

APS1070

Foundations of Data Analytics and
Machine Learning

Summer 2020

Wed July 22 / Week 11:

- *Binary Classification*
- *Multiclass Classification*
- *Convexity*
- *Big-O Notation*

Jason Riordon, PhD



Question 6

`precision_score(y_true, y_pred)`



Known Labels



Model Predictions for a particular threshold

For Project 2, you used something like...

```
precision = precision_score(label, P1 < sortedP1)
```

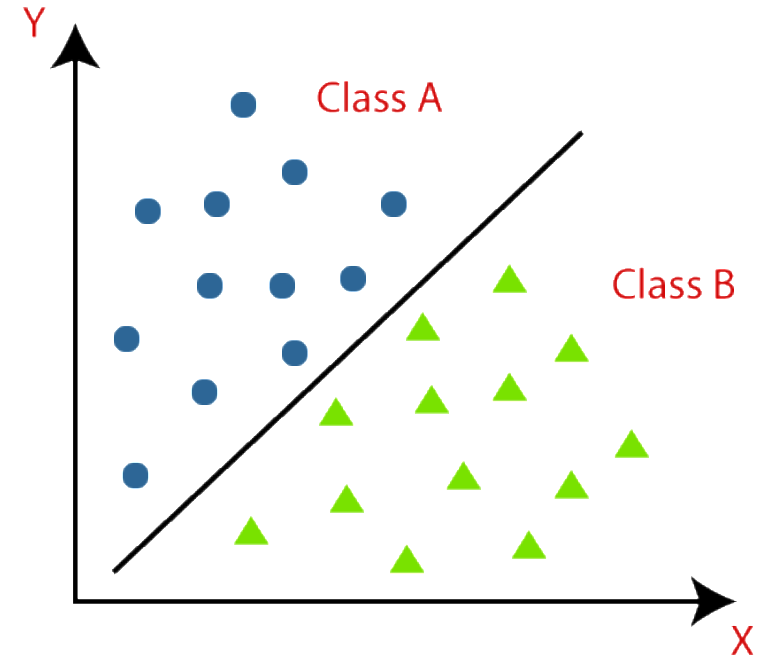
For Project 2, P1 was the probability that a point belongs to the gaussian model... Here, for Q6, the given probabilities are that a point is spam, an outlier.

Note on slides

Certain slides presented herein are reproduced or modified from external sources -> special thanks to **Roger Grosse** and **Scott Sanner** this week!

Binary Classification

Overview



- **Classification**: predicting a discrete-valued target
 - **Binary classification**: predicting a binary-valued target
- **Examples**
 - predict whether a patient has a disease, given the presence or absence of various symptoms
 - classify e-mails as spam or non-spam
 - predict whether a financial transaction is fraudulent

Binary Classification

Binary linear classification

- **classification:** predict a discrete-valued target
- **binary:** predict a binary target $t \in \{0, 1\}$
 - Training examples with $t = 1$ are called **positive examples**, and training examples with $t = 0$ are called **negative examples**.
- **linear:** model is a linear function of \mathbf{x} , followed by a threshold:

$$z = \mathbf{w}^T \mathbf{x} + b$$

$$y = \begin{cases} 1 & \text{if } z \geq r \\ 0 & \text{if } z < r \end{cases}$$

Binary Classification

Eliminating the threshold

- We can assume WLOG that the threshold $r = 0$:

$$\mathbf{w}^T \mathbf{x} + b \geq r \iff \mathbf{w}^T \mathbf{x} + \underbrace{b - r}_{\triangleq b'} \geq 0.$$

Eliminating the bias

- Add a dummy feature x_0 which always takes the value 1. The weight w_0 is equivalent to a bias.

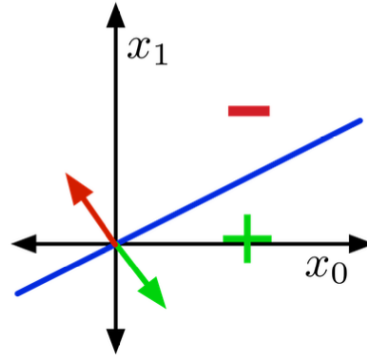
Simplified model

$$z = \mathbf{w}^T \mathbf{x}$$
$$y = \begin{cases} 1 & \text{if } z \geq 0 \\ 0 & \text{if } z < 0 \end{cases}$$



The Geometric Picture

Input Space, or Data Space



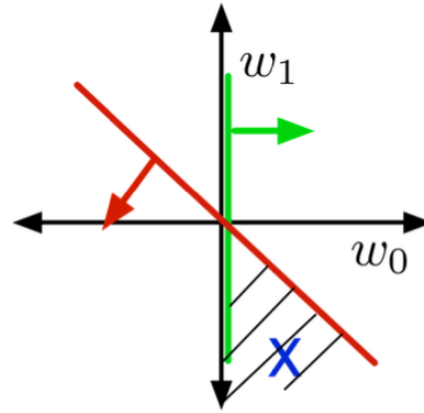
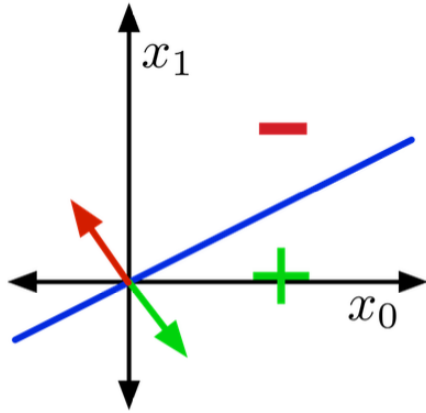
NOT

x_0	x_1	t
1	0	1
1	1	0

- Here we're visualizing the **NOT** example
- Training examples are points
- Hypotheses are **half-spaces** whose boundaries pass through the origin
- The boundary is the **decision boundary**
 - In 2-D, it's a line, but think of it as a hyperplane
- If the training examples can be separated by a linear decision rule, they are **linearly separable**.

The Geometric Picture

Weight Space



$$w_0 > 0$$
$$w_0 + w_1 < 0$$

- Hypotheses are points
- Training examples are half-spaces whose boundaries pass through the origin
- The region satisfying all the constraints is the **feasible region**; if this region is nonempty, the problem is **feasible**

What is the loss function?

- **Recall: binary linear classifiers.** Targets $t \in \{0, 1\}$

$$z = \mathbf{w}^T \mathbf{x} + b$$

$$y = \begin{cases} 1 & \text{if } z \geq 0 \\ 0 & \text{if } z < 0 \end{cases}$$

- What if we can't classify all the training examples correctly?
- Seemingly obvious loss function: **0-1 loss**

$$\mathcal{L}_{0-1}(y, t) = \begin{cases} 0 & \text{if } y = t \\ 1 & \text{if } y \neq t \end{cases}$$

Attempt 1: 0-1 loss

$$\mathcal{L}_{0-1}(y, t) = \begin{cases} 0 & \text{if } y = t \\ 1 & \text{if } y \neq t \end{cases}$$

- Problem: how to optimize?
- Chain rule:

$$\frac{\partial \mathcal{L}_{0-1}}{\partial w_j} = \frac{\partial \mathcal{L}_{0-1}}{\partial z} \frac{\partial z}{\partial w_j}$$

- But $\partial \mathcal{L}_{0-1} / \partial z$ is zero everywhere it's defined!
 - $\partial \mathcal{L}_{0-1} / \partial w_j = 0$ means that changing the weights by a very small amount probably has no effect on the loss.
 - The gradient descent update is a no-op.

