CS540 Machine learning
Lecture 6

Last time

- Linear and ridge regression (QR, SVD, LMS)

