Machine Learning 1

Lecture 07 - Logistic Regression - Neural Nets

Patrick Forré

1. Logistic Regression

2. Supervised Learning: Neural Nets

Linear Classification: Logistic Regression for two classes (I)

- Given: Data set $D = (x_1, \ldots, x_N)^T$ with binary classes $T = (t_1, \ldots, t_N)^T$ with $t_i \in \{c_0, c_1\} = \{0, 1\}$.
- Basis functions: $\phi = \phi(x) = (\phi_0(x), \ldots, \phi_M(x))^T$ with $\phi_0 \equiv 1$.
- Model assumption of Logistic Regression: The posterior probability $p(c_1|\phi)$ is the sigmoid of a linear function in the feature vector ϕ:

$$p(c_1|\phi, w) = \sigma(w^T\phi)$$

  with weight vector $w = (w_0, \ldots, w_M) \in \mathbb{R}^{M+1}$.
- Conditional distribution:

$$\begin{aligned} p(t|\phi, w) &= \begin{cases} \sigma(w^T\phi) & \text{if} \quad t = 1, \\ 1 - \sigma(w^T\phi) & \text{if} \quad t = 0, \end{cases} \\ &= \sigma(w^T\phi)^t \cdot (1 - \sigma(w^T\phi))^{1-t}. \end{aligned}$$

Linear Classification: Logistic Regression for two classes (II)

- Conditional Likelihood (under i.i.d. assumptions):

$$\begin{aligned} p(T|\Phi, w) &= \prod_{n=1}^{N} p(t_n|\phi(x_n), w) \\ &= \prod_{n=1}^{N} \sigma(w^T\phi(x_n))^{t_n} \cdot (1-\sigma(w^T\phi(x_n)))^{1-t_n} \\ &= \prod_{n=1}^{N} y_n^{t_n} \cdot (1-y_n)^{1-t_n} \end{aligned}$$

  with $y_n := \sigma(w^T\phi(x_n))$. Put $Y = (y_1, \ldots, y_N)^T$.
- For the maximum likelihood approach we either needed to know $p(\Phi|w)$ or at least assume that it does not depend on w.
- This leads to maximizing the conditional likelihood w.r.t. w.
- This is equivalent to minimizing the cross-entropy error:

$$\begin{aligned} E(w) &= -\ln p(T|\Phi, w) \\ &= -\sum_{n=1}^{N} [t_n \ln y_n + (1-t_n)\ln(1-y_n)] \end{aligned}$$

- $E(w)$ is convex, but no closed form solution exists (due to the non-linearity of σ).
- For minimizing $E(w)$ use e.g. stochastic gradient descent or iteratively reweighted least squares.

Preliminary: Chain Rule

Theorem (Chain Rule of Calculus)

For two differentiable and composable functions f, g we have:

$$(f \circ g)'(x) = f'(g(x)) \cdot g'(x).$$

For three functions h, f, g we have:

$$(h \circ f \circ g)'(x) = h'(f(g(x)) \cdot f'(g(x)) \cdot g'(x).$$

Example: Logistic sigmoid function $\sigma(x) = (1 + \exp(-x))^{-1}$.

$$\begin{aligned}
\sigma'(x) &= (-1)(1 + \exp(-x))^{-2} \cdot \exp(-x) \cdot (-1) \\
&= \tfrac{1}{1+\exp(-x)} \tfrac{\exp(-x)}{1+\exp(-x)} \\
&= \tfrac{1}{1+\exp(-x)} \tfrac{1}{\exp(x)+1} \\
&= \sigma(x)\sigma(-x) \\
&= \sigma(x)(1 - \sigma(x)).
\end{aligned}$$

Stochastic Gradient Descent for Cross-Entropy (I)