Attempt 2: Linear Regression

- Sometimes we can replace the loss function we care about with one which is easier to optimize. This is known as a **surrogate loss function**.
- We already know how to fit a linear regression model. Can we use this instead?

$$y = \mathbf{w}^\top \mathbf{x} + b$$

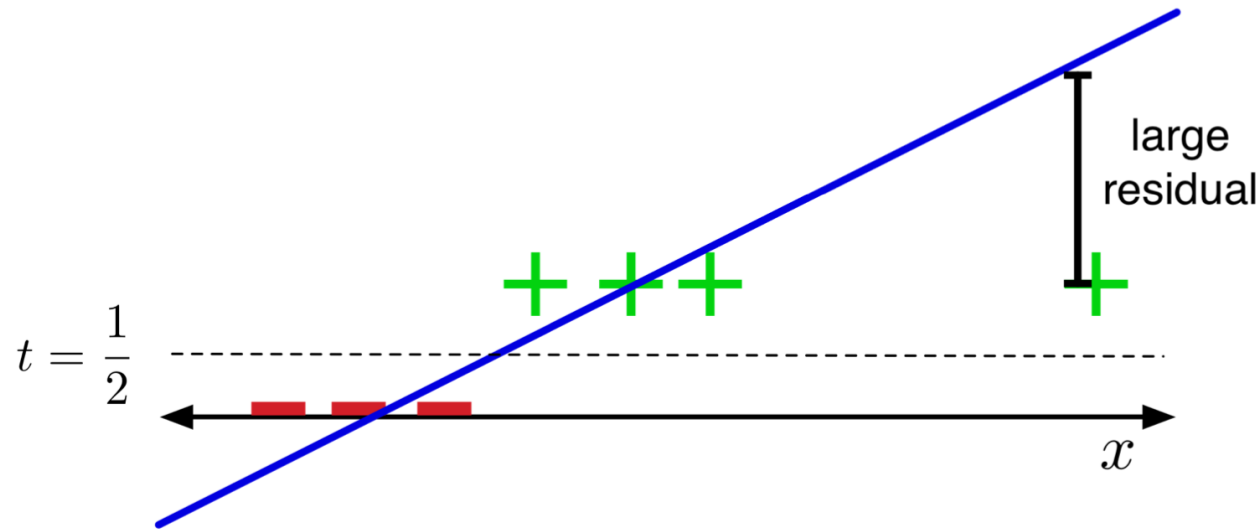
$$\mathcal{L}_{\text{SE}}(y, t) = \frac{1}{2}(y - t)^2$$

- Doesn't matter that the targets are actually binary.
- Threshold predictions at $y = 1/2$.

Attempt 2: Linear Regression

$$\mathcal{L}_{\text{SE}}(y, t) = \frac{1}{2}(y - t)^2$$

The problem:

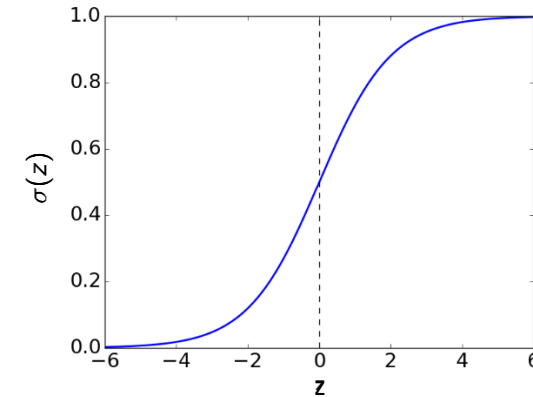


- The loss function hates when you make correct predictions with high confidence!
- If $t = 1$, it's more unhappy about $y = 10$ than $y = 0$.

Attempt 3: Logistic Activation Function

- There's obviously no reason to predict values outside $[0, 1]$. Let's squash y into this interval.
- The **logistic function** is a kind of **sigmoidal**, or S-shaped, function:

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



- A linear model with a logistic nonlinearity is known as **log-linear**:

$$z = \mathbf{w}^\top \mathbf{x} + b$$

$$y = \sigma(z)$$

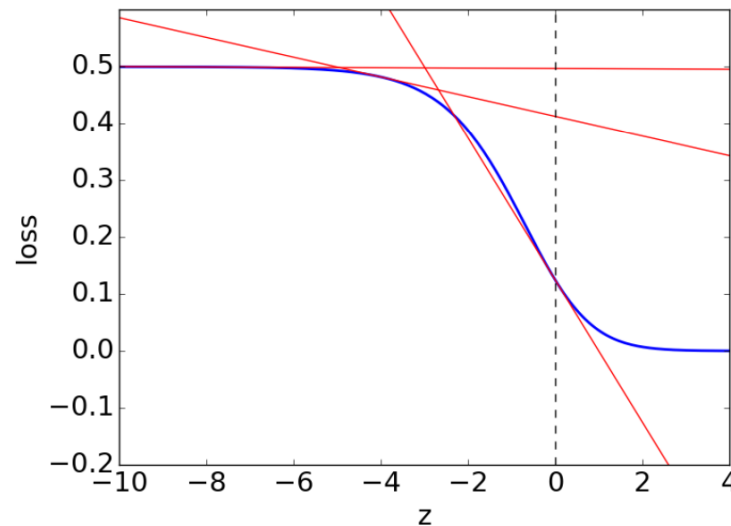
$$\mathcal{L}_{\text{SE}}(y, t) = \frac{1}{2}(y - t)^2.$$

- Used in this way, σ is called an **activation function**, and z is called the **logit**.

Attempt 3: Logistic Activation Function

The problem:

(plot of \mathcal{L}_{SE} as a function of z)



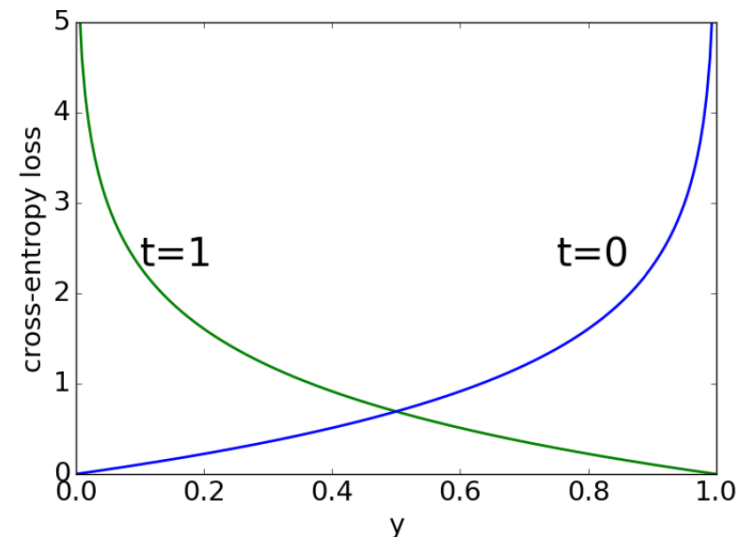
$$\frac{\partial \mathcal{L}}{\partial w_j} = \frac{\partial \mathcal{L}}{\partial z} \frac{\partial z}{\partial w_j}$$
$$w_j \leftarrow w_j - \alpha \frac{\partial \mathcal{L}}{\partial w_j}$$

- In gradient descent, a small gradient (in magnitude) implies a small step.
- If the prediction is really wrong, shouldn't you take a large step?

Logistic Regression

- Because $y \in [0, 1]$, we can interpret it as the estimated probability that $t = 1$.
- The pundits who were 99% confident Clinton would win were much more wrong than the ones who were only 90% confident.
- **Cross-entropy loss** captures this intuition:

$$\begin{aligned}\mathcal{L}_{\text{CE}}(y, t) &= \begin{cases} -\log y & \text{if } t = 1 \\ -\log(1 - y) & \text{if } t = 0 \end{cases} \\ &= -t \log y - (1 - t) \log(1 - y)\end{aligned}$$



