Exercise 9

Machine Learning I

9A-1.

Let $f: \mathbb{R}^n \to \mathbb{R}$. Recall, that the real valued partial derivative can be defined with respect to the standard basis:

Prerequisites

$$\frac{\partial f}{\partial x_i} = \lim_{\substack{h \to 0 \\ h>0}} \frac{f(\mathbf{x} + h\mathbf{e}_\mathrm{i}) - f(\mathbf{x})}{h}$$

where $\mathbf{e}_i$ denotes the ith (standard) basis vector. This leads to:

$$\begin{aligned} \mathbf{units} \frac{\partial f}{\partial x_i} &= \mathbf{units} \lim_{\substack{h \to 0 \\ h>0}} \frac{f(\mathbf{x} + h\mathbf{e}_\mathrm{i}) - f(\mathbf{x})}{h} \\ &= \frac{\mathbf{units}\ f}{\mathbf{units}\ h} \end{aligned}$$

As $h\mathbf{e}_\mathrm{i}$ only adds to the $\mathbf{x}_i$-th entry and unit vectors are generally unitless, we have

$$\mathbf{units}\ h = \mathbf{units}\ x_i$$

We are now ready to investigate the units!

Gradient descent

For $\alpha \in \mathbb{R}^+$, we have the gradient update:

$$\mathbf{x}^{t+1} = \mathbf{x}^t - \alpha \nabla f$$

$$\Rightarrow \quad \mathbf{x}_i^{t+1} - \mathbf{x}_i^t = -\alpha \frac{\partial f}{\partial x_i}, \qquad \forall\, i: 1, .., n$$

From a units perspective, this leads to:

$$\mathbf{units}\ \alpha \frac{\partial f}{\partial x_i} = \mathbf{units}\ \alpha \frac{\mathbf{units}\ f}{\mathbf{units}\ x_i}$$

Therefore $\textbf{units}\ x_i = \textbf{units}\ \alpha = \frac{\textbf{units}\ x_i^2}{\textbf{units}\ f}$. We omitted the "-" in front of the α as it does not affect the units.

In the task, f is dimensionless and $\textbf{units}\ x_1 = m, \textbf{units}\ x_2 = kg$.

Therefore, the two update steps have for each dimension:

$$\textbf{units}\ \alpha \frac{\textbf{units}\ f}{\textbf{units}\ x_1} = \underbrace{\frac{m^2}{1}}_{\textbf{units}\ \alpha} \cdot \frac{1}{m}$$
$$\textbf{units}\ \alpha \frac{\textbf{units}\ f}{\textbf{units}\ x_2} = \frac{(kg)^2}{1} \cdot \frac{1}{kg}$$

As can be seen, α must have different units for both dimensions in order to compensate for the incompatible units. As α is theoretically a scalar, this would not be possible.

Momentum gradient descent

For $\gamma, \alpha \in \mathbb{R}$, we have the gradient update:

$$\mathbf{x}^{t+1} = \mathbf{x}^t - \underbrace{[\gamma \mathbf{v}_t + \alpha \nabla f]}_{\mathbf{v}_{t+1}}$$

$$\Rightarrow \quad \mathbf{x}_i^{t+1} - \mathbf{x}_i^t = -\left[\gamma \mathbf{v}_{ti} + \alpha \frac{\partial f}{\partial x_i}\right], \qquad \forall\, i: 1, .., n$$

From a units perspective, this leads to:

$$\textbf{units}\left[\gamma \mathbf{v}_{ti} + \alpha \frac{\partial f}{\partial x_i}\right] = \textbf{units}\ \gamma \cdot \textbf{units}\ \mathbf{v}_{ti} + \textbf{units}\ \alpha \frac{\partial f}{\partial x_i}$$
$$(*) \qquad = \textbf{units}\ \gamma \cdot \textbf{units}\ \mathbf{v}_{ti} + \textbf{units}\ \alpha \frac{\textbf{units}\ f}{\textbf{units}\ x_i}$$

From the definition of the gradient updates, we also have:

$$\textbf{units}\ \mathbf{v}_{ti} = \textbf{units}\ \mathbf{x}_i^t$$

Setting this equality into $(*)$, we arrive at:

$$\textbf{units}\ \gamma \cdot \textbf{units}\ \mathbf{v}_{ti} + \textbf{units}\ \alpha \frac{\textbf{units}\ f}{\textbf{units}\ x_i} = \textbf{units}\ \gamma \cdot \textbf{units}\ \mathbf{x}_i^t + \textbf{units}\ \alpha \frac{\textbf{units}\ f}{\textbf{units}\ x_i}$$