- Given: Data set $D = (x_1, \dots, x_N)^T$ with binary classes $T = (t_1, \dots, t_N)^T$ with $t_i \in \{c_0, c_1\} = \{0, 1\}$.
- Put $\Phi = (\phi(x_1), \dots, \phi(x_N))^T$ with basis functions $\phi(x) = (\phi_0(x), \dots, \phi_M(x))^T$ and $y_n = \sigma(w^T\phi(x_n))$.
- Cross entropy error for one observation: $E_n(w) = -(t_n \ln y_n + (1 - t_n)\ln(1 - y_n))$.

$$
\begin{aligned}
\frac{\partial \ln \sigma(w^T\phi(x_n))}{\partial w_j} &= \frac{1}{\sigma(w^T\phi(x_n))} \cdot \sigma'(w^T\phi(x_n)) \cdot \frac{w^T\phi(x_n)}{\partial w_j} \\
&= \frac{\sigma(w^T\phi(x_n))\sigma(-w^T\phi(x_n))}{\sigma(w^T\phi(x_n))} \cdot \frac{\sum_m w_m\phi_m(x_n)}{\partial w_j} \\
&= \sigma(-w^T\phi(x_n)) \cdot \phi_j(x_n). \\
\frac{\partial^2 \ln \sigma(w^T\phi(x_n))}{\partial^2 w_j} &= -\sigma(w^T\phi(x_n)) \cdot \sigma(-w^T\phi(x_n)) \cdot \phi_j(x_n)^2 < 0.
\end{aligned}
$$

$$
\begin{aligned}
\frac{\partial E_n(w)}{\partial w_j} &= -t_n \frac{\partial \ln y_n}{\partial w_j} - (1 - t_n)\frac{\partial \ln(1-y_n)}{\partial w_j}) \\
&= -t_n \frac{\partial \ln \sigma(w^T\phi(x_n))}{\partial w_j} - (1 - t_n)\frac{\partial \ln(1-\sigma(w^T\phi(x_n)))}{\partial w_j}) \\
&= -t_n\sigma(-w^T\phi(x_n)) \cdot \phi_j(x_n) + (1 - t_n)\sigma(w^T\phi(x_n)) \cdot \phi_j(x \\
&= (-t_n(1 - y_n) + (1 - t_n)y_n) \cdot \phi_j(x_n) \\
&= (y_n - t_n) \cdot \phi_j(x_n).
\end{aligned}
$$

Stochastic Gradient Descent for Cross-Entropy (II)

- Cross-entropy error for one observation:
  $E_n(w) = -\left(t_n \ln y_n + (1-t_n)\ln(1-y_n)\right)$ with
  $y_n = \sigma(w^T\phi(x_n))$.
- $\frac{\partial E_n(w)}{\partial w_j} = (y_n - t_n)\cdot\phi_j(x_n)$.
- Gradient: $\nabla E_n(w) = \left(\frac{\partial E_n(w)}{\partial w_0}, \ldots, \frac{\partial E_n(w)}{\partial w_M}\right)^T = (y_n - t_n)\cdot\phi(x_n)$.
- Cross-entropy: $E(w) = \sum_{n=1}^N E_n(w) =$
  $-\sum_{n=1}^N \left[t_n \ln y_n + (1-t_n)\ln(1-y_n)\right]$.
- Stochastic Gradient Descent:
  - Carefully choose an initial value: $w^{(0)}$ (often $w^{(0)} = 0$ works).
  - Carefully choose a learning rate $\eta > 0$.
  - Randomly choose an observation x_n and use update rule:

$$\begin{aligned} w^{(\tau+1)} &:= w^{(\tau)} - \eta\nabla E(w^{(\tau)}) \\ &= w^{(\tau)} - \eta\cdot(y_n^{(\tau)} - t_n)\cdot\phi(x_n) \end{aligned}$$

  - Iterating the last point will make the algorithm converges (if η is not too big): $E(w)$ is convex (Hessian positive definite).
  - If η is too small the algorithm is too slow.
- We end up with an approximate minimizer w^* of $E(w)$.