Logistic Regression

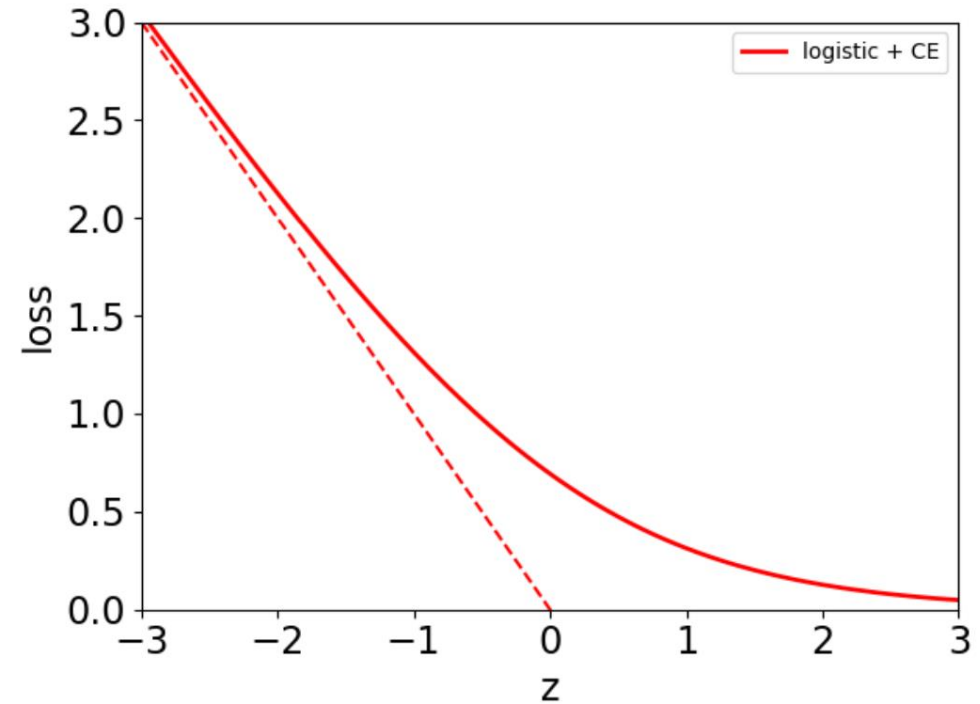
Logistic Regression:

$$z = \mathbf{w}^\top \mathbf{x} + b$$

$$y = \sigma(z)$$

$$= \frac{1}{1 + e^{-z}}$$

$$\mathcal{L}_{\text{CE}} = -t \log y - (1 - t) \log(1 - y)$$



Logistic Regression

- Problem: what if $t = 1$ but you're really confident it's a negative example ($z \ll 0$)?
- If y is small enough, it may be **numerically zero**. This can cause very subtle and hard-to-find bugs.

$$y = \sigma(z) \qquad \Rightarrow y \approx 0$$

$$\mathcal{L}_{\text{CE}} = -t \log y - (1 - t) \log(1 - y) \quad \Rightarrow \text{computes } \log 0$$

- Instead, we combine the activation function and the loss into a single **logistic-cross-entropy** function.

$$\mathcal{L}_{\text{LCE}}(z, t) = \mathcal{L}_{\text{CE}}(\sigma(z), t) = t \log(1 + e^{-z}) + (1 - t) \log(1 + e^z)$$

- Numerically stable computation:

$$E = t * \text{np.logaddexp}(0, -z) + (1-t) * \text{np.logaddexp}(0, z)$$



Weight updates

Comparison of gradient descent updates:

- Linear regression:

$$\mathbf{w} \leftarrow \mathbf{w} - \frac{\alpha}{N} \sum_{i=1}^N (y^{(i)} - t^{(i)}) \mathbf{x}^{(i)}$$

- Logistic regression:

$$\mathbf{w} \leftarrow \mathbf{w} - \frac{\alpha}{N} \sum_{i=1}^N (y^{(i)} - t^{(i)}) \mathbf{x}^{(i)}$$

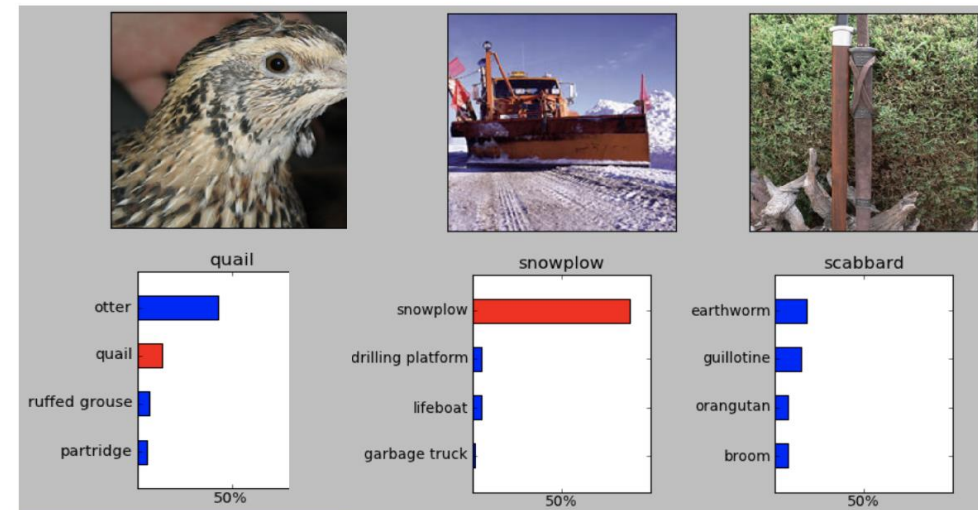
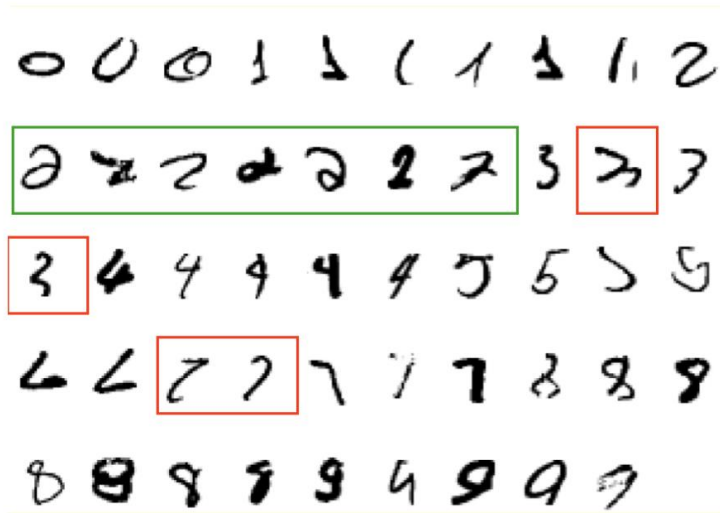
Using activation function

$$y = \frac{1}{1 + e^{-z}}$$

Multiclass Classification

Multiclass Classification

- What about classification tasks with more than two categories?



Multiclass Classification

- Targets form a discrete set $\{1, \dots, K\}$.
- It's often more convenient to represent them as **one-hot vectors**, or a **one-of-K encoding**:

$$\mathbf{t} = \underbrace{(0, \dots, 0, 1, 0, \dots, 0)}_{\text{entry } k \text{ is } 1}$$

Multiclass Classification

- Now there are D input dimensions and K output dimensions, so we need $K \times D$ weights, which we arrange as a **weight matrix \mathbf{W}** .
- Also, we have a K -dimensional vector **\mathbf{b}** of biases.
- Linear predictions:

$$z_k = \sum_j w_{kj} x_j + b_k$$

- Vectorized:

$$\mathbf{z} = \mathbf{W}\mathbf{x} + \mathbf{b}$$

Activation Function

- A natural activation function to use is the **softmax function**, a multivariable generalization of the logistic function:

$$y_k = \text{softmax}(z_1, \dots, z_K)_k = \frac{e^{z_k}}{\sum_{k'} e^{z_{k'}}}$$

- The inputs z_k are called the **logits**.
- Properties:
 - Outputs are positive and sum to 1 (so they can be interpreted as probabilities)
 - If one of the z_k 's is much larger than the others, $\text{softmax}(\mathbf{z})$ is approximately the argmax. (So really it's more like "soft-argmax".)

Loss function

- If a model outputs a vector of class probabilities, we can use cross-entropy as the loss function:

$$\begin{aligned}\mathcal{L}_{\text{CE}}(\mathbf{y}, \mathbf{t}) &= - \sum_{k=1}^K t_k \log y_k \\ &= -\mathbf{t}^\top (\log \mathbf{y}),\end{aligned}$$

where the log is applied elementwise.

- Just like with logistic regression, we typically combine the softmax and cross-entropy into a **softmax-cross-entropy** function.

