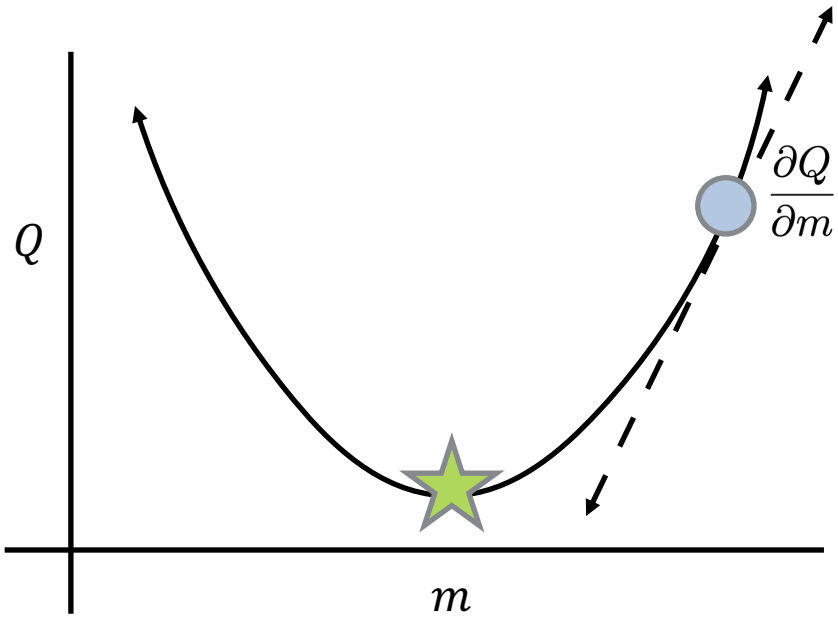
- We cannot/don't want to rely on asymptotic assumptions or convergence
- We don't want to explore the entire space of possible parameters
  - Very computationally inefficient
- Can we explore the space in a smart way?
- Goal is unchanged: find the model that minimizes

$$Q = \sum_{i=1}^{n} \left(\bar{Y} - f(x)\right)^2 = \sum_{i=1}^{n} (\bar{Y} - (mx + b))^2$$



Start with a random guess of the parameters (m,b) and compute Q

Training with Gradient Descent

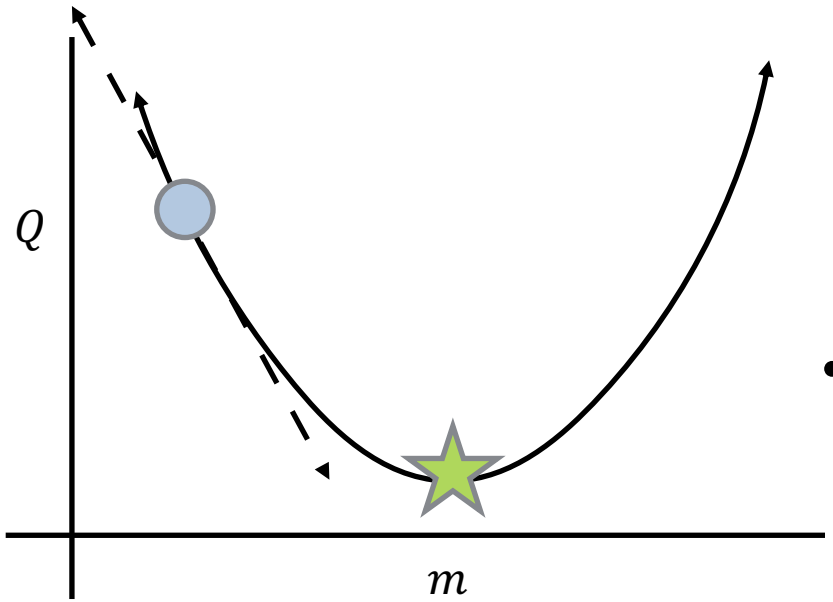


$$\frac{\partial Q}{\partial m} = \sum_{i=1}^{n} -2X_i(Y_i - b - mX_i)$$

- Positive derivative: re-evalua
  for smaller value m

- We compute the partial derivative $\frac{\partial Q}{\partial m}$ (the gradient) Q with respect to the parameter m in correspondence of the selected point
- We adjust the current selection of the parameter value based on the sign of the derivative
- The new m will be $m - \alpha \frac{\partial Q}{\partial m}$
- α is a Hyperparameter called "learning rate"

Training with Gradient Descent



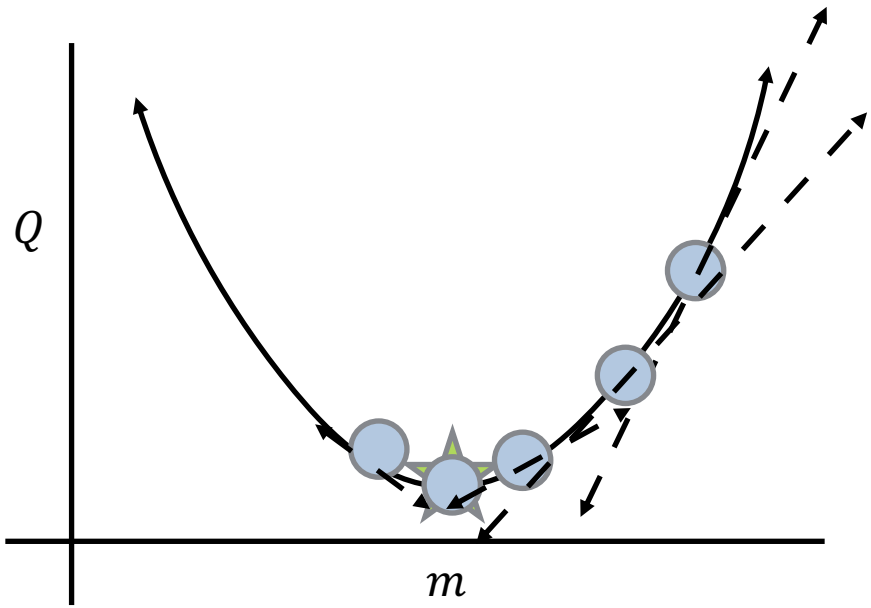
$$\frac{\partial Q}{\partial m} = \sum_{i=1}^{n} -2X_i(Y_i - b - mX_i)$$

