Vectorization and Softmax

$$P(y|\mathbf{x}) = \frac{\exp(\mathbf{w}_y^\top f(\mathbf{x}))}{\sum_{y' \in \mathcal{Y}} \exp(\mathbf{w}_{y'}^\top f(\mathbf{x}))}$$

- Single scalar probability

- Three classes, "different weights"

$$\begin{array}{ccccc} \mathbf{w}_1^\top f(\mathbf{x}) & & -1.1 & & 0.036 \\ \mathbf{w}_2^\top f(\mathbf{x}) & = & 2.1 & \xrightarrow{\text{softmax}} & 0.89 \\ \mathbf{w}_3^\top f(\mathbf{x}) & & -0.4 & & 0.07 \end{array}$$

class probs

- Softmax operation = "exponentiate and normalize"

- We write this as: $\mathrm{softmax}(W f(\mathbf{x}))$

Logistic Regression with NNs

$$P(y|\mathbf{x}) = \frac{\exp(\mathbf{w}_y^\top f(\mathbf{x}))}{\sum_{y' \in \mathcal{Y}} \exp(\mathbf{w}_{y'}^\top f(\mathbf{x}))}$$

- Single scalar probability

$$P(\mathbf{y}|\mathbf{x}) = \mathrm{softmax}(W f(\mathbf{x}))$$

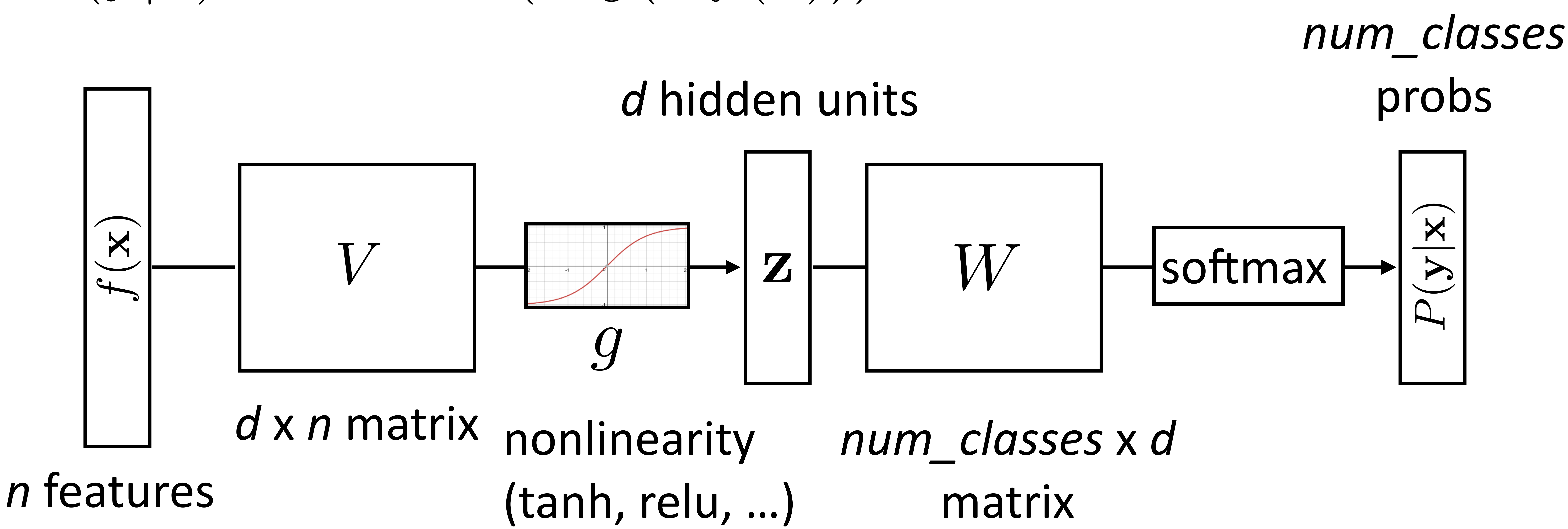
- Weight vector per class;
  W is [num classes x num feats]