Softmax Regression

- Softmax regression:

$$\mathbf{z} = \mathbf{W}\mathbf{x} + \mathbf{b}$$

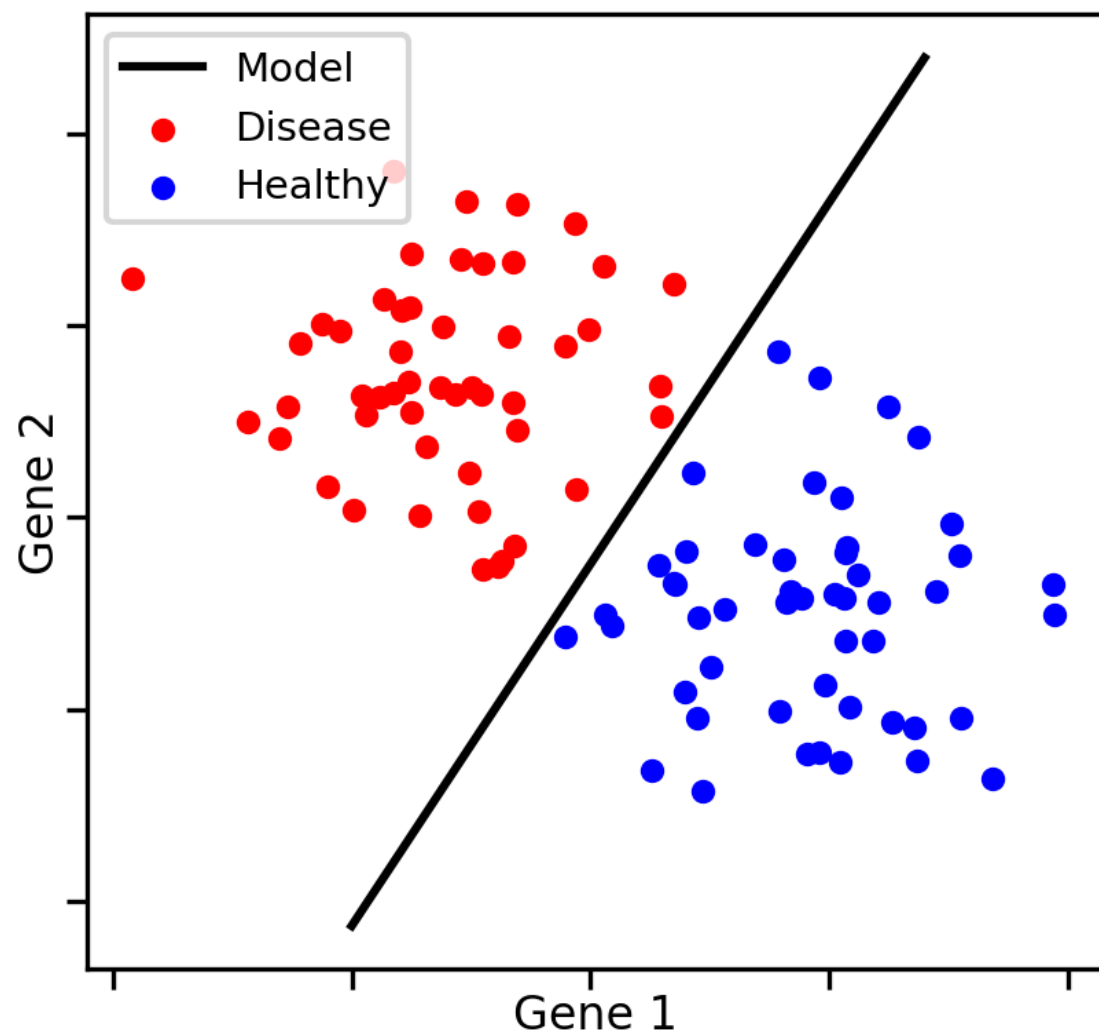
$$\mathbf{y} = \text{softmax}(\mathbf{z})$$

$$\mathcal{L}_{\text{CE}} = -\mathbf{t}^\top (\log \mathbf{y})$$

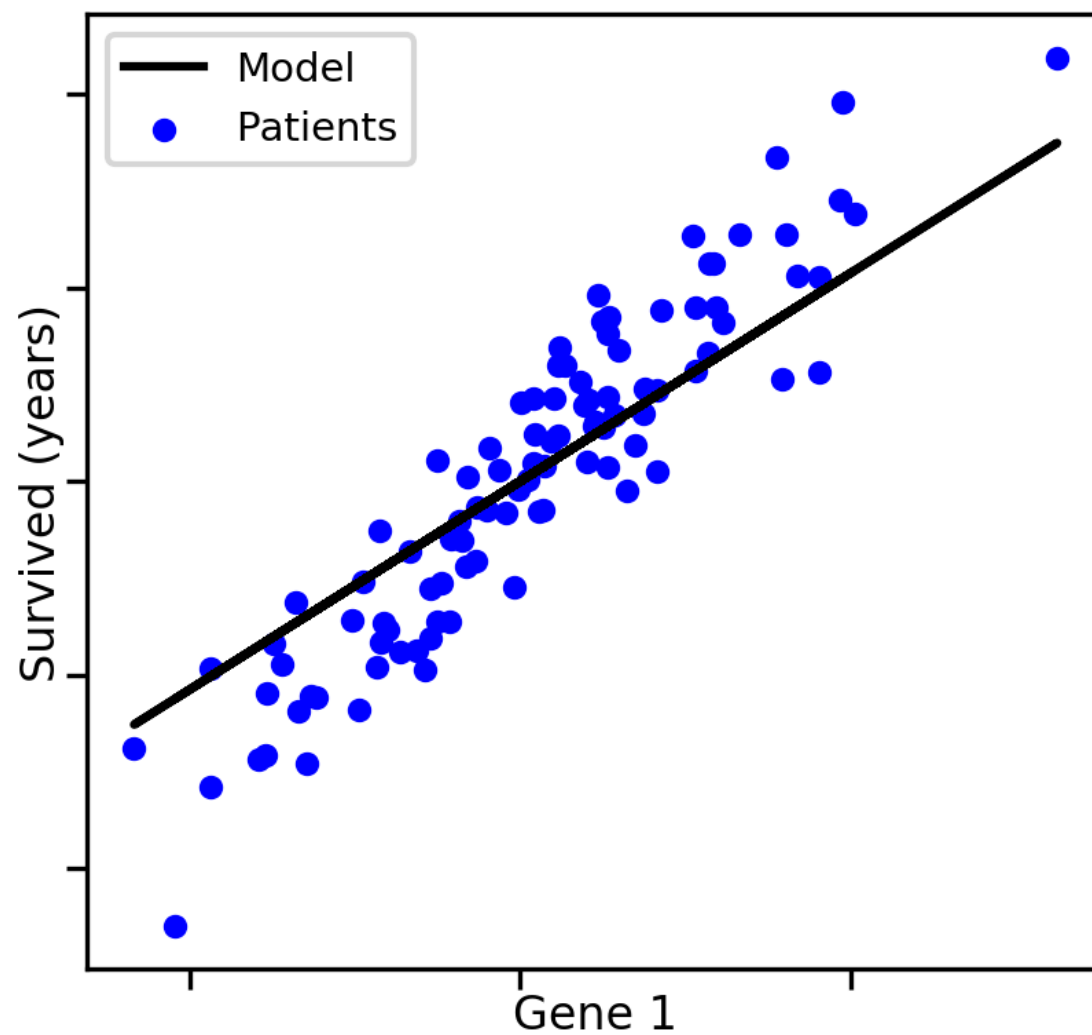
- Gradient descent updates

$$\frac{\partial \mathcal{L}_{\text{CE}}}{\partial \mathbf{z}} = \mathbf{y} - \mathbf{t}$$