Compare: Stochastic Gradient Descent for Linear Regression

- Sum-of-Squares error: $E(w) = \sum_{n=1}^{N} E_n(w)$ with:
- Sum-of-Squares error for one observation:
  $E_n(w) = \frac{1}{2}(y_n - t_n)^2$ with $y_n = w^T \phi(x_n)$.
- $\frac{\partial E_n(w)}{\partial w_j} = (y_n - t_n) \cdot \phi_j(x_n)$.
- Gradient: $\nabla E_n(w) = (\frac{\partial E_n(w)}{\partial w_0}, \ldots, \frac{\partial E_n(w)}{\partial w_M})^T = (y_n - t_n) \cdot \phi(x_n)$.
- Stochastic Gradient Descent with same update rule as in Logistic Regression:

$$\begin{aligned} w^{(\tau+1)} &:= w^{(\tau)} - \eta \nabla E(w^{(\tau)}) \\ &= w^{(\tau)} - \eta \cdot (y_n^{(\tau)} - t_n) \cdot \phi(x_n) \end{aligned}$$

Iteratively Reweighted Least Squares for Cross-Entropy

- Given: Data set $D = (x_1, \dots, x_N)^T$ with binary classes $T = (t_1, \dots, t_N)^T$ with $t_i \in \{c_0, c_1\} = \{0, 1\}$.
- Put $\Phi = (\phi(x_1), \dots, \phi(x_N))^T$ with basis functions $\phi(x) = (\phi_0(x), \dots, \phi_M(x))^T$.
- Put $y_n = \sigma(w^T \phi(x_n))$ and $Y = (y_1, \dots, y_N)^T$ with $w \in \mathbb{R}^{M+1}$.
- $E(w) = -\sum_{n=1}^{N} \left[t_n \ln y_n + (1 - t_n) \ln(1 - y_n)\right]$.
- Gradient: $\nabla E(w) = \sum_{n=1} (y_n - t_n)\phi(x_n) = \Phi^T (Y - T)$.
- Hessian: $H(w) = \sum_{n=1}^{N} y_n(1 - y_n)\phi(x_n)\phi(x_n)^T = \Phi^T R \Phi$, with
- diagonal matrix R with entries $R_{nn} = y_n(1 - y_n)$.
- Newton-Raphson update:

$$
\begin{aligned}
w^{(t+1)} &:= w^{(t)} - H(w^{(t)})^{-1} \nabla E(w^{(t)}) \\
&= w^{(t)} - (\Phi^T R^{(t)} \Phi)^{-1} \Phi^T (Y^{(t)} - T) \\
&= (\Phi^T R^{(t)} \Phi)^{-1} \left[\Phi^T R^{(t)} \Phi w^{(t)} - \Phi^T (Y^{(t)} - T)\right] \\
&= (\Phi^T R^{(t)} \Phi)^{-1} \Phi^T R^{(t)} Z^{(t)}, \\
Z^{(t)} &= \Phi w^{(t)} - (R^{(t)})^{-1} (Y^{(t)} - T)
\end{aligned}
$$

- We end up with an approximate minimizer w^* of $E(w)$.

Geometry of Gradient Descent and Newton Optimization

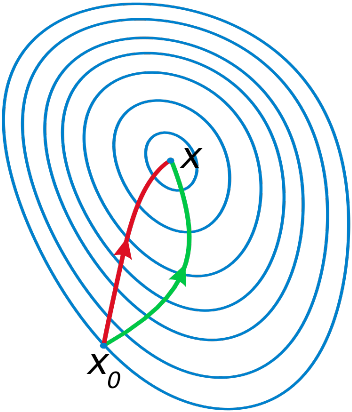

Figure: Contours of a convex error function $E(w)$. $w = X$ is the minimum of $E(w)$ and $w = X_0$ a starting point. Green: Gradient descent follows the steepest descent at each point, othogonal to the contours. Red: Newton-Raphson method also takes the curvature into account to shorten the way. (Source: Wikipedia - Newton's method in optimization)