- Negative derivative: re-evaluate for higher value m

- We compute the partial derivative $\frac{\partial Q}{\partial m}$ (the gradient) Q with respect to the parameter m in correspondence of the selected point
- We adjust the current selection of the parameter value based on the sign of the derivative
- The new m will be $m - \alpha \frac{\partial Q}{\partial m}$
- α is a Hyperparameter called "learning rate"

Training with Gradient Descent

- The process is repeated iteratively until no further reduction of Q is possible
- We can have stopping conditions (e.g., minimum improvement, maximum #iterations)
- #max iterations, learning rate, minimum improvement are the hyperparameters of the algorithm

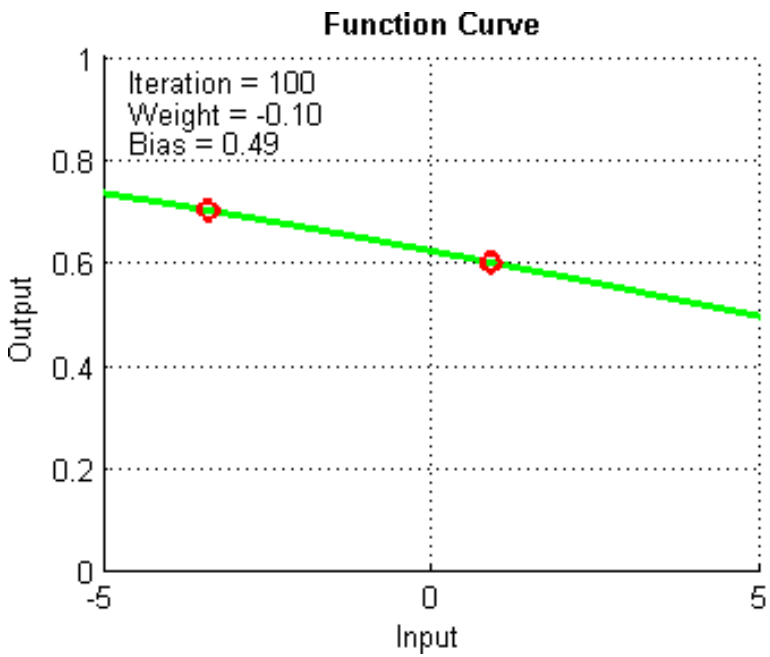


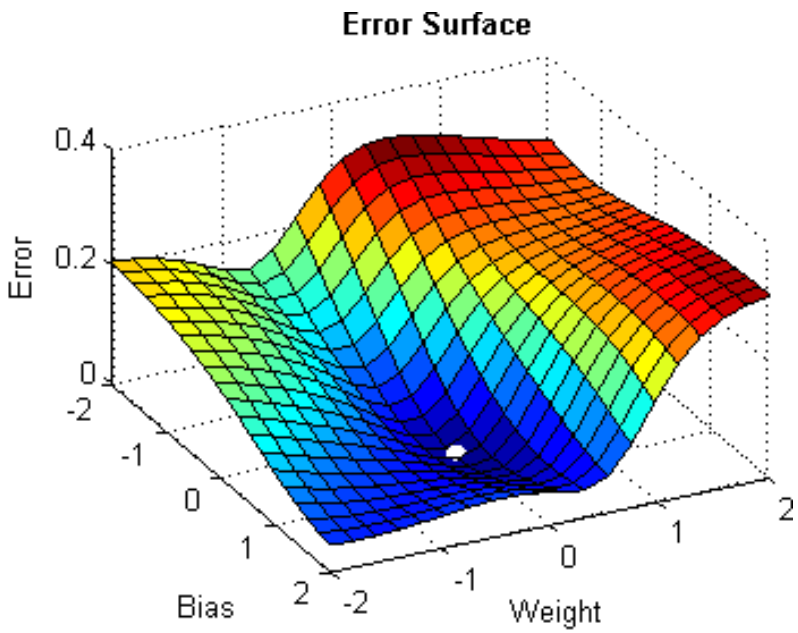
Training with Gradient Descent

- The process is repeated iteratively until no further reduction of Q is possible
- If there are multiple parameters, do the same operations for each feature individually in each iteration

```
def update_weights(m, b, X, Y, learning_rate):
    m_deriv = 0
    b_deriv = 0
    N = len(X)
    for i in range(N):
        # Calculate partial derivatives
        # -2x(y - (mx + b))
        m_deriv += -2*X[i] * (Y[i] - (m*X[i] + b))

        # -2(y - (mx + b))
        b_deriv += -2*(Y[i] - (m*X[i] + b))

    # We subtract because the derivatives point in direction of steepest ascent
    m -= (m_deriv / float(N)) * learning_rate
    b -= (b_deriv / float(N)) * learning_rate

    return m, b
```

Training with Gradient Descent





Stochastic Gradient Descent (SGD)

- It is an iterative method for optimizing an objective function with suitable smoothness properties (e.g. differentiable or subdifferentiable).
- It can be regarded as a stochastic approximation of gradient descent optimization:
  - It replaces the actual gradient calculated from the entire data set
  - Uses an estimate calculated from a randomly selected subset of the data (generally a single point).
- Especially in high-dimensional optimization problems this reduces the very high computational burden
  - Achieves faster iterations in trade for a lower convergence rate