$$P(\mathbf{y}|\mathbf{x}) = \mathrm{softmax}(W g(V f(\mathbf{x})))$$

- Now one hidden layer

Neural Networks for Classification

$$P(\mathbf{y}|\mathbf{x}) = \mathrm{softmax}(W g(V f(\mathbf{x})))$$



Training Neural Networks

$$P(\mathbf{y}|\mathbf{x}) = \text{softmax}(W\mathbf{z}) \qquad \mathbf{z} = g(Vf(\mathbf{x}))$$

- Maximize log likelihood of training data

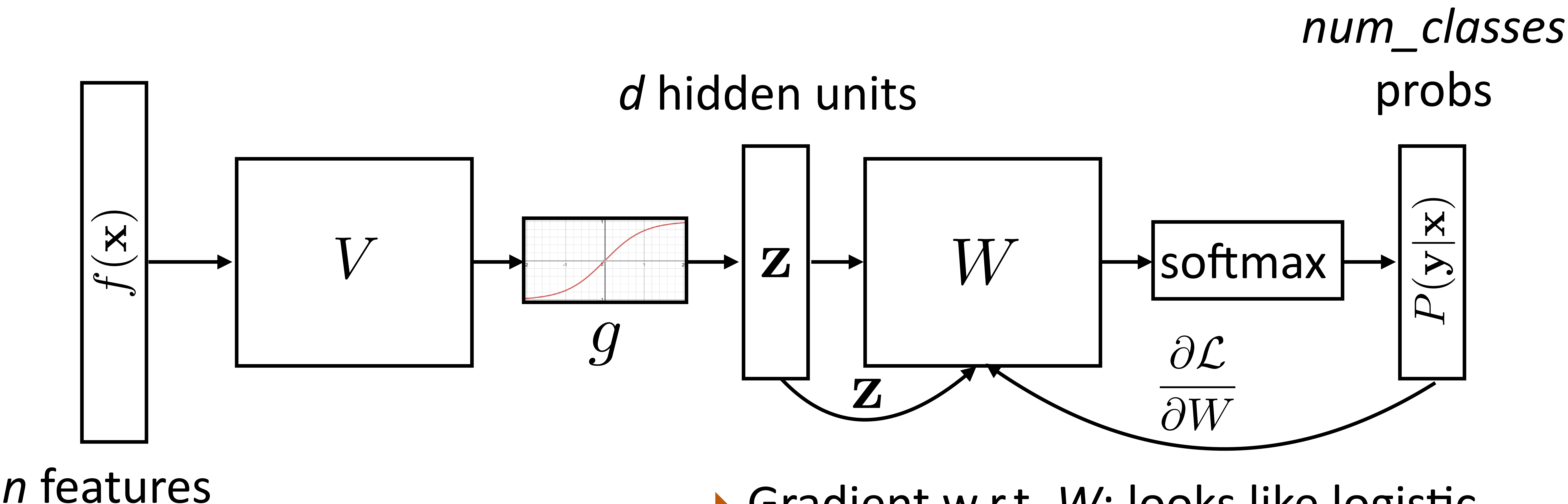
$$\mathcal{L}(\mathbf{x}, i^*) = \log P(y = i^*|\mathbf{x}) = \log\left(\text{softmax}(W\mathbf{z}) \cdot e_{i^*}\right)$$

- *i**: index of the gold label
- e_i: 1 in the *i*th row, zero elsewhere. Dot by this = select *i*th index

$$\mathcal{L}(\mathbf{x}, i^*) = W\mathbf{z} \cdot e_{i^*} - \log \sum_j \exp(W\mathbf{z}) \cdot e_j$$