Classification with Logistic Regression

- Given: Data set $D = (x_1, \ldots, x_N)^T$ with binary classes $T = (t_1, \ldots, t_N)^T$ with $t_i \in \{c_0, c_1\} = \{0, 1\}$.
- Basis functions: $\phi = \phi(x) = (\phi_0(x), \ldots, \phi_M(x))^T$.
- Model assumption of Logistic Regression: $p(c_1|\phi, w) = \sigma(w^T\phi)$.
- Minimizing the cross-entropy error:

$$\begin{aligned} E(w) &= -\ln p(T|\Phi, w) \\ &= -\textstyle\sum_{n=1}^{N} \left[t_n \ln y_n + (1 - t_n) \ln(1 - y_n) \right]. \end{aligned}$$

- Using stochastic gradient descent or iterative reweighted least squares we end up with a approximate minimizer w^* of $E(w)$.
- We assign a new data point x to class c_1 if $\sigma((w^*)^T\phi(x)) > \frac{1}{2}$, i.e. if $(w^*)^T\phi(x) > 0$.
- Decision regions: $\mathcal{R}_1 = \{x|(w^*)^T\phi(x) > 0\}$ and $\mathcal{R}_0 = \{x|(w^*)^T\phi(x) < 0\}$.
- Decision boundaries: $\mathcal{B} = \{x|(w^*)^T\phi(x) = 0\}$.

Logistic Regression for multiple classes

- Data $D = (x_1, \dots, x_N)^T$ with $T = (t_1, \dots, t_N)^T$ of K-dim one-vs-the-rest vectors $t_i = (0, \dots, 1, \dots, 0)^T$.
- Model assumption of Logistic Regression:

$$p(c_k|\phi, w_1, \dots, w_k) = \sigma_k(w_1^T\phi, \dots, w_K^T\phi),$$

  with weight vectors $w_k = (w_{k,0}, \dots, w_{k,M}) \in \mathbb{R}^{M+1}$.
- Put $y_{nk} := \sigma_k(w_1^T\phi(x_n), \dots, w_K^T\phi(x_n))$.
- Minimize the cross-entropy error w.r.t. w:

$$E(W) = -\ln p(T|\Phi, W) = -\sum_{n=1}^{N}\sum_{k=1}^{K} t_{nk} \ln y_{nk}.$$

- Gradient: $\nabla_{w_j} E(W) = \sum_{n=1}^{N}(y_{nj} - t_{nj})\phi(x_n)$
- Hessian: $\nabla_{w_k}\nabla_{w_j} E(W) = -\sum_{n=1}^{N} y_{nk}(\mathbb{1}_{nj} - y_{nj})\phi(x_n)\phi(x_n)^T$.
- We assign x to class c_k if $\sigma_k > \sigma_j$ for all $j \neq k$, i.e.:
- Decision regions: $\mathcal{R}_k = \{x|(w_k^*)^T\phi(x) > (w_j^*)^T\phi(x), \forall j \neq k\}$.
- Decision boundaries: $\mathcal{B}_{jk} = \{x|(w_j^*)^T\phi(x) = (w_k^*)^T\phi(x)\}$.

Bayesian Model Comparison and BIC

- Given: model $\mathcal{M}_i$ with parameters $\mathcal{W}_i$ of dim M_i and data D.
- Bayesian Model Comparison: Highest model evidence: $p(D|\mathcal{M}_i) = \int_{\mathcal{W}_i} p(D|w,\mathcal{M}_i)p(w|\mathcal{M}_i)dw$.
- (In model) Posterior: $p(w|D,\mathcal{M}_i) = \frac{p(D|w,\mathcal{M}_i)p(w|\mathcal{M}_i)}{p(D|\mathcal{M}_i)}$.
- Laplace approximation (see next slide) of maximum a posteriori for model $\mathcal{M}_i$:

$$\begin{aligned} p(w|D,\mathcal{M}_i) &\approx \mathcal{N}(w|w_{\mathrm{MAP}}, A^{-1}), \\ A &= -\nabla\nabla \ln p(w_{\mathrm{MAP}}|D,\mathcal{M}_i) \\ &= -\nabla\nabla \ln p(D|w_{\mathrm{MAP}},\mathcal{M}_i)p(w_{\mathrm{MAP}}|\mathcal{M}_i). \end{aligned}$$