To add the units on the right side additively, we have:

$$\textbf{units}\ \gamma \cdot \textbf{units}\ \mathbf{x}_i^t = \textbf{units}\ \alpha \frac{\textbf{units}\ f}{\textbf{units}\ x_i}$$

As this all has to be equal to $\textbf{units}\ \mathbf{x}_i^t$, (because $\mathbf{v}_{n+1}$ needs the same units as $\mathbf{x}^t$) we arrive at:

$$\mathbf{units}\ \alpha = \frac{\mathbf{units}\ x_i^2}{\mathbf{units}\ f}$$

$$\mathbf{units}\ \gamma = 1$$

Therefore γ is dimensionless. Again, the units are not consistent, because

$\mathbf{units}\ \alpha$ is dependent on $\mathbf{units}\ \mathbf{x}_i^t$ (Which it shouldn't as α is a scalar and could theoretically not give different units for each component of $\mathbf{x}$).

AdaGrad

A very nice explanation about the notation and AdaGrad can be seen in the lecture PyBook or be found here:

https://medium.com/konvergen/an-introduction-to-adagrad-f130ae871827

For $\alpha \in \mathbb{R}^+$, we have the gradient update:

$$\mathbf{x}^{t+1} = \mathbf{x}^t - \frac{\alpha}{\sqrt{\epsilon\mathbf{I} + diag(\mathbf{G}_t)}}\nabla f_t$$

$$\Rightarrow \quad \mathbf{x}_i^{t+1} - \mathbf{x}_i^t = -\frac{\alpha}{\sqrt{\epsilon + \mathbf{G}_t^{(i,i)}}}\frac{\partial f_t}{\partial x_i}, \qquad \forall\, i: 1, .., n$$

From a units perspective, this leads to:

$$\begin{aligned}\mathbf{units}\left[\alpha\left(\epsilon + \mathbf{G}_t^{(i,i)}\right)^{-\frac{1}{2}}\frac{\partial f_t}{\partial x_i}\right] &= \mathbf{units}\ \alpha \cdot \mathbf{units}\left(\epsilon + \mathbf{G}_t^{(i,i)}\right)^{-\frac{1}{2}} \cdot \mathbf{units}\frac{\partial f_t}{\partial x_i} \\ &= \mathbf{units}\ \alpha \cdot \frac{\mathbf{units}\ x_i}{\mathbf{units}\ f} \cdot \frac{\mathbf{units}\ f}{\mathbf{units}\ x_i} \\ &= \mathbf{units}\ \alpha\end{aligned}$$

Therefore, $\mathbf{units}\ \alpha = \mathbf{units}\ x_i$, which again is not consistent across all dimensions.

AdaDelta

For description see here:

http://ruder.io/optimizing-gradient-descent/index.html#adadelta

From a units perspective, AdaDelta differs from AdaGrad only by the corrective factor $RMS[\Delta x]_{t-1}$ in the nominator. This corrective factor will have $\mathbf{units}\ RMS[\Delta x]_i = \mathbf{units}\ x_i$, therefore AdaDelta is consistent across units. Therefore, its units will not be consistent.

RMSProp

From a units perspective, the update rule is similar to AdaGrad without the corrective factor in the nominator. Therefore, the units are not consistent.

Nestograd

As it is only a slightly modified momentum methods, it has the same units. Therefore, it is not consistent.

Adam

Same as Adagrad, not consistent.

There are many more Gradient descent algorithms available. A good overview of the more common ones, can be found here:

http://ruder.io/optimizing-gradient-descent/

9A-2.