Stochastic Gradient Descent (SGD)

- How would you modify this?

```
def update_weights(m, b, X, Y, learning_rate):
    m_deriv = 0
    b_deriv = 0
    N = len(X)
    for i in range(N):
        # Calculate partial derivatives
        # -2x(y - (mx + b))
        m_deriv += -2*X[i] * (Y[i] - (m*X[i] + b))

        # -2(y - (mx + b))
        b_deriv += -2*(Y[i] - (m*X[i] + b))

    # We subtract because the derivatives point in direction of steepest ascent
    m -= (m_deriv / float(N)) * learning_rate
    b -= (b_deriv / float(N)) * learning_rate

    return m, b
```

Training with Gradient Descent

Helpful equations for following along in the jupyter notebook

$$Q = \sum_{i=1}^{n} (Y_i - (mX_i + b))^2$$

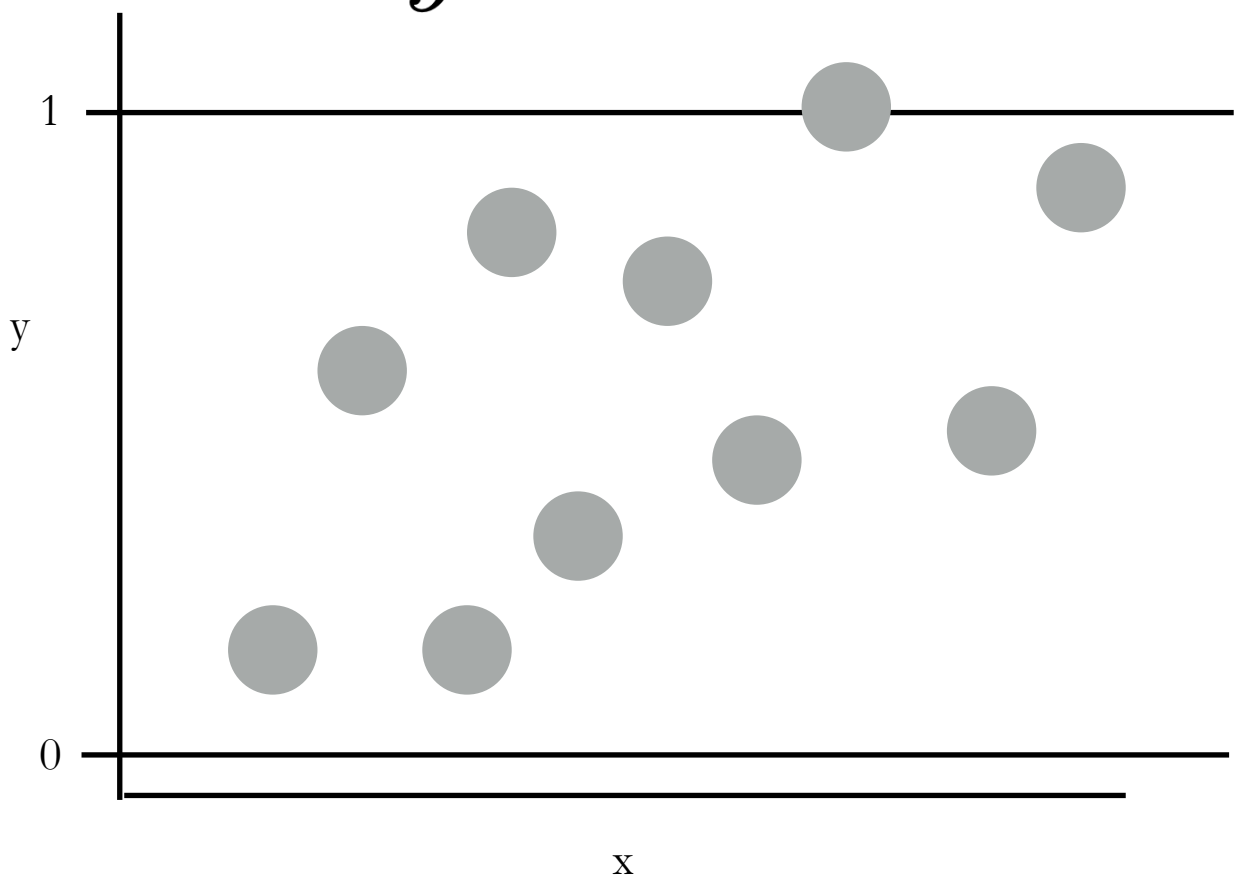
$$\frac{\partial Q}{\partial b} = \sum_{i=1}^{n} -2(Y_i - mX_i - b) = 0$$

$$\frac{\partial Q}{\partial m} = \sum_{i=1}^{n} -2X_i(Y_i - b - mX_i) = 0$$

$$m = \frac{Cov(X,Y)}{Var(X)} \qquad b = \bar{Y} - m\bar{X}$$

Linear Regression for Classification?

