REGULARISATION

Dr. Brian Mc Ginley

REGULARISATION

- Another term that applies in general to "most" Machine Learning systems is the idea of Regularisation.
- Definition: penalize the model if its weights are too high
  - Regularisation is a technique used in machine learning models to reduce overfitting, improve model generalization, and manage model complexity. It achieves this by adding an additional penalty term in the error function. The additional term controls for excessively fluctuating model parameters so that the coefficients don't take extreme values.

coefficients = model weights

- This technique of keeping a check or reducing the value of coefficients are called shrinkage methods or weight decay in case of neural networks. or regression in our case

The main rationale is, is that you if you have weights that are too big, especially for some feature in particular,
it can become overly sensitive to that feature and a small change in that feature can have a big swing on the output value.
So regularisation is all about desensitising the network or desensitising the model to any particular feature so that it looks at all features in a little bit more of a balanced way and that it's not overtraining or overfishing from one particular feature.

REGULARISATION

- Some Points:
  - Regularisation is a more general way of avoiding overfitting.
  - Overfitting means your model has high-variance
    - Variance is the amount your model will change if you change the training data
    - More parameters and flexibility cause high-variance more weights
  - Regularisation works be adding a term to the loss function which adds loss if the model is not of a preferred type.
    - In particular, we don't want any weights to become very large. it rewards smaller weights
    - A large weight can change by a large amount very quickly (very sensitive to small changes in the associated feature). desensitising the network to any particular input.
    - As differences in the magnitude of weights can quickly spiral out of control and individual weights influence can quickly exceed their importance.
    - For this reason, we add a penalty to the loss function that is related to the size of the weights.

If 3D: $d = \sqrt{(x_2-x_1)^2+(y_2-y_1)^2+(z_2-z_1)^2}$

So if >3D: $d = \sqrt{(y_1-x_1)^2+(y_2-x_2)^2+(y_3-x_3)^2}$

MEAN SQUARED ERROR

$d = \sqrt{\sum_{i=1}^{n}(y_i-x_i)^2}$

if >3D:

$x_1, y_1, z_1 \rightarrow x_1, x_2, x_3$

$x_2, y_2, z_2 \rightarrow y_1, y_2, y_3$

an extension of Pythagoras Theorem

$$MSE(\boldsymbol{w}) = \frac{1}{2m}\sum_{i=1}^{m}(y^i - f_{\boldsymbol{w}}(\boldsymbol{x}^i))^2$$

from sigma onwards,
this is an L2 norm of the errors.
y true - y predict

- It can happen that when optimising this Loss function, some of the weights may become very large.
- That means the features corresponding to these large weights will increase in their importance it's memorizing certain features
- This can cause overfitting.
- So, what can we do with these large weights?

So to desensitise and to try and get a more a model with less variance and less sensitivity,
we can try and apply a penalty for these large weights.