- This leads to the approximation: $\ln p(D|\mathcal{M}_i) \approx \ln p(D|w_{\mathrm{MAP}},\mathcal{M}_i) + \ln p(w|\mathcal{M}_i) + \frac{M_i}{2}\ln(2\pi) - \frac{1}{2}\ln|A|$.
- Broad Gaussian prior and A of full rank gives us BIC:

$$\ln p(D|\mathcal{M}_i) \approx \ln p(D|w_{\mathrm{MAP}},\mathcal{M}_i) - \frac{M_i}{2}\ln N + \mathrm{const.}$$

- This is a very rough approximation and the assumptions often fail in practice.

Laplace Approximation

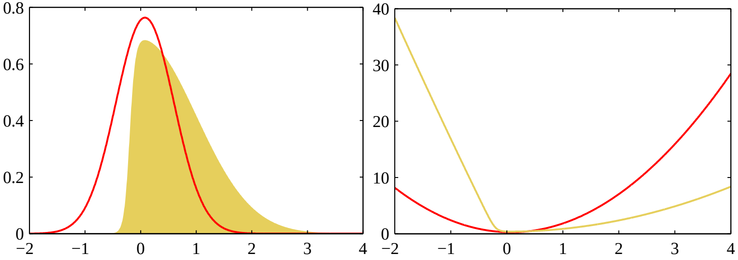

Figure: Left: Distribution $p(z)$ approximated by Gaussian distribution $\mathcal{N}(z|z_0, A^{-1})$ around mode $z = z_0$ with Hessian maxtrix $A = -\nabla\nabla \ln p(z)|_{z=z_0}$. Right: $-\ln p(x)$ (yellow) and $-\ln \mathcal{N}(z|z_0, A^{-1})$ (red). (Bishop 4.14)

Bayesian Logistic Regression

- Likelihood: $p(T|\Phi, w) = \prod_{n=1}^{N} y_n^{t_n} \cdot (1 - y_n)^{1-t_n}$ with $y_n := \sigma(w^T \phi(x_n))$.
- Gaussian prior: $p(w) = \mathcal{N}(w|\mu_0, \Sigma_0)$.
- Posterior then is:

$$\begin{aligned} \ln p(w|\Phi, T) &= -\tfrac{1}{2}(x - \mu_0)^T \Sigma_0^{-1}(x - \mu_0) \\ &\quad + \textstyle\sum_n [t_n \ln y_n + (1 - t_n) \ln(1 - y_n)] + \text{const.} \end{aligned}$$

- Laplace approximation of the posterior:

$$\begin{aligned} p(w|\Phi, T) &\approx \mathcal{N}(w|w_{\mathrm{MAP}}, \Sigma_N) \\ \Sigma_N^{-1} &= -\nabla\nabla \ln p(w|T, \Phi) \\ &= \Sigma_0^{-1} + \textstyle\sum_{n=1}^{N} y_n(1 - y_n)\phi(x_n)\phi(x_n)^T. \end{aligned}$$

- Predictive distribution: $p(c_1|\phi, \Phi, T) \approx$

$$\int \sigma(w^T \phi)\mathcal{N}(w|w_{\mathrm{MAP}}, \Sigma_N)dw \approx \sigma\left((1 + \frac{\pi}{8}\Phi^T \Sigma_N \Phi)^{-1/2} w_{\mathrm{MAP}}^T \phi\right).$$