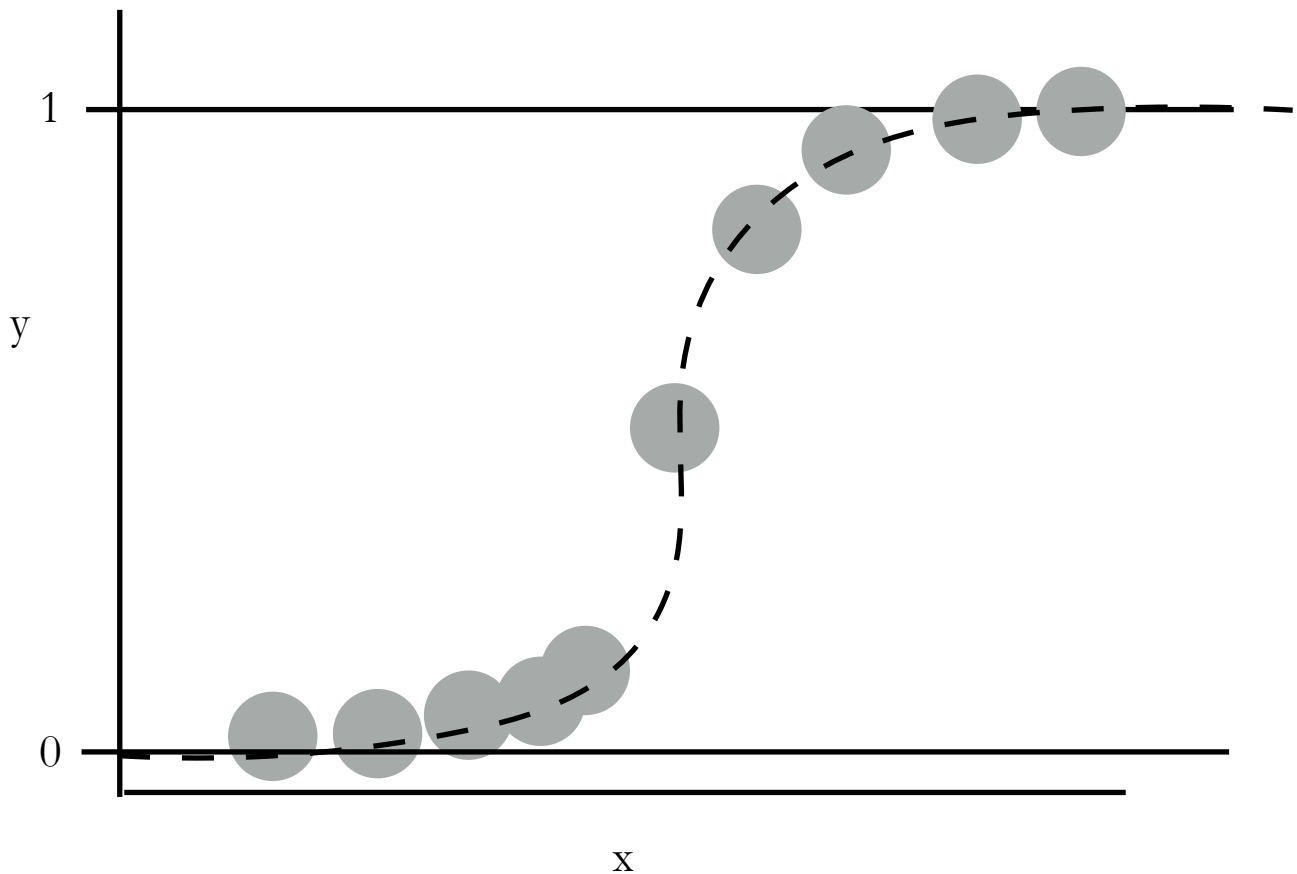
$$y = \vec{w} \cdot \vec{x}$$



Logistic Regression

$$y = \frac{1}{1 + e^{-(\vec{w} \cdot \vec{x})}}$$

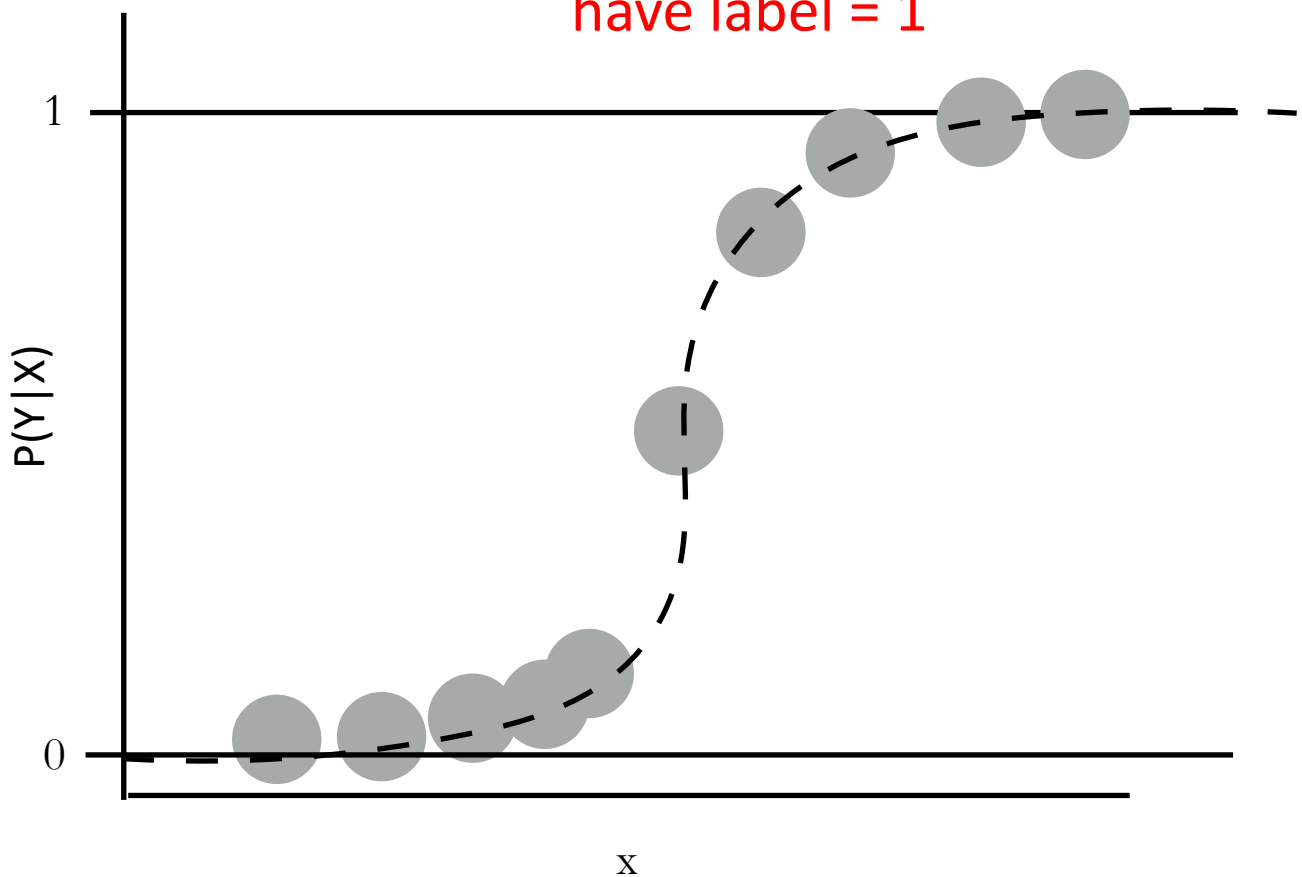
Sigmoid function



Logistic Regression

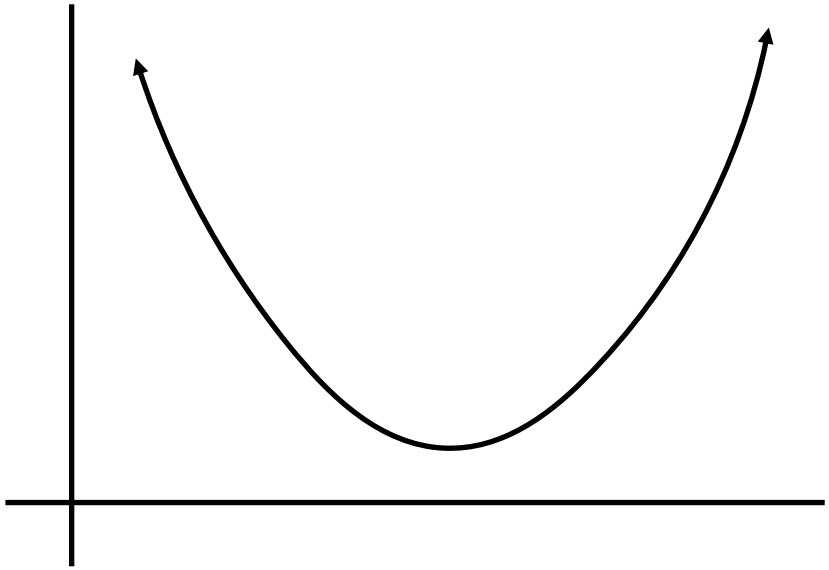
$$y = \frac{1}{1 + e^{-(\vec{w} \cdot \vec{x})}}$$