Backpropagation Picture

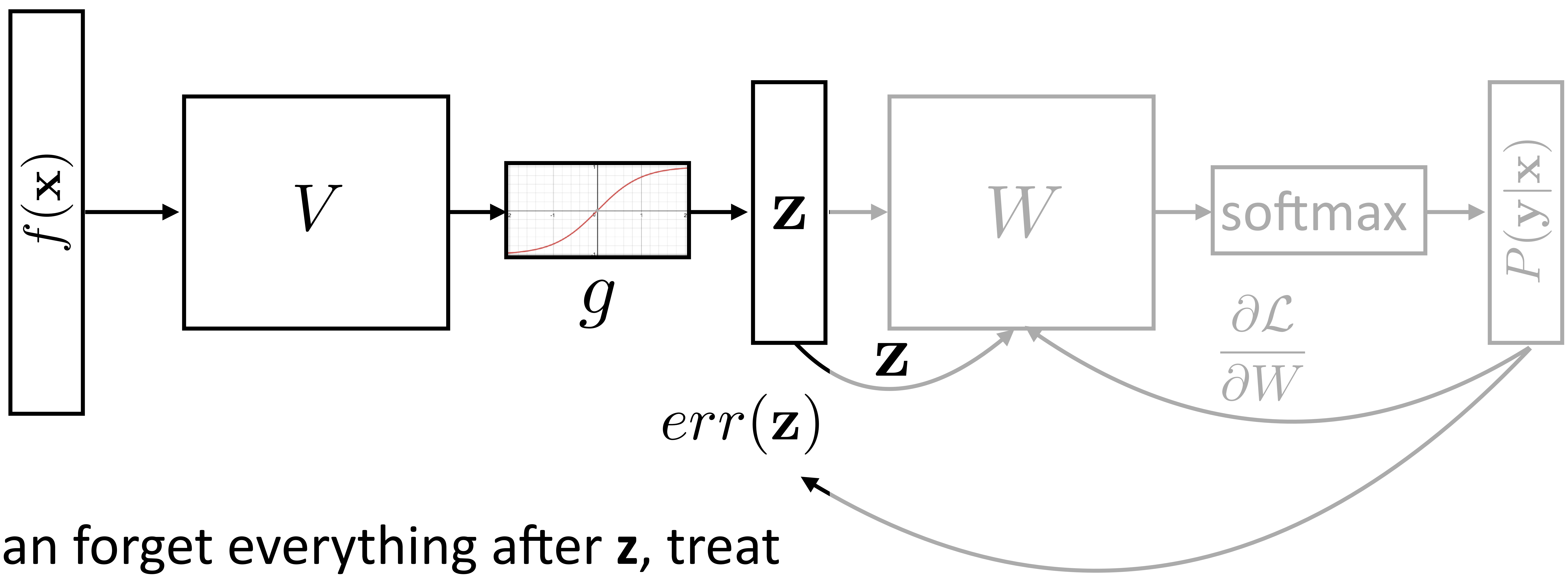
$$P(\mathbf{y}|\mathbf{x}) = \text{softmax}(Wg(Vf(\mathbf{x})))$$



- Gradient w.r.t. *W*: looks like logistic regression, can be computed treating **z** as the features

Backpropagation Picture

$$P(\mathbf{y}|\mathbf{x}) = \text{softmax}(Wg(Vf(\mathbf{x})))$$



- Can forget everything after **z**, treat it as the output and keep backpropping