1. Logistic Regression

2. Supervised Learning: Neural Nets

Neural Nets: The Idea

- First consider Linear Regression and Logistic Regression.
- Training data: D with targets $T = (t_1, \ldots, t_N)^T$.
- Targets (simplest case): LinReg: $t_i \in \mathbb{R}$. LogReg: $t_i \in \{0, 1\}$.
- Features: $\phi(x) = (\phi_0(x), \ldots, \phi_M(x))^T$ with $\phi_0 \equiv 1$.
- Model functions in LinReg: $y(x, w) = w^T \phi(x)$.
- Model functions in LogReg: $y(x, w) = \sigma(w^T \phi(x))$.
- Problem: Feature functions need to be known or handcrafted!
- Idea: Create "flexible" non-linear features and learn them:

$$\phi_m(x) = \sigma({w'_m}^T x) = \sigma\left(\sum_{d=0}^{D} w'_{m,d} \cdot x_d\right).$$

  with weights $w'_m \in \mathbb{R}^{D+1}$, $x = (1, x_1, \ldots, x_D)$ (here: components, not observations), $x_0 \equiv 1$.
- Regression: $y(x, w, w') = \sum_{m=0}^{M} w_m \cdot \sigma\left(\sum_{d=0}^{D} w'_{m,d} \cdot x_d\right)$.
- Classification:
  $y(x, w, w') = \sigma\left(\sum_{m=0}^{M} w_m \cdot \sigma\left(\sum_{d=0}^{D} w'_{m,d} \cdot x_d\right)\right)$.
- Caution: The inner functions are not linear anymore!!!

Question (Why should this work?)

Why should the approach to model functions like:

$$y(x, w, w') = \sum_{m=0}^{M} w_m \cdot \sigma\left(\sum_{d=0}^{D} w'_{m,d} \cdot x_d\right),$$

work? Why the logistic sigmoid function σ?

Theorem (Universal Approximator)

Let f be any continuous function on a compact area of $\mathbb{R}^D$ and σ any fixed analytic function which is not a polynomial (e.g. σ logistic function). Given any small number $\epsilon > 0$ of an acceptable error we can find a number M and weights $w_m, w'_{m,d} \in \mathbb{R}$ such that:

$$\left|f(x) - y(x, w, w')\right| < \epsilon.$$

<u>Caution:</u> With smaller ϵ we usually have to take bigger M!

Example: Approximation Power of Neural Nets

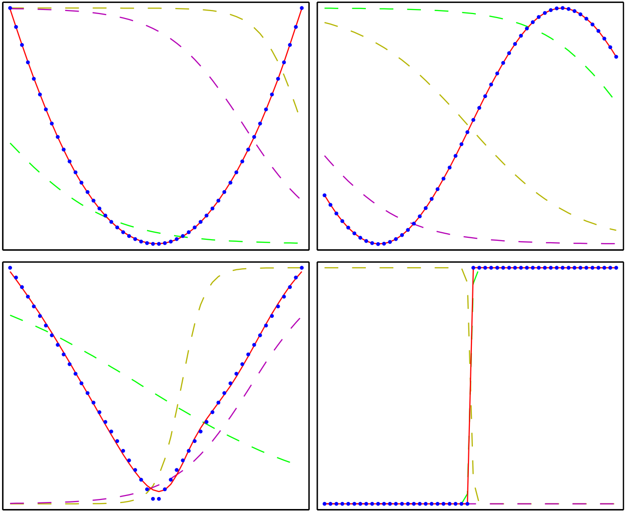

Figure: Approximation power of neural net with $M = 3$ features. $N = 50$ data points. a) $f(x) = x^2$, b) $f(x) = \sin(x)$, c) $f(x) = |x|$, d) $f(x) =$ step function. Dashed lines are the curves of the features. (Bishop 5.3)