We can interpret the computed y as the probability of a point characterized by features $\vec{x}$ to have label = 1



Linear Regression

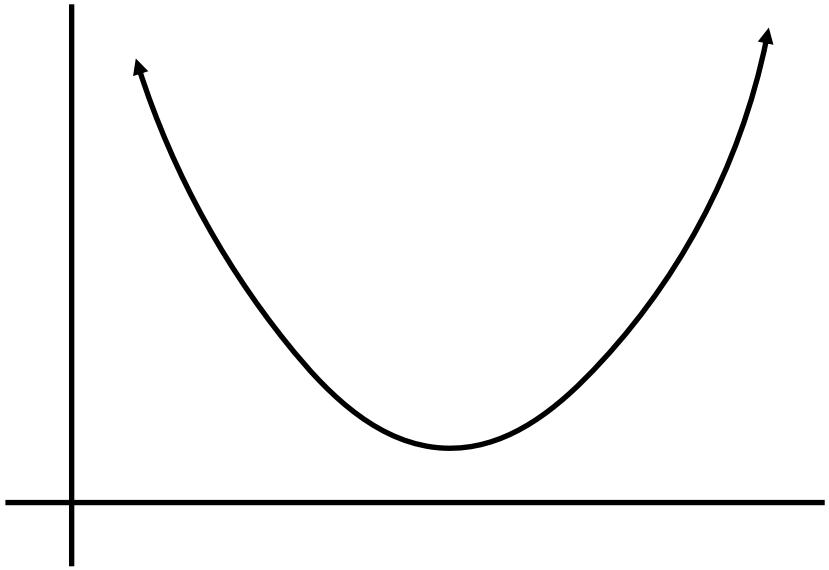
$$\text{minimize} \quad \sum_{i=1}^{n} (Y_i - \hat{Y})^2$$



Logistic Regression

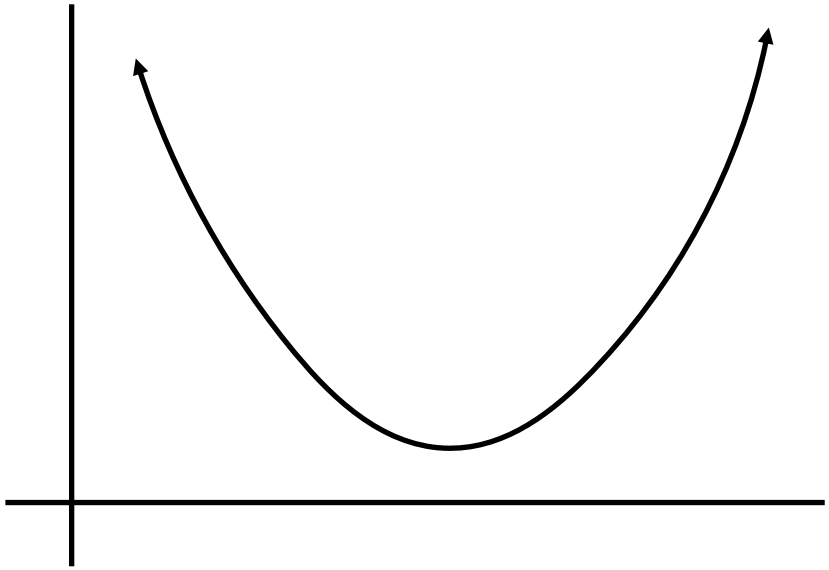
Find the parameters w_i which minimize the logistic loss