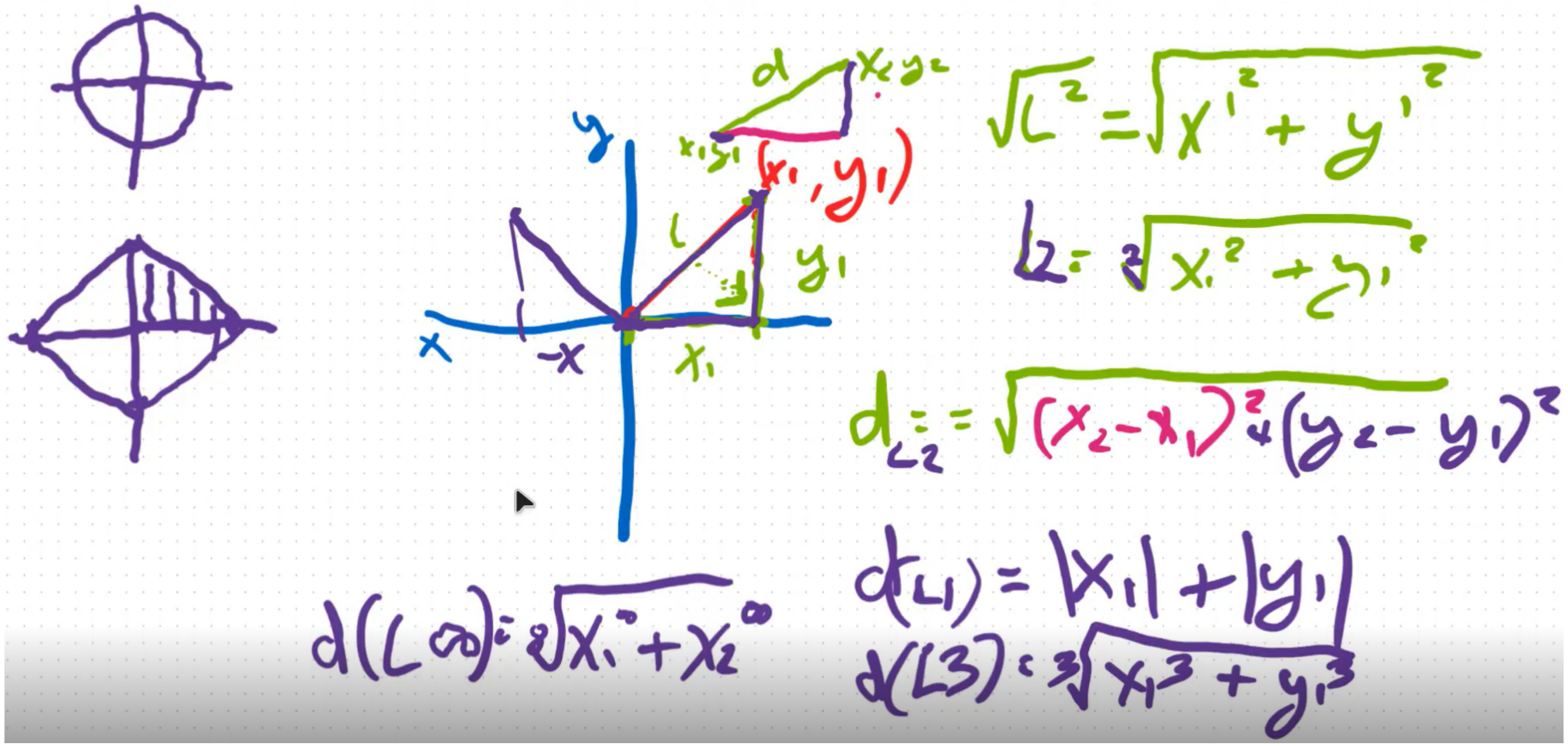
So let's have a look at how mean squared error is augmented or this loss metric is augmented to penalise large weights.

L1 AND L2 NORM

L2 = Euclidean distance (hypothenuse),
L1 = Manhattan distance (jalan jauh)

L2 circle
euclid

L1 square
manhattan



y
x
-x
x_1
y_1
(x_1, y_1)
$x_1 y_1$
d
$x_2 y_2$
$\sqrt{L^2} = \sqrt{x_1^2 + y_1^2}$
$L_2 = \sqrt[2]{x_1^2 + y_1^2}$
$d_{L2} = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$
$d(L_1) = |x_1| + |y_1|$
$d(L_\infty) = \sqrt[\infty]{x_1^\infty + x_2^\infty}$
$d(L_3) = \sqrt[3]{x_1^3 + y_1^3}$

L1 and L2 norms can be extended to higher dimensions. D =square root [(x2-xi)2 +(y2-y1)2 +(z2-z1)2]
Beyond 3D, we use x1,x2, x3, x4....

