Deep Learning with MXNet Gluon
Supervised Learning
Supervised Learning

\[ \hat{p}(y \mid x) \]
Supervised Learning

\[ \mathbf{p}(y \mid x) \]
Loss Function

\[ \ell(p(y \mid x)) \]

\[ \hat{p}(y \mid x) \]

\[ p(y \mid x) \]
Cross Entropy Loss

\[ \ell = - \sum_{y \in \{0, 1, \ldots, 9\}} p(y | x) \log \hat{p}(y | x) \]
Cross Entropy Loss

\[ \ell = -\log \hat{p}(y^* \mid x) \quad y^* = 3 \]
Gradient Descent

\[ \ell = - \log \hat{p}_\theta(y^* \mid x) \]

\[ \theta \leftarrow \theta - \eta \nabla_\theta \ell \]

\[ \ell = - \frac{1}{N} \sum_{n=1}^{N} \log \hat{p}_\theta(y_n^* \mid x_n) \]

\[ \ell = - \frac{1}{B} \sum_{n=1}^{B} \log \hat{p}_\theta(y_b^* \mid x_b) \]
Logistic Regression

\[ x \in \mathbb{R}^D \quad \beta_y = \theta_y^\top x \]

\[ \hat{p}_\theta(y \mid x) = \frac{\exp(\beta_y)}{\sum_{y \in \{0,1,\ldots,9\}} \exp(\beta_y)} \]

Softmax Function
Multi-Layer Perceptron

\[ \beta_y = \theta_y^T x \]

One Linear Layer

\[ \beta_y \leftarrow f(\beta_y) \]

One Non-Linear Layer
Multi-Layer Perceptron

\[ \beta_y = \theta_y^T x \]

One Linear Layer

\[ \beta_y \leftarrow f(\beta_y) \]

One Non-Linear Layer
MXNet Gluon

Let’s make this cool stuff now!

Documents: Dope

Examples: IPython Notebooks