Example: Classification Power of Neural Nets

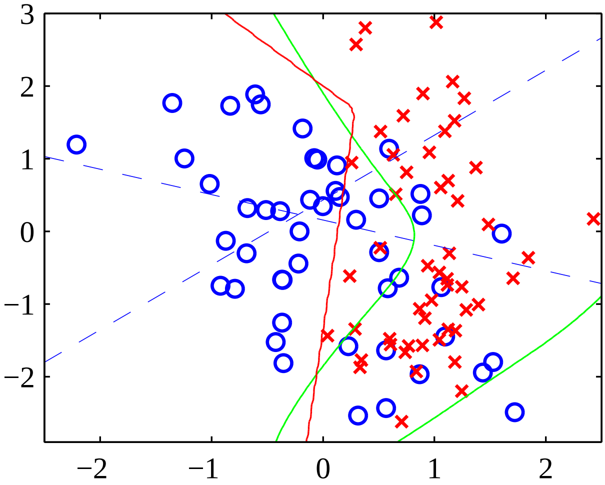

Figure: Classification power of neural net with $M = 2$ features. Decision boundary in red. Optimal boundary in green. Dashed blue lines are $z = 0.5$ contours of the features. (Bishop 5.4)

Neural Nets: Multiple Outputs and Multiple Classes

- Training data: $D = (x_1, \ldots, x_N)^T$ with $T = (t_1, \ldots, t_N)^T$.
- Regression: $t_i \in \mathbb{R}^K$ (K-components).
- Classification: K-dim one-vs-the-rest: $t_i = (0, \ldots, 1, \ldots, 0)^T$.
- Model functions of Neural Nets for Regression:

$$y_k(x, w^{(1)}, w^{(2)}) = \sum_{m=0}^{M} w_{k,m}^{(2)} \cdot \sigma\left(\sum_{d=0}^{D} w_{m,d}^{(1)} \cdot x_d\right),$$

  where σ is the logistic sigmoid function.
  After learning $w^{(1)}, w^{(2)}$ the target $t = (t_{,1}, \ldots, t_{,K})^T$ of new data x is predicted by $(y_1(x), \ldots, y_K(x))^T$.
- Model functions of Neural Nets for Classification:

$$y_k(x, w^{(1)}, w^{(2)}) = \sigma_k\left(\left(\sum_{m=0}^{M} w_{k',m}^{(2)} \cdot \sigma\left(\sum_{d=0}^{D} w_{m,d}^{(1)} \cdot x_d\right)\right)_{k'=1,\ldots,K}\right)$$

  where σ_k is k-th comp. of softmax or the sigmoid of k-comp..
  After learning $w^{(1)}, w^{(2)}$ a new data point x is assigned to class c_k if $y_k(x) > y_j(x)$ for all $j \neq k$.

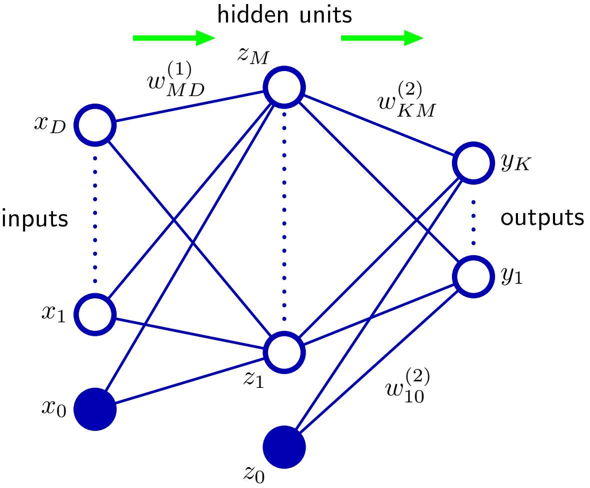

Figure: Edges correspond to weights $w^{(\ell)}_{i,j}$ and nodes to input, output and hidden units $z_m = \sigma(\sum_d w^{(1)}_{m,d} x_d)$. Big blue nodes correspond to constant functions $x_0 = 1$, $z_0 = 1$ and their edges to bias terms. (Bishop 5.1)