MEAN SQUARED ERROR - L2 (RIDGE REGRESSION)

- We add a regularisation term (λ) to our Loss function that penalises large weights

lambda is scaling factor. the bigger Lambda is, the more focus and the more emphasis is going to be placed upon minimising these weights.

$$L(\boldsymbol{w}) = MSE(\boldsymbol{w}) + \lambda \sum_{j=1}^{m} w_j^2$$

the bigger the way it is, the bigger this penalty gets
Anything squared is big

- This means that if any particular w_j is large, the overall Loss will be large too.
- This is known as the L2 norm.
- It does not matter if we are using MSE or the logistic regression loss function, we can add on the regularisation term to any loss metric.
- Here is the MSE with regularisation fully written out:

$$L(\boldsymbol{w}) = \frac{1}{2m} \sum_{i=1}^{m} (y^i - f_{\boldsymbol{w}}(\boldsymbol{x}^i))^2 + \lambda \sum_{j=1}^{n} w_j^2$$

- where n is the number of features and m is sample size. Notice we do not include the bias term (w_0)

the bias term is just a static bias that's added to kind of improve the performance of the network, but it's not actually multiplied by any input feature.

We're adding extra, essentially bias, extra information in the model of creation.But what that does in this trade off that always exists in machine learning is by adding bias, we're reducing the variance.So that means that our model will be less sensitive to the training because there's no big weights in there that are having a big impact.

L2 REGULARISATION

- The effect of regularisation is to make it so the model prefers to learn small weights, all other things being equal.
- This means that the trained model will be a slightly worse fit for the training data but will avoid overtraining (adds bias (penalty), reduces variance)
- Large weights will only be allowed if they considerably improve the first part of the cost function.
- The relative importance of the two elements of the compromise depends on the value of λ:
  - when λ is small, we prefer to minimize the original cost function when λ is large we prefer small weights.
  - Large λ moves in the direction of underfitting, and small λ moves in the direction of overfitting.
- So it is another hyperparameter that we have to choose - perhaps via cross-validation.
- Additionally Gradient Descent applies just as it does without regularisation, it just has an extra term when calculating the updates/gradients.

BEFORE WAS RIDGE REGRESSION (WITH L2 NORM)

LASSO IS REGULARISATION WITH L1

LASSO REGRESSION

Ridge (L2) also tries to force weights towards 0, but the L1 norm or lasso regression has a special feature which tends to push coefficients to 0. But where it's really useful is that it increases sparsity.

- LASSO Regression is very similar to Ridge Regression except that it takes the absolute value of the coefficients/weights (rather than squaring them).
- This is known as the L1 norm (Manhattan Distance) vs. L2 Norm (Euclidean Distance)

