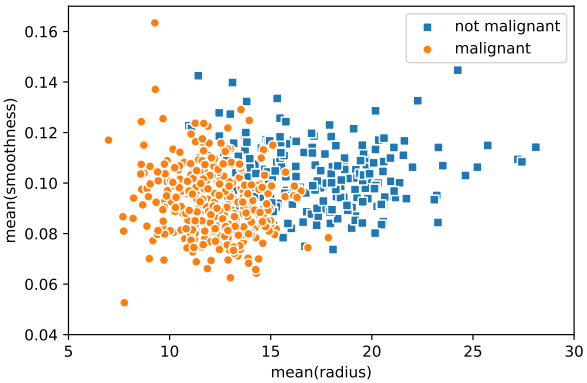
Preprocessing: Normalisation, scaling and categorical data

Herman Kamper

Feature normalisation example

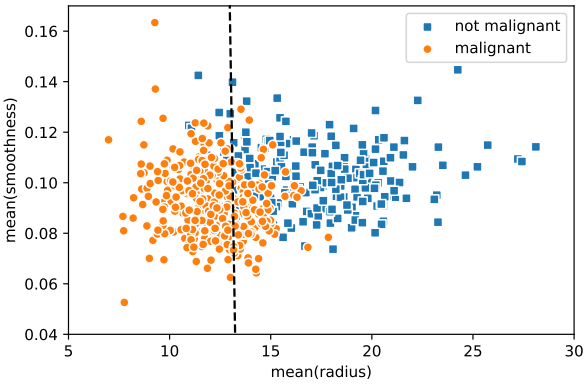
Breast cancer data:[1]



Gradients on original data (for logistic regression):

	w_0	w_1	w_2
Iteration 1000 gradients: [	-289.919	-3694.766	-26.513]
Iteration 2000 gradients: [	-246.909	-3223.000	-22.423]
Iteration 3000 gradients: [	-93.780	-1352.985	-8.034]
Iteration 4000 gradients: [	-92.243	-1332.636	-7.894]

Logistic regression on original data:



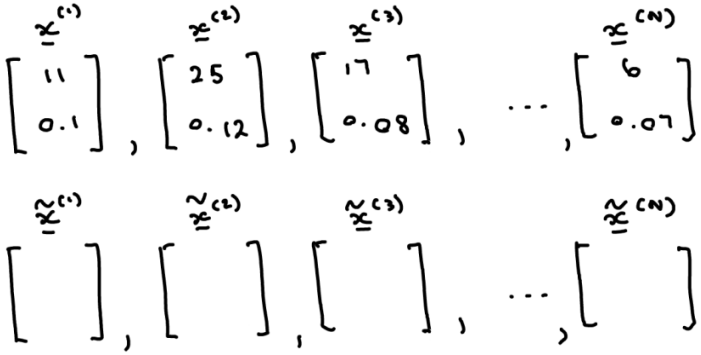
[1] Data from the UCI Machine Learning Repository.

Feature normalisation

Standardise the means and variances of the data:

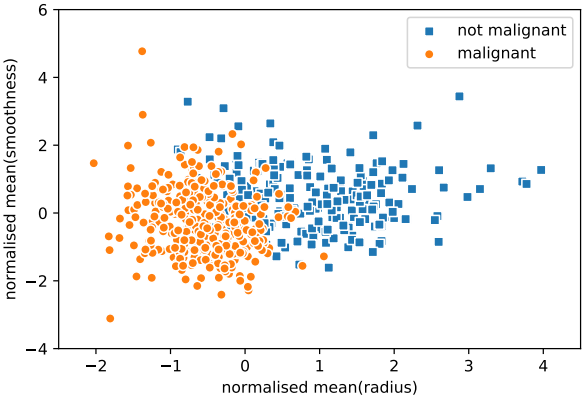
$$\tilde{x}_d^{(n)} = \frac{x_d^{(n)} - \hat{\mu}_d}{\hat{\sigma}_d}$$

where $\hat{\mu}_d$ and $\hat{\sigma}_d^2$ are, respectively, the sample mean and variance of the dth feature.



Feature normalisation example

Normalised breast cancer data:

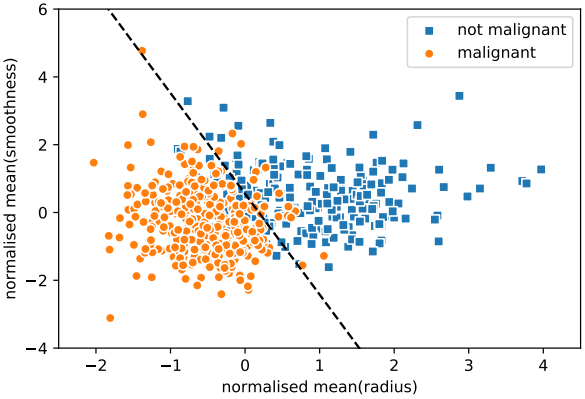


Gradients on normalised data (for logistic regression):

$$w_0 \quad w_1 \quad w_2$$

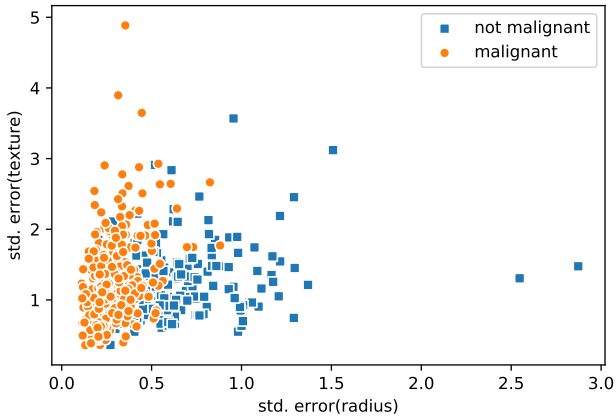
```
Iteration 1000 gradients: [ -0.525 9.179 2.472 ]
Iteration 2000 gradients: [ -0.194 3.588 0.990 ]
Iteration 3000 gradients: [ -0.096 1.752 0.486 ]
Iteration 4000 gradients: [ -0.051 0.928 0.258 ]
```

Logistic regression on normalised data:

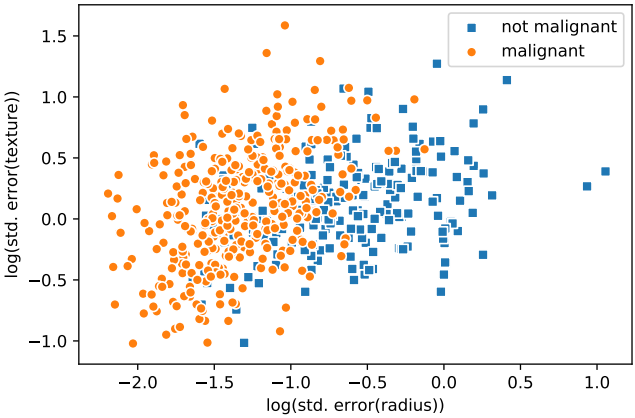


Log scaling example

Unnormalised breast cancer data:



Log-scaled breast cancer data:



Feature normalisation and scaling in practice

Feature normalisation and scaling are often a bit of an art.

You can develop an intuition as you play around with different models and optimisation algorithms.

Note: Always think about how you will apply your model to new unseen data.

Categorical output

In multiclass classification we have categorical output,
i.e. $y \in \{1, 2, \ldots, K\}$.

We can just save these target values explicitly. E.g. for softmax regression, you can write the loss as:

$$J(\mathbf{W}) = -\sum_{n=1}^{N}\sum_{k=1}^{K} \mathbb{I}\{y^{(n)} = k\} \log f_k(\mathbf{x}^{(n)}; \mathbf{W})$$

Alternatively, we can encode the target output using a *one-hot* vector:

$$\mathbf{y}^{(n)} = \begin{bmatrix} 0 & 0 & \ldots & 0 & 1 & 0 & \ldots & 0 \end{bmatrix}^\top$$

E.g. for softmax regression, you can write the loss as:

$$J(\mathbf{W}) = -\sum_{n=1}^{N}\sum_{k=1}^{K} y_k^{(n)} \log f_k(\mathbf{x}^{(n)}; \mathbf{W})$$

Note that these two formulations are mathematically exactly equivalent.

Categorical input

We might have inputs that are categorical (also called *discrete* or *qualitative* features).

E.g. someone's `occupation` might be `student`, `lecturer` or `artist`. How do we represent this?

One option is to create a new feature:

$$x = \begin{cases} 0 & \text{if } \texttt{student} \\ 1 & \text{if } \texttt{lecturer} \\ 2 & \text{if } \texttt{artist} \end{cases}$$

But this implies an ordering, which might not be true. E.g. in the above represtation, `artist` is closer to `lecturer` than to `student`.

Instead use one-hot vector (also called a *one-of-*K vector) to encode input:

Sometimes such a one-hot $\mathbf{x}$ is called a *dummy variable*.

Common misconceptions

One-hot vectors for representing inputs and outputs

Students often think that the issues with using integers to encode categorical inputs also translates to using integers to represent categorical outputs. But this isn't always the case (depending on the definitions used). E.g. in this note the use of one-hot vectors for representing outputs are exactly equivalent to using integers, as long as the loss function is adapted appropriately. But in our discussion of categorical inputs here, using an integer input feature is very different from using a one-hot vector.

Videos covered in this note

- Preprocessing 1: Feature normalisation and scaling (14 min)
- Preprocessing 2: Categorical features and categorical output (9 min)

Reading

- ISLR 3.3.1