Classification



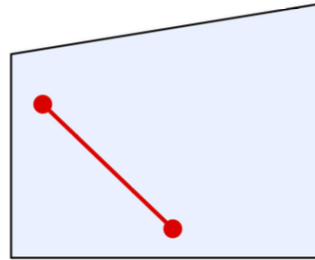
Regression



Convexity

Convexity

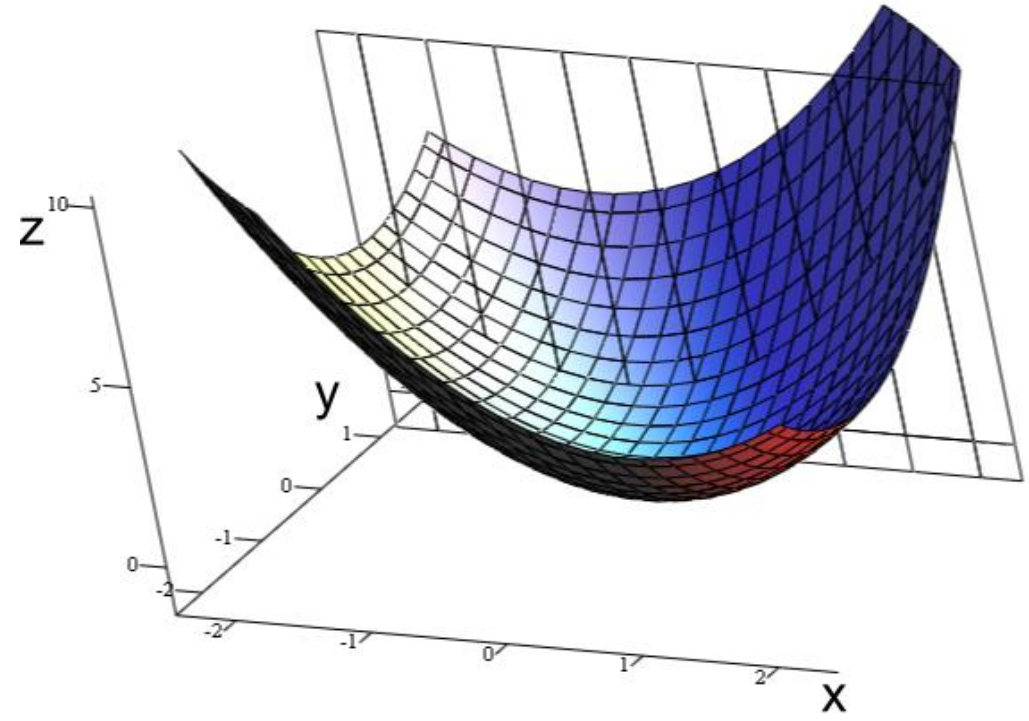
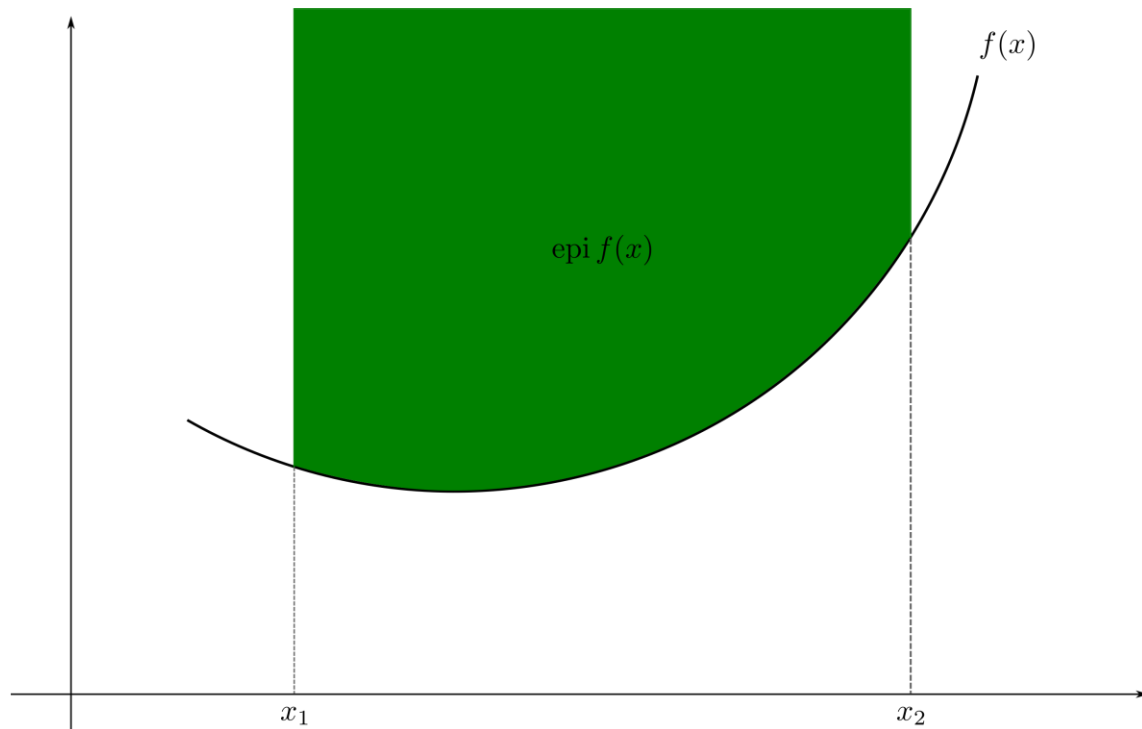
Convex Sets



- A set \mathcal{S} is **convex** if any line segment connecting points in \mathcal{S} lies entirely within \mathcal{S} . Mathematically,

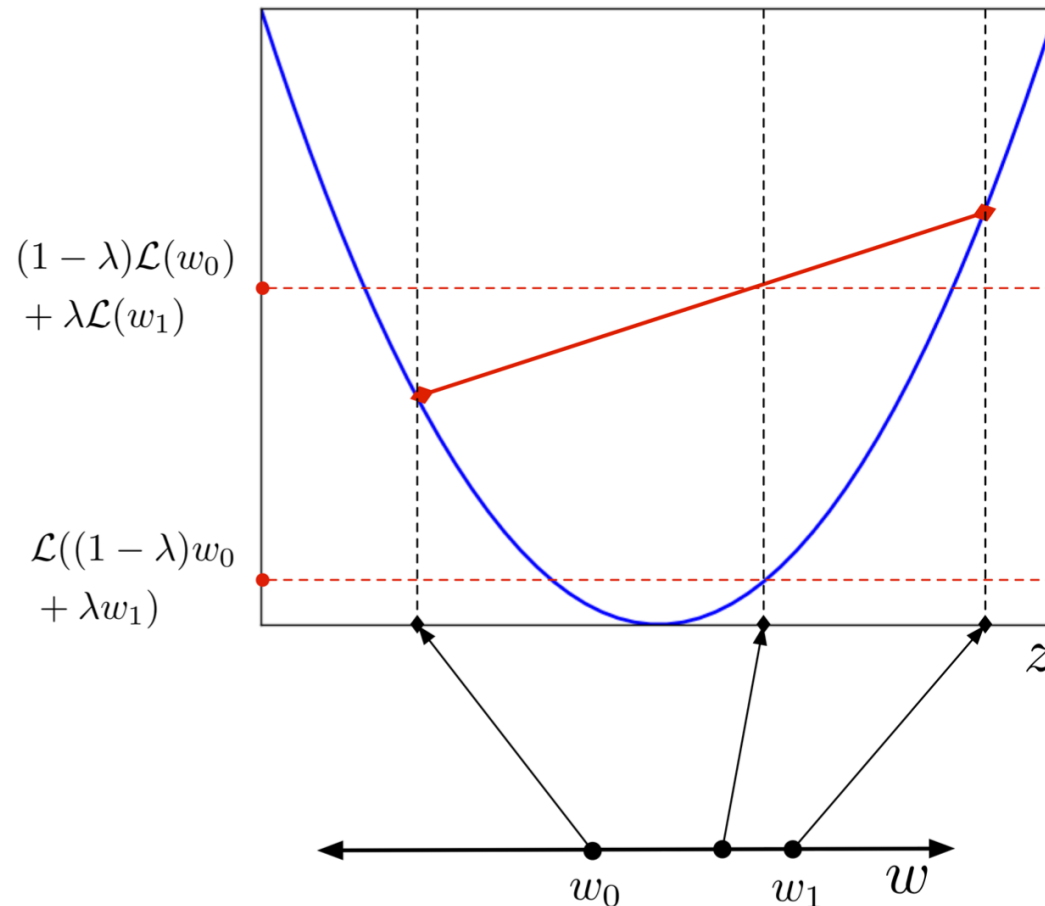
$$\mathbf{x}_1, \mathbf{x}_2 \in \mathcal{S} \implies \lambda \mathbf{x}_1 + (1 - \lambda) \mathbf{x}_2 \in \mathcal{S} \quad \text{for } 0 \leq \lambda \leq 1.$$