Computing Gradients with Backprop

$$\mathcal{L}(\mathbf{x}, i^*) = W\mathbf{z} \cdot e_{i^*} - \log \sum_j \exp(W\mathbf{z}) \cdot e_j$$

$$\mathbf{z} = g(Vf(\mathbf{x}))$$

Activations at hidden layer

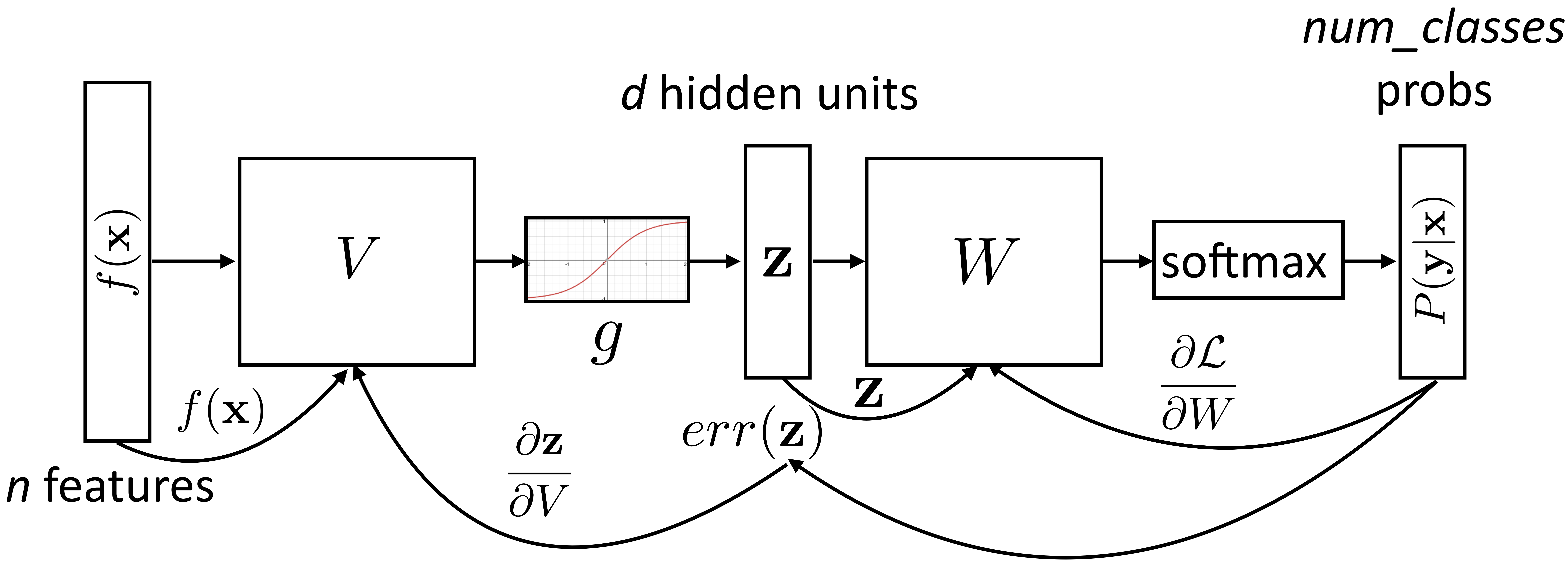
- Gradient with respect to *V*: apply the chain rule

$$\frac{\partial \mathcal{L}(\mathbf{x}, i^*)}{\partial V_{ij}} = \boxed{\frac{\partial \mathcal{L}(\mathbf{x}, i^*)}{\partial \mathbf{z}}} \boxed{\frac{\partial \mathbf{z}}{\partial V_{ij}}} \qquad \frac{\partial \mathbf{z}}{\partial V_{ij}} = \boxed{\frac{\partial g(\mathbf{a})}{\partial \mathbf{a}}} \boxed{\frac{\partial \mathbf{a}}{\partial V_{ij}}} \qquad \mathbf{a} = Vf(\mathbf{x})$$

- First term: *err*(**z**); represents gradient w.r.t. **z**
- First term: gradient of nonlinear activation function at ***a*** (depends on current value)
- Second term: gradient of linear function

Backpropagation Picture

$$P(\mathbf{y}|\mathbf{x}) = \mathrm{softmax}(W g(V f(\mathbf{x})))$$



- Combine backward gradients with forward-pass products