$$\text{minimize} \qquad -logP(Y|\hat{Y})$$



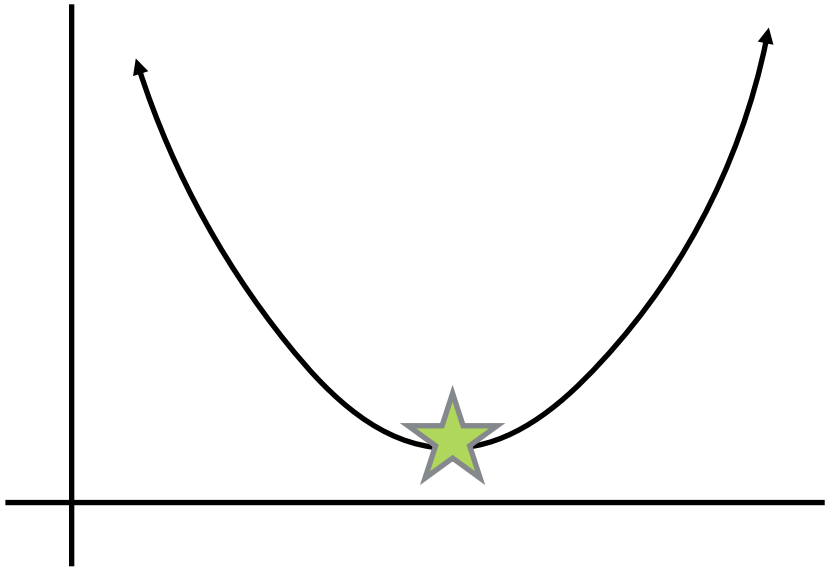
Logistic Regression

$$\text{minimize} \qquad -Y log\hat{Y} + (1-Y)log(1-\hat{Y})$$



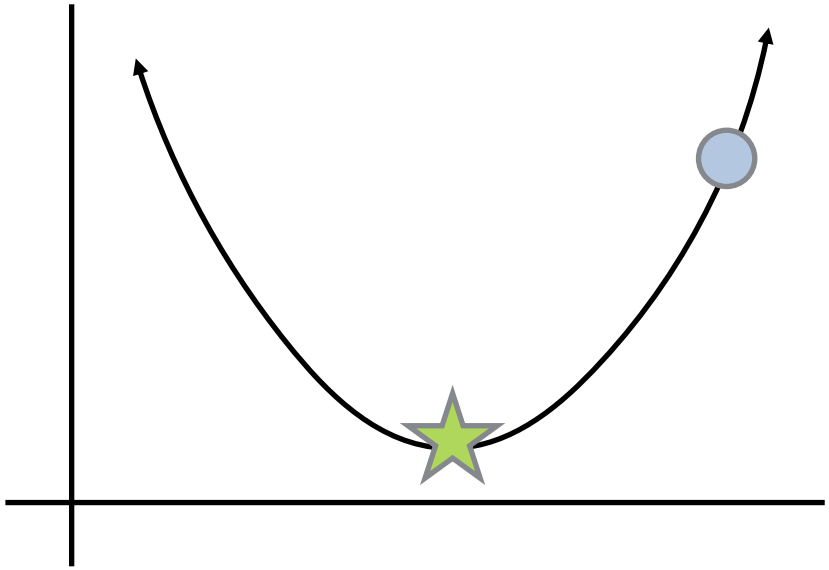
Logistic Regression

$$\text{minimize} \qquad -Y log\hat{Y} + (1-Y)log(1-\hat{Y})$$



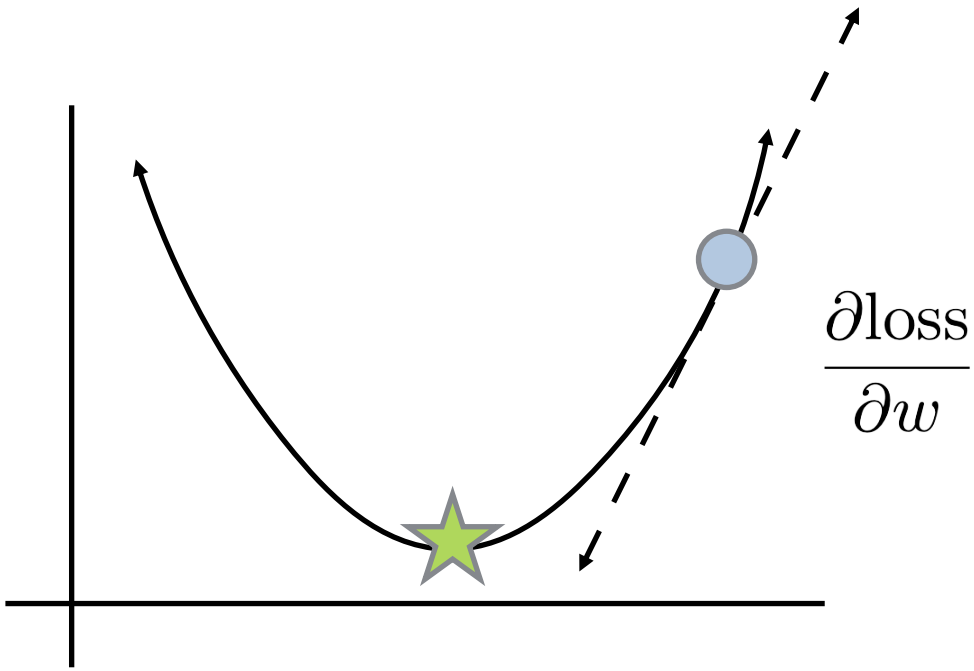
Logistic Regression

$$\text{minimize} \qquad -Y log\hat{Y} + (1-Y)log(1-\hat{Y})$$



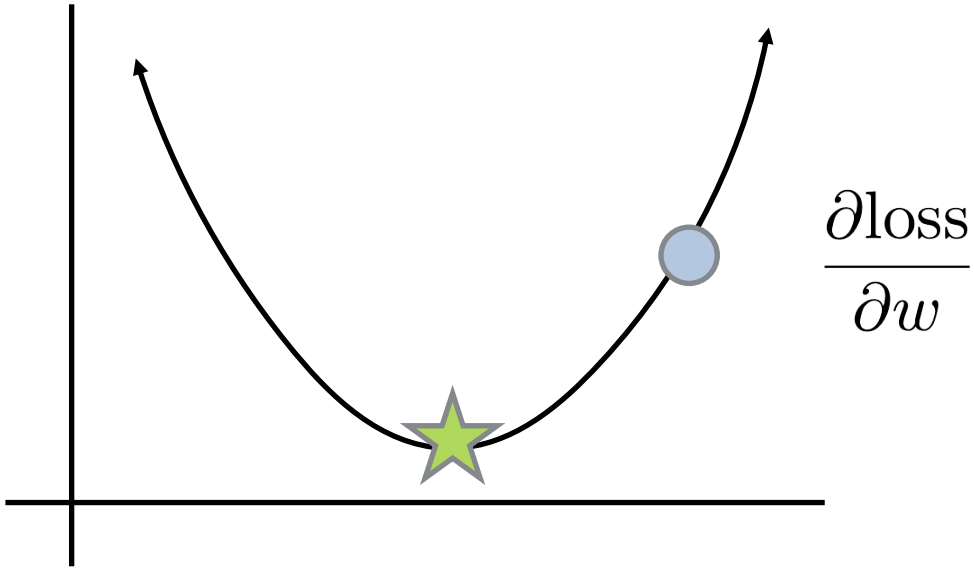