2D, 3D Convex

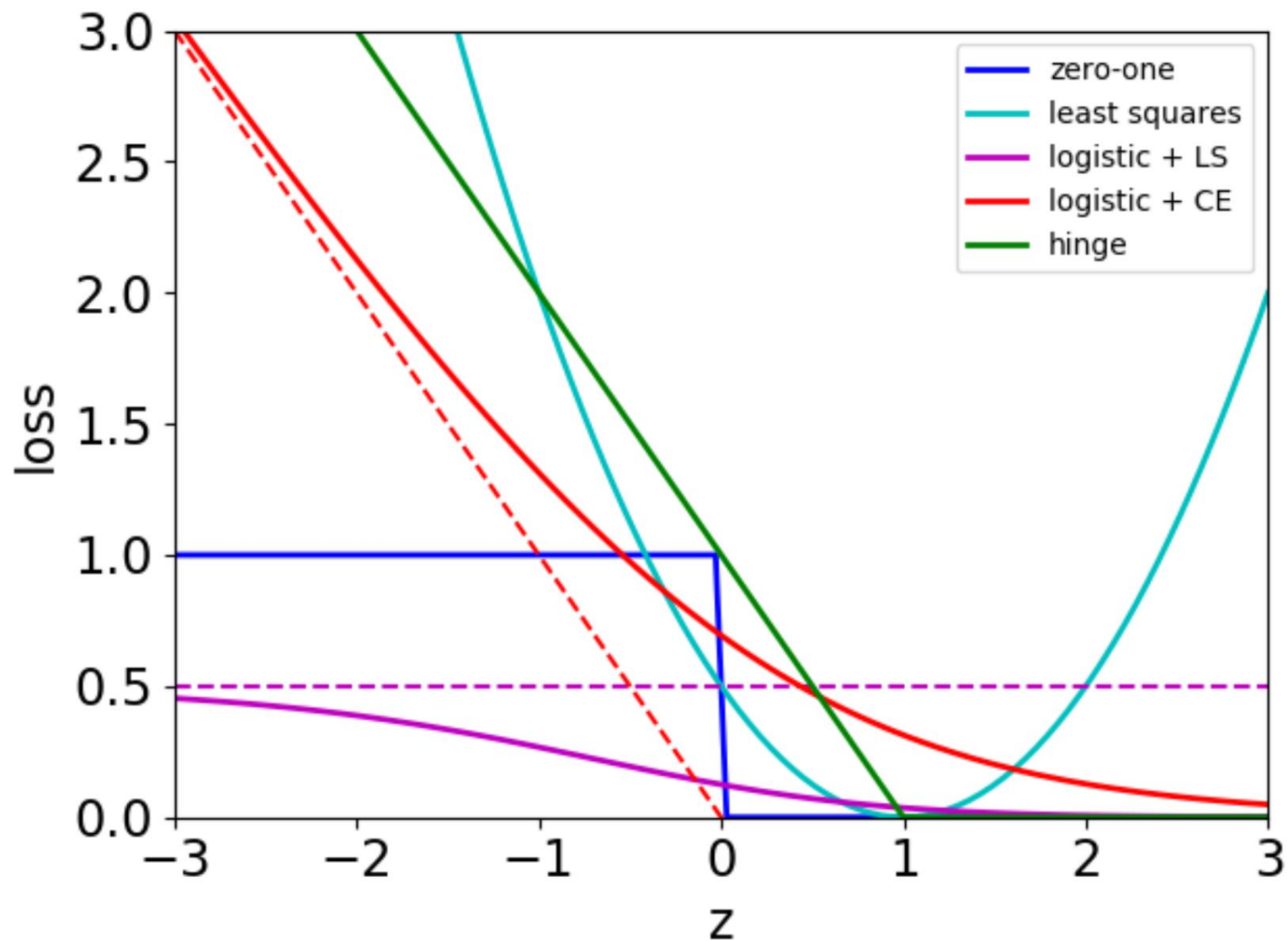


Convex Function

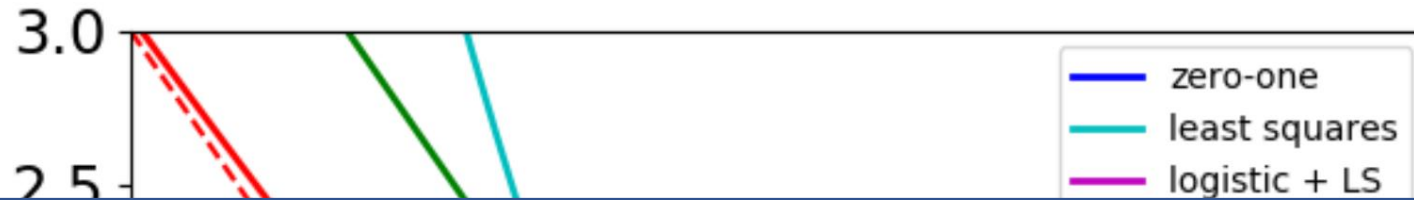
- We just saw that the least-squares loss function $\frac{1}{2}(y - t)^2$ is convex as a function of y
- For a linear model, $z = \mathbf{w}^\top \mathbf{x} + b$ is a linear function of \mathbf{w} and b . If the loss function is convex as a function of z , then it is convex as a function of \mathbf{w} and b .



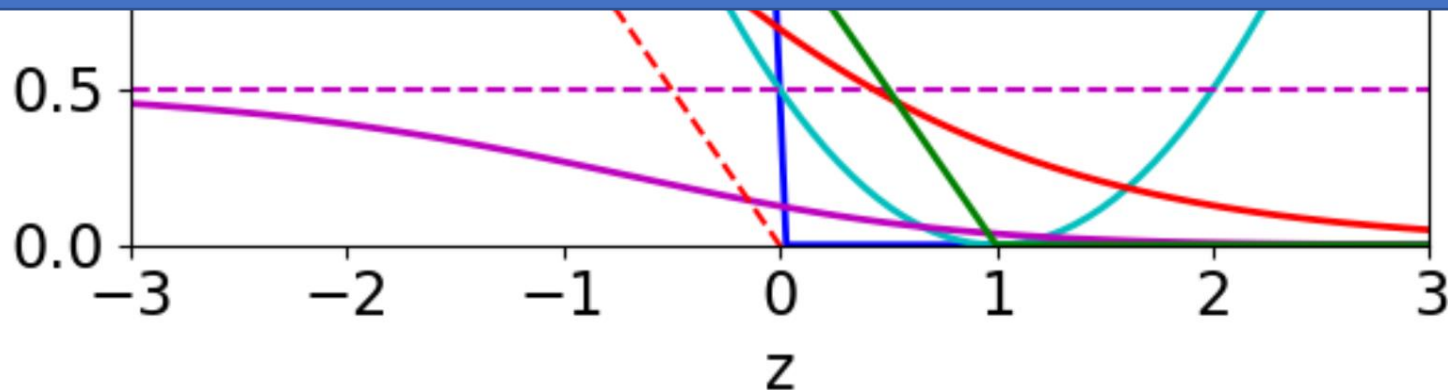
Which loss functions are convex?