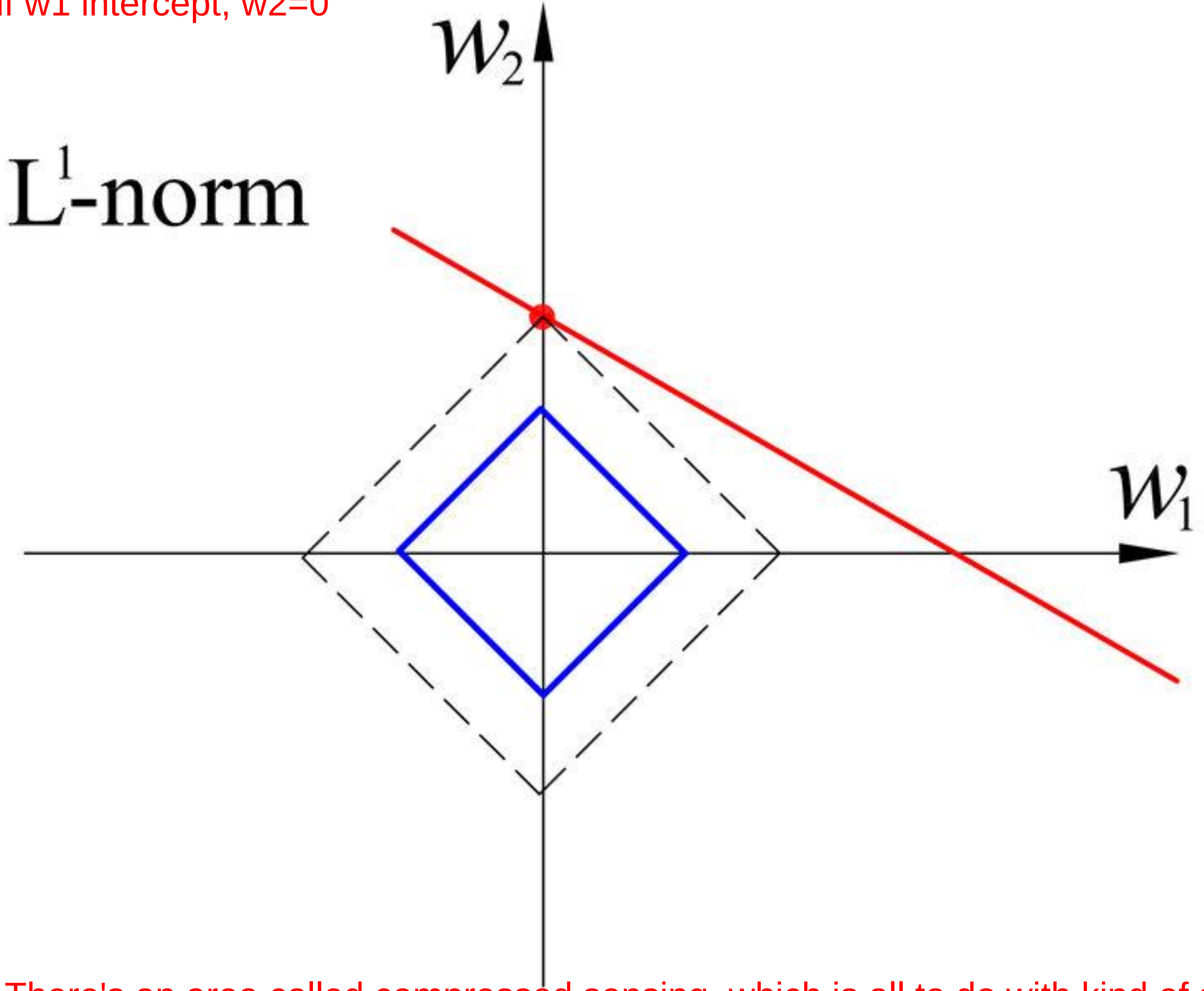
many features highly correlated (multicollinearity) means many redundant data

