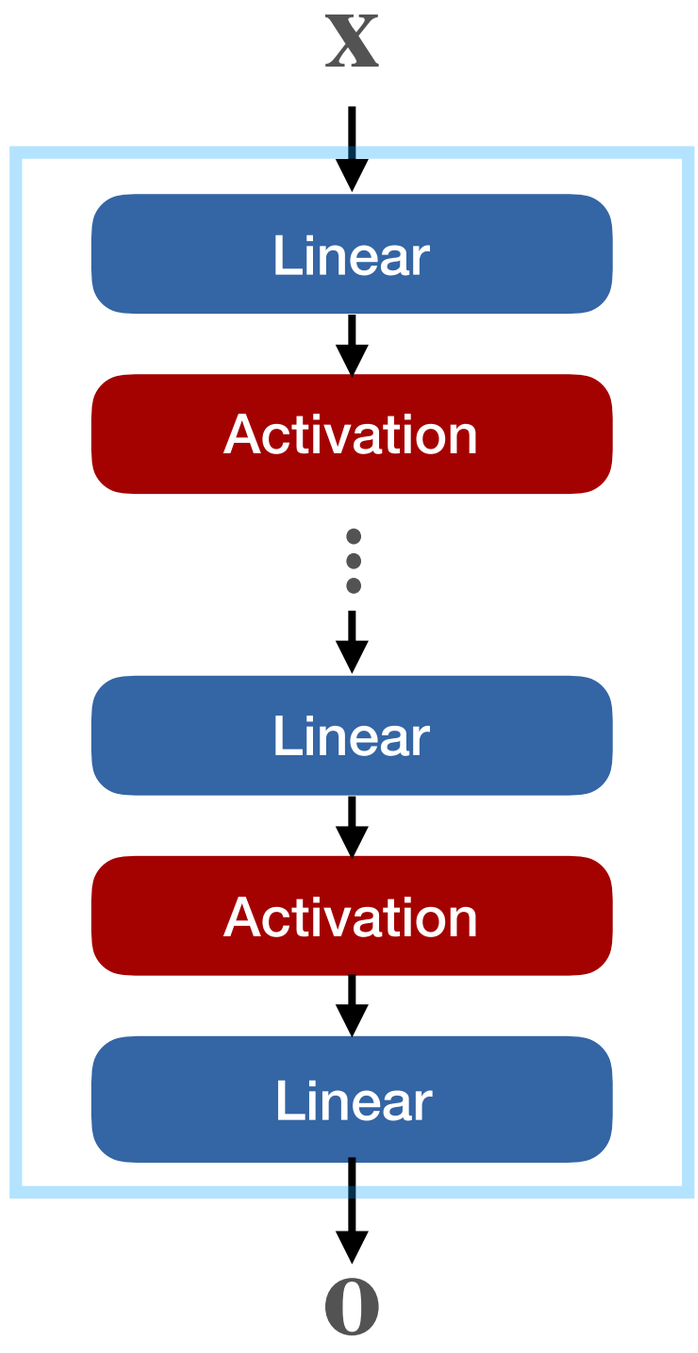
Loss functions

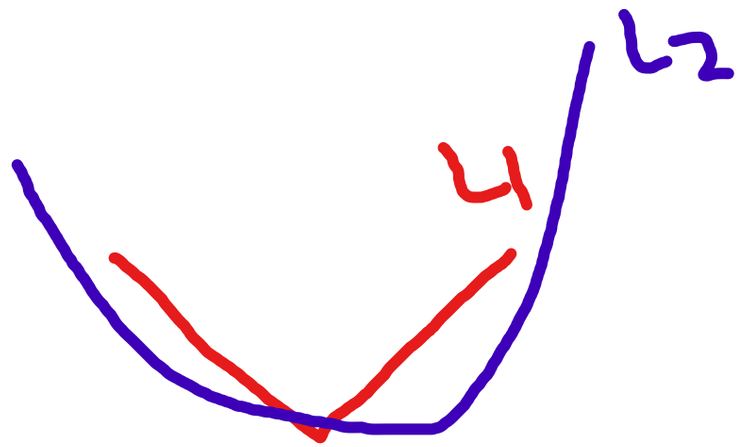
Inputs and outputs of networks

- Input:
  - Tensor $\mathbf{x}$
- Output:
  - Tensor $\mathbf{o}$



Regression

- L1 loss
  - $\ell = |\mathbf{y} - \mathbf{o}|$
- L2 loss
  - $\ell = \|\mathbf{y} - \mathbf{o}\|^2$



The L1 loss is a lot more forgiving to outliers.
So if you have a very, very high loss value,
the L1 loss increases linearly,
while the L2 loss increases quadratically,
which leads to huge gradients and huge loss values.

Classification

- Compute likelihood
  - $p(1) = \sigma(o)$
  - $\mathbf{p} = \text{softmax}(\mathbf{o})$

if it's a binary classification we're going to use a sigmoid
as an output transformation, if it's a multi-class classification problem,
we're going to use the softmax.

Then we're going to take the log of the probability that we want to predict,
we take the negative log of that probability, and
that's the loss function we're going to use for classification.

- Cross entropy / -Log likelihood
  - $-\log(p(y))$

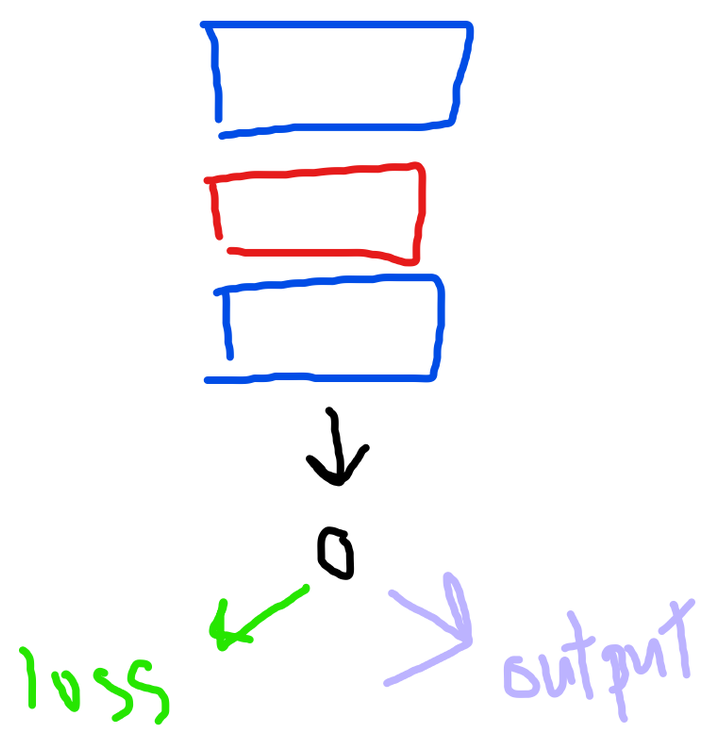
$-\sum_y q(x) \log p(y)$

$q = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$

Classification losses in practice

- $\sigma(o) = 0$ for $o \to -50$
  - $\log(\sigma(o)) = \log(0)$ is undefined
- Combine $\log$ and σ

  now in deep learning packages, the log and sigmoid are combined in one function
  - `BCEWithLogitsLoss`
  - `CrossEntropyLoss`



Never bake the output transformation in your network.
Output the raw values and then put the output transformation together with the loss into one function