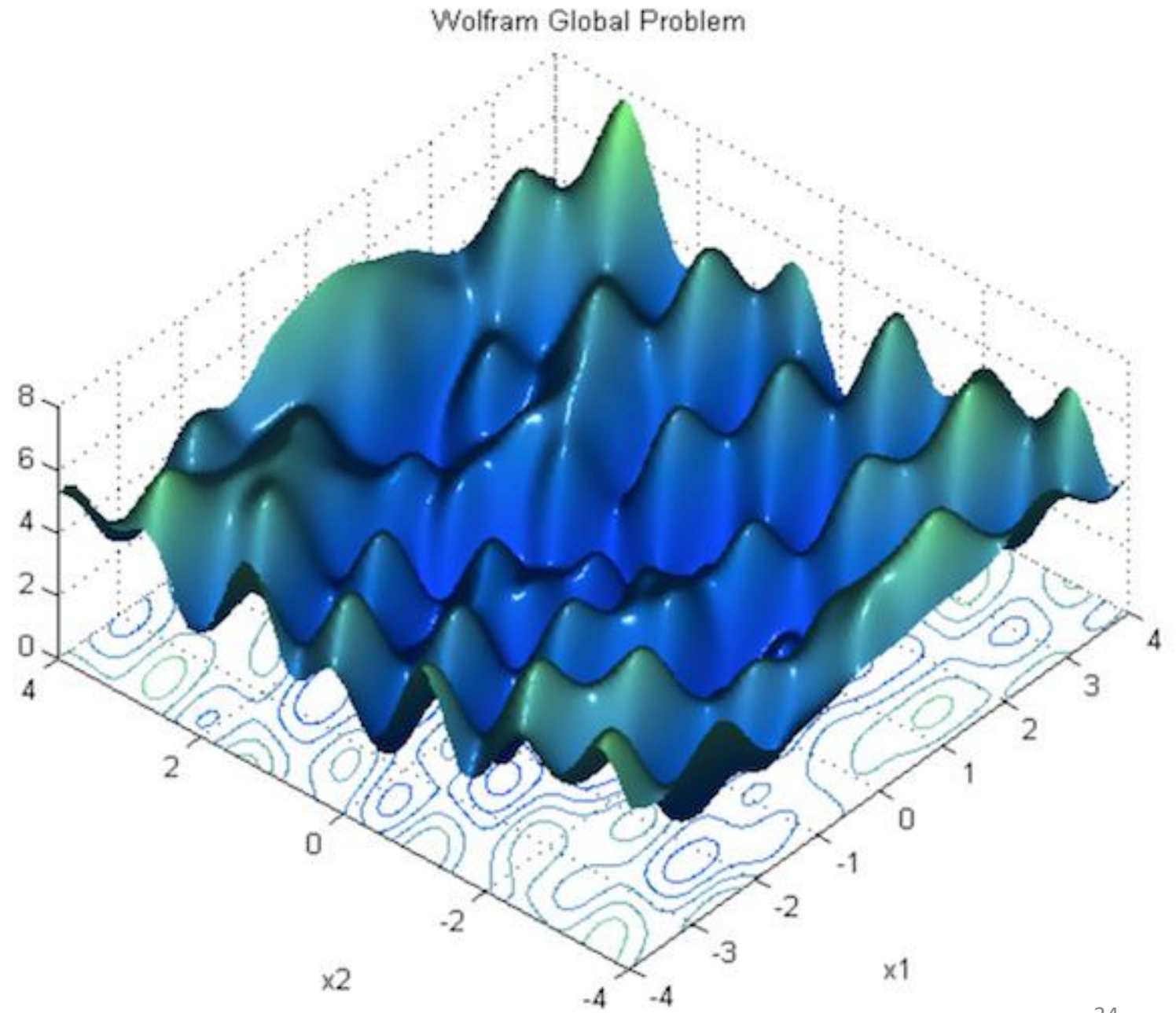
Which loss functions are convex?



All critical points are minima
Gradient descent finds the optimal solution



Non-Convex



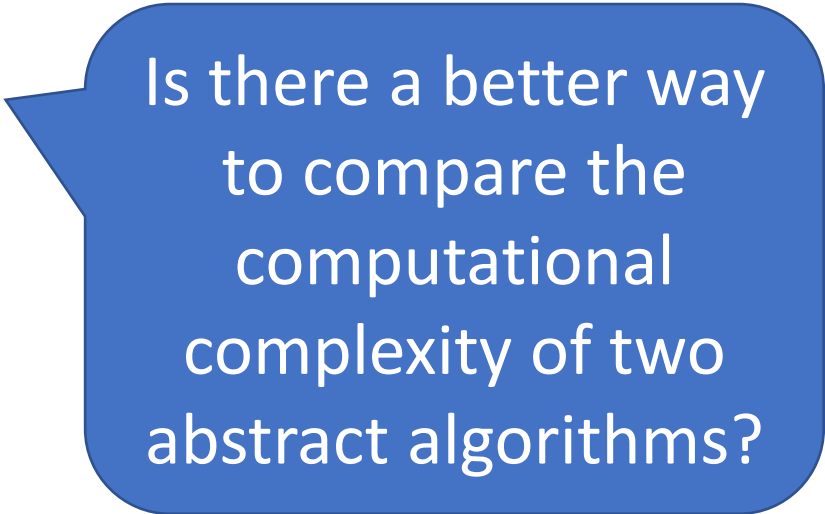
Big-O Notation

How efficient is
your code?

- We can always profile it
 - See classes that dominate memory usage
 - See methods that dominate runtime

Caveats of Complexity Analysis via Empirical Timing

- Hardware: processor(s), memory, cache, etc.
- Programs running in the background
- Implementation dependent
- Choice of input
- Which inputs to test



Is there a better way
to compare the
computational
complexity of two
abstract algorithms?

Algorithm Analysis

As the “size” of an algorithm’s input grows:

- Time: How much longer does it run?
- Space: How much memory does it use?

How do we answer these questions?

For now, we will focus on time only.

Basic Lesson

Evaluating an implementation?

Use empirical timing (how long it takes to run)

Evaluating an algorithm?

Use asymptotic analysis (study limiting behavior)

Assumptions in Asymptotic Algorithm Analysis

Basic operations take **constant time**

- Arithmetic
- Assignment
- Access one array index
- Comparing two simple values (is $x < 3$?)

Other operations are summations or products

- Consecutive statements are summed
- Loops are (cost of loop body) \times (number of loops)

What about conditionals?

Worst-Case Analysis

- In general, we are interested in three types of performance
 - Best-case / Fastest
 - Average-case
 - Worst-case / Slowest
- When determining worst-case, we tend to be pessimistic
 - If there is a conditional, count the branch that will run the slowest
 - This will give a loose bound on how slow the algorithm may run

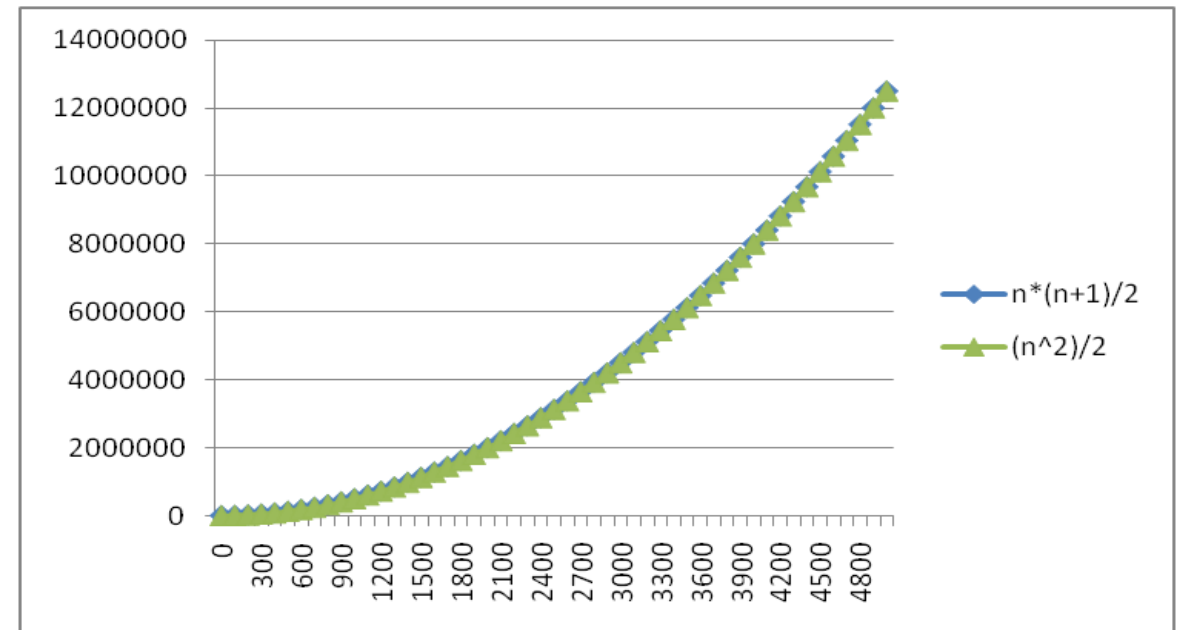
No Need To Be So Exact

Constants do not matter

- Consider $6N^2$ and $20N^2$
- When $N \gg 20$, the N^2 is what is driving the function's increase

