Neural networks
Training neural networks - empirical risk minimization
Neural Network

**Topics:** multilayer neural network

- Could have $L$ hidden layers:
  - layer input activation for $k > 0$ $(h^{(0)}(x) = x)$
    \[
    a^{(k)}(x) = b^{(k)} + W^{(k)} h^{(k-1)}(x)
    \]
  - hidden layer activation ($k$ from 1 to $L$):
    \[
    h^{(k)}(x) = g(a^{(k)}(x))
    \]
  - output layer activation ($k = L + 1$):
    \[
    h^{(L+1)}(x) = o(a^{(L+1)}(x)) = f(x)
    \]
MACHINE LEARNING

**Topics:** empirical risk minimization, regularization

- Empirical risk minimization
  - framework to design learning algorithms
  
  \[
  \arg\min_{\theta} \frac{1}{T} \sum_{t} l(f(x^{(t)}; \theta), y^{(t)}) + \lambda \Omega(\theta)
  \]
  
  - \( l(f(x^{(t)}; \theta), y^{(t)}) \) is a loss function
  - \( \Omega(\theta) \) is a regularizer (penalizes certain values of \( \theta \))

- Learning is cast as optimization
  
  - ideally, we'd optimize classification error; but it's not smooth
  - loss function is a surrogate for what we truly should optimize (e.g. upper bound)
MACHINE LEARNING

Topics: stochastic gradient descent (SGD)

• Algorithm that performs updates after each example
  ‣ initialize $\theta$ ( $\theta \equiv \{W^{(1)}, b^{(1)}, \ldots, W^{(L+1)}, b^{(L+1)}\}$)
  ‣ for $N$ iterations
    - for each training example $(x^{(t)}, y^{(t)})$
    \[ \Delta = -\nabla_{\theta} l(f(x^{(t)}; \theta), y^{(t)}) - \lambda \nabla_{\theta} \Omega(\theta) \]
    \[ \theta \leftarrow \theta + \alpha \Delta \]

• To apply this algorithm to neural network training, we need
  ‣ the loss function $l(f(x^{(t)}; \theta), y^{(t)})$
  ‣ a procedure to compute the parameter gradients $\nabla_{\theta} l(f(x^{(t)}; \theta), y^{(t)})$
  ‣ the regularizer $\Omega(\theta)$ (and the gradient $\nabla_{\theta} \Omega(\theta)$)
  ‣ initialization method