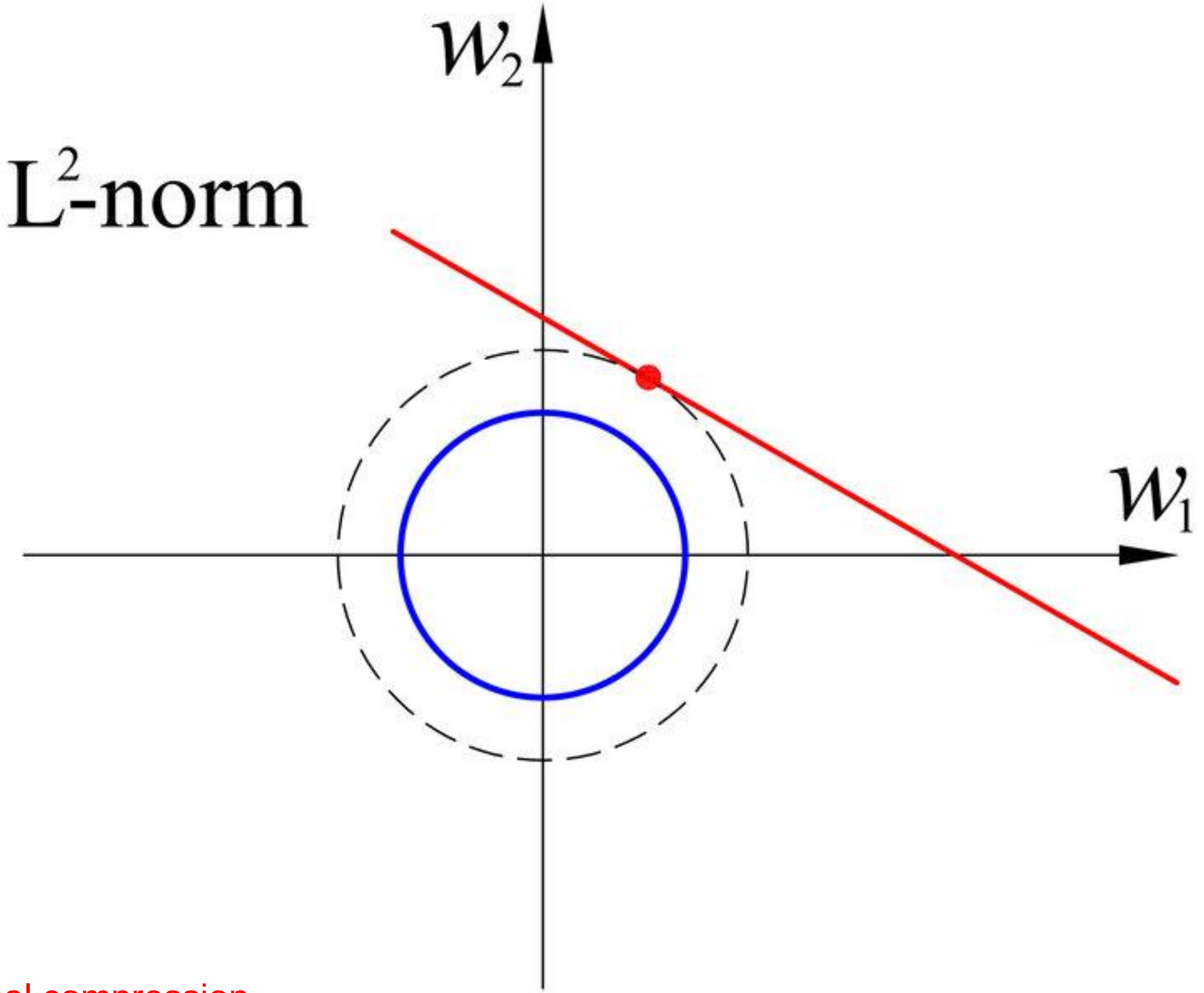
$$L(\boldsymbol{w}) = \frac{1}{2m}\sum_{i=1}^{m}(y^i - f_{\boldsymbol{w}}(\boldsymbol{x}^i))^2 + \lambda\sum_{j=1}^{m}|w_j|$$

- The main difference is that Lasso Regression can exclude non-relevant features from model (increasing sparsity) (e.g. biomedical dataset with irrelevant biomarkers).
- Ridge regression performs better when the variables are known to be useful.

with L1 norm,as you expand out the model to try and or expand out your coefficients to try and match a model, what you'll find, because it has this pointiness along each of the axes, you'll find that the point strikes or is the intercept that meets first the desired model. if w2 intercept, w1=0.
if w1 intercept, w2=0

As you try and expand out to try and fit to some sort of model, if you expand the L2 norm, You will come to a solution that has a significant value for both W1 and W2 and indeed this extends into higher dimensions





There's an area called compressed sensing, which is all to do with kind of signal compression and essentially trying to express signals with as much sparsity as possible, as much zeros in them so that they can be efficiently stored or transmitted or or whatever it is.

REGULARIZATION ADVANTAGES

- Regularization usually performs better when:
  - The data has multicollinearity (e.g. in a housing dataset): Features are highly correlated, and Regularization can prevent overfitting. eg area in sqm and sqft
  - The dataset has a large number of features (especially when there are more features than observations) (e.g. gene expression datasets): Regularization can prevent overfitting by shrinking the coefficients and selecting the most relevant features. So, if you have limited data, Regularization helps keep model variance low. regularisation isn't so needed when you've got lots of data.
  - The dataset has some irrelevant features. Regularization helps by forcing these coefficients towards zero
  - The dataset has noise: Regularization can reduce the model's sensitivity to noise. In contrast, Linear Regression does not have this penalty mechanism and will try to fit all features, including noisy ones (which might cause worse performance). but maybe lasso can get better

if you've got some features that are completely irrelevant, linear regression doesn't really have the capacity or doesn't have the desire to move features towards zero. Well, because of this penalty term that's with your Ridge or your lasso regularisation, it helps force these coefficients to 0 in the case of L1 and towards 0 in the case of L2.