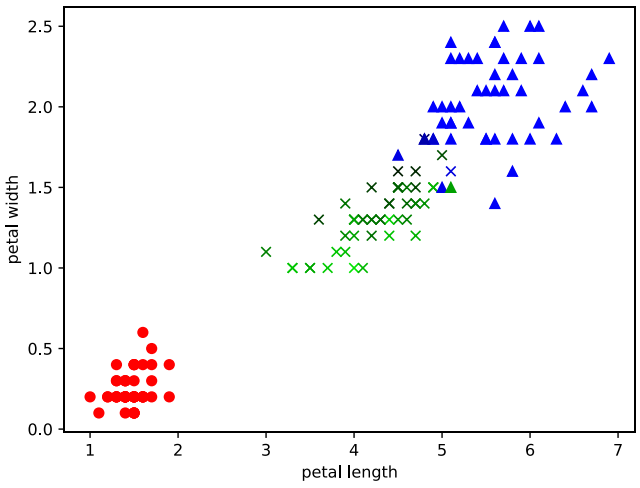


Figure 1 One vs many classification

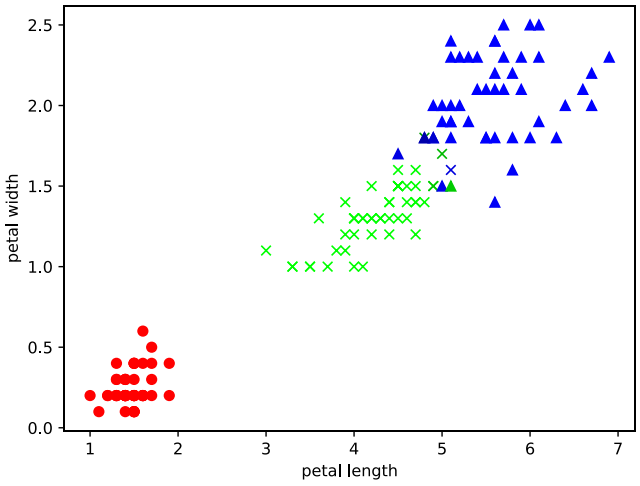


Figure 2 Classification through one single multinomial logistic regressor

The first image classified the images according to the "one-vs-rest" scheme. This means, we used binary regression for each class against its complement. To make an inference, we compared all the binary classifications and took the maximum estimated density. The second image shows multivariate classification.

As you can see, the classifications are identical although the probabilities differ slightly. Concerning the underlying model assumptions, however, both approaches are not equal.

For a detailed analysis of the different assumptions, see here:

https://stats.stackexchange.com/questions/52104/multinomial-logistic-regression-vs-one-vs-rest-binary-logistic-regression

9A-3.

For the calculation, please consult the "LectureLogistic" slide and look at page 13.

$$Error\ rate = \frac{FP + FN}{TP + TN + FP + FN} = \frac{50 + 10}{90 + 10 + 50 + 150} = \frac{60}{300} = 0.2$$

$$Precision = \frac{TP}{TP + FP} = \frac{90}{90 + 10} = 0.9$$

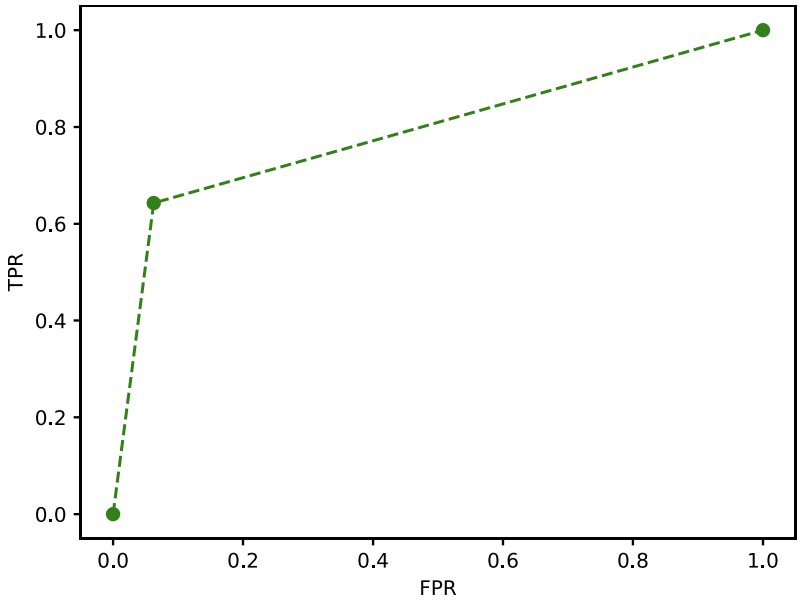
$$Recall = TPR = \frac{TP}{TP + FN} = \frac{90}{90 + 50} = \frac{9}{14}$$

$$Specifity = \frac{TN}{TN + FP} = \frac{150}{150 + 10} = \frac{15}{16}$$

To makes an inference about the ROC curve, let us calculate FPR:

$$FPR = \frac{FP}{FP + TN} = \frac{10}{10 + 150} = \frac{1}{16}$$

This gives us the ROC coordinate $(FPR, TPR) = (1/16, 9/14)$.
Naturally, we also have the points $\{(0,0), (1,1)\}$. This is the case, because the class probability $P(C_1|\mathbf{x})$ can never reach exactly one (see previous exercise A.7-4 for that) if our boundary is $\theta = 1$.
A rough sketch is therefore:



This classifier appears to be robust at separating the data, as $FPR \approx 0$ appears to be near the origin and $TPR \approx 0.64$.