Lower-order terms are also less important

- $N*(N+1)/2$ vs. just $N^2/2$
- The linear term is inconsequential



We need a better notation for performance that focuses on dominant terms only

A Big Warning

Variables that change with the input size are not constants:

n^3 is $O(n^2)$ is FALSE

3^n is $O(2^n)$ is FALSE

When in doubt, refer to the rigorous definition of Big-Oh

Big Oh: Common Categories

From fastest to slowest

$O(1)$	constant (or $O(k)$ for constant k)
$O(\log n)$	logarithmic
$O(n)$	linear
$O(n \log n)$	" $n \log n$ "
$O(n^2)$	quadratic
$O(n^3)$	cubic
$O(n^k)$	polynomial (where k is constant)
$O(k^n)$	exponential (where constant $k > 1$)

Complexity Analysis of Code

- `for (int i=0; i<N; i++)`
 `sum += i;`

- $O(N)$ – linear time

- `for (int i=0; i<N; i++)`
 `for (int j=0; j<N; j++)`
 `sum += i + j;`

- $O(N^2)$ – quadratic time

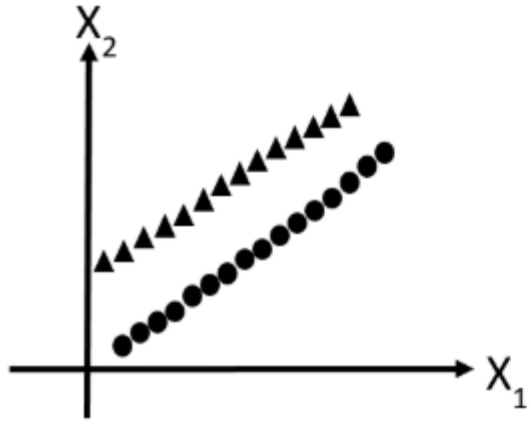
- `for (int i=0; i<N; i++)`
 `for (int j=0; j<N; j++)`
 `for (int k=0; k<N; k++)`
 `sum += i + j + k;`

- $O(N^3)$ – cubic time



Review /Exercises

Here is a dataset with two labeled classes:



We first apply PCA on the entire dataset and obtain the first principal component (PC1) and second principal component (PC2).

If we were to use these components to classify two classes, which component is the most useful in discriminating between classes? Explain your thinking.

A fraud detection model...

- Correctly identifies 472 events (true positives);
- Incorrectly identifies 17 events (false positives);
- Correctly rejects 312159 events (true negatives);
- Incorrectly rejects 22 events (false negatives);

1. How many actual frauds are there in the data set?
2. Calculate recall. What does this number signify?
3. Calculate precision. What does this number signify?
4. Calculate the F1-score. What does this number signify?
5. Calculate specificity. What does this number signify?
6. If you were asked to further improve the model by hyperparameter optimization and had to choose a single metric to maximize, which of the above would you choose (recall vs precision vs F1-score vs specificity)? Why?

The impact of viral diseases such as COVID-19 is very much top of mind nowadays.

Let's assume 0.1% of Earth's population is infected by a particular virus.

A randomly selected person tests positive for the virus, and asks their doctor "what is the accuracy of these tests?" The doctor replies "This test correctly identifies 99% of the people who have the virus, and incorrectly identifies 1% of the people who don't have the virus."

Calculate the probability of a person being infected with the virus, if the result of this test is positive?

- 1. What is the difference between multiclass classification and binary classification?
- 2. What is the shape of weights in both cases?
- 3. In binary linear classification, we used the linear model $y = w^T x + b$. What was the problem in using a squared error loss $L(y, t) = \frac{1}{2}(y - t)^2$?
- 4. Is it possible to solve that problem using an absolute value loss $L(y, t) = |y - t|$?

Consider this binary classification problem:

x_1	x_2	x_3	t
0	0	0	0
0	1	0	1
0	1	1	0
1	0	0	0
1	1	1	1

Your job is to find a linear classifier with weights w_1, w_2, w_3 and b which correctly classifies all of these training examples.

1. Give the set of linear inequalities the weights and bias must satisfy.
2. Give a set of the weights and bias that correctly classifies all the training examples, or provide a mathematical proof that finding such parameters is not possible.

Write a Numpy code to implement a 1-nearest-neighbour classifier.
Your inputs are:

- Training data $\mathbf{X}_{N \times D}$: where N is the number of samples and D is the number of features.
- Labels $\mathbf{Y}_{N \times 1}$: contains the corresponding integer label for each training example. (You may assume the labels are integers from 1 to K.)
- Query (test) vector $\mathbf{Q}_{1 \times D}$: you will return the predicted class for this vector.

Compute covariance matrix for the dataset below:

Datapoint	X1	X2
1	1	-1
2	0	0
3	1	0
4	0	-1
5	1	-1
6	1	-1
7	0	0
8	0	0

Plot a graph where x-axis is the *test set size* and y-axis show (i) the *training error* and (ii) the *test error*. Assume a well-trained, typical machine learning model with fixed complexity and a fixed training set. Describe your graph.

Using row echelon technique, show if the vectors below are linearly independent or not.

$$A=[1,0,1,-1,0] ; B=[2,-1,0,1,0] ; C=[-2,0,1,0,0]; D=[1,-2,0,1,1]$$

Which of the following statements is true?

- a) In k -fold cross-validation, training data is divided into k folds, then the model is trained on k folds and validated on 1 fold, for a total of $k-1$ times (splits).
- b) In k -fold cross-validation, training data is divided into k folds, then the model is trained on k folds and validated on 1 fold, for a total of k times (splits).
- c) In k -fold cross-validation, training data is divided into k folds, then the model is trained on $k-1$ folds and validated on 1 fold, for a total of k times (splits).
- d) In k -fold cross-validation, training data is divided into k folds, then the model is trained on $k-1$ folds and validated on 1 fold, for a total of $k-1$ times (splits).

When performing PCA, the goal is to accomplish which of the following?

- a) Maximize the variance of the primary components and maximize the residuals.
- b) Minimize the variance of the primary components and minimize the residuals.
- c) Maximize the variance of the primary components and minimize the residuals.
- d) Minimize the variance of the primary components and maximize the residuals.

Is the matrix $A = \begin{pmatrix} 1 & 2 & -1 \\ 1 & 0 & 3 \\ -2 & -4 & 2 \end{pmatrix}$ invertible?

We have $x = [1, 1, 2]^T \in \mathbb{R}^3$ and $y = [1, 0, 3]^T \in \mathbb{R}^3$. What is the angle between vectors?

Calculate the Jacobian $\mathbf{J}_F(x_1, x_2, x_3)$ of the function $\mathbf{F} : \mathbb{R}^3 \rightarrow \mathbb{R}^4$, which has components:

$$y_1 = x_1 + 2x_3^2 ; y_2 = x_1 \sin x_2 ; y_3 = x_2 \exp(x_3) ; y_4 = x_1 + x_2$$

Consider functions $f : \mathbb{R}^3 \rightarrow \mathbb{R}^1$ and $\mathbf{x} : \mathbb{R}^1 \rightarrow \mathbb{R}^3$, where:

$$f(\mathbf{x}) = x_1 + 2x_2^2 + x_3 ; \mathbf{x}(t) = \begin{bmatrix} t \\ 3t \\ \exp(t) \end{bmatrix}$$

Calculate the gradient $\frac{df}{dt}$ using the chain rule.

What is a learning rate? Explain what happens if it is set too high or too low (you may use a schematic/drawing if helpful).

In general, is using a gradient descent algorithm a good choice for finding optimal hyperparameters? Why or why not?