Logistic Regression

minimize $$-Y log\hat{Y} + (1-Y)log(1-\hat{Y})$$



Logistic Regression

$$\text{minimize} \qquad -Y log\hat{Y} + (1-Y)log(1-\hat{Y})$$



Logistic Regression

Naive Bayes

x	P(x|Y=1)
a	0.9
bit	0.2
dramatic	0.6
gamy	0.1
good	0.2
lovely	0.5
mushroomy	0.2
quite	0.7

Logistic Regression

Logistic Regression

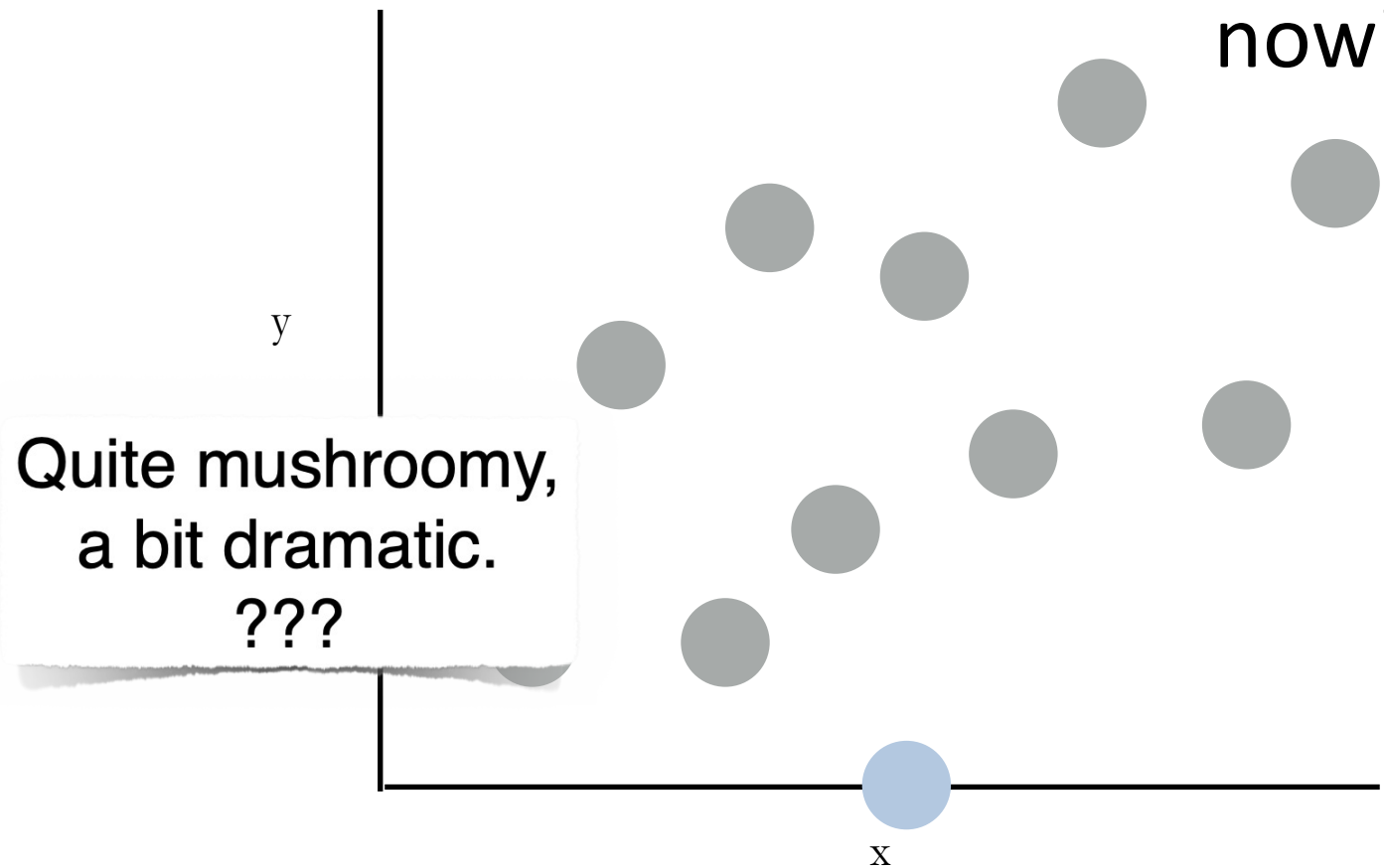
x	???
a	0.9
bit	0.4
dramatic	1.0
gamy	0.7
good	0.2
lovely	0.4
mushroomy	0.8
quite	0.7

What do these coefficient mean?

a) There is a 1.0 probability of observing "dramatic" given Y = 1
b) There is a 1.0 probability that Y = 1 given we observe "dramatic"
c) 1 is the co-efficient on the "dramatic" variable in the best fit linear regression.
d) 1 is the co-efficient on the "dramatic" variable in linear regression that minimizes the log loss.

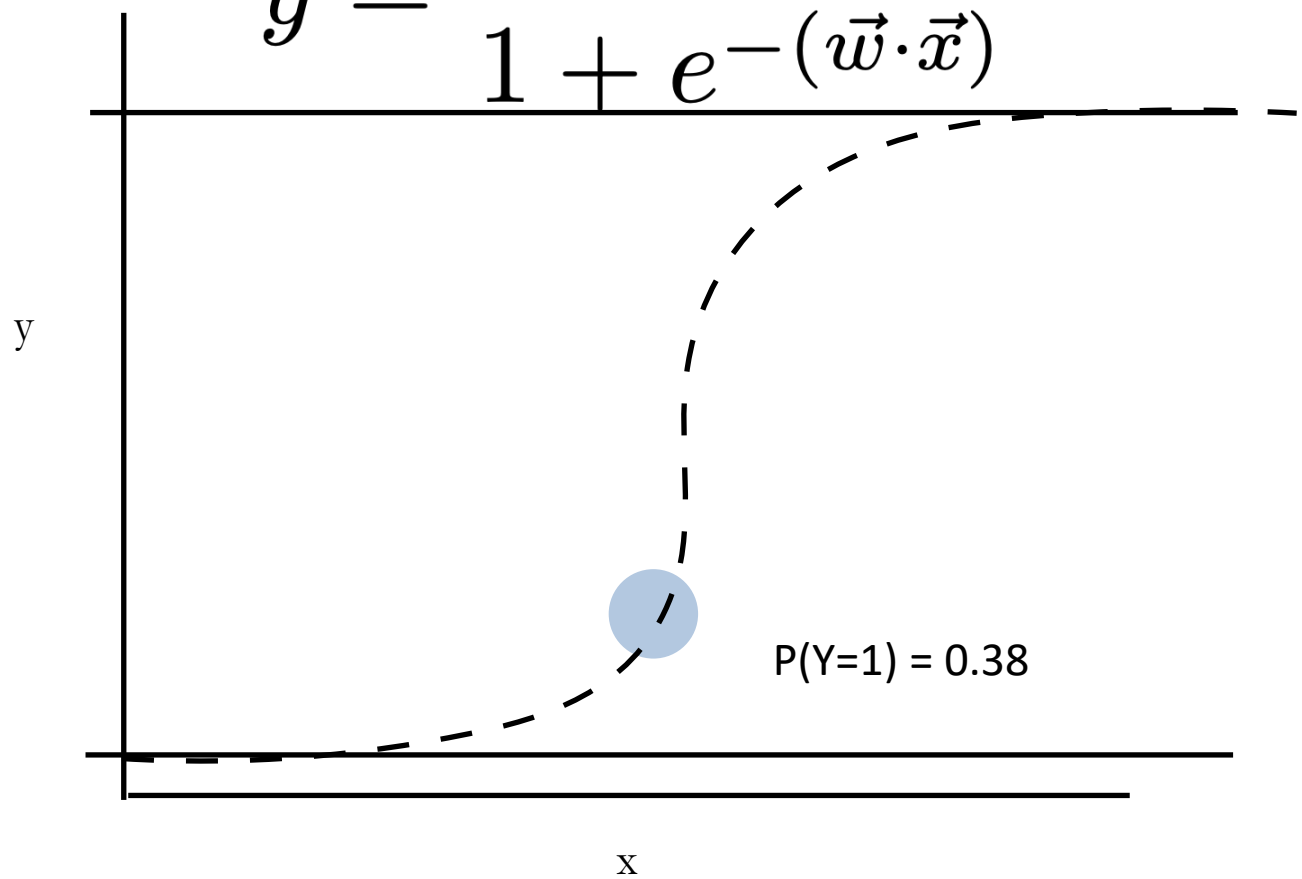
Logistic Regression - Prediction

What do we do now?



Logistic Regression - Prediction

$$y = \frac{1}{1 + e^{-(\vec{w} \cdot \vec{x})}}$$



Logistic Regression - Prediction

$$y = \frac{1}{1 + e^{-(\vec{w} \cdot \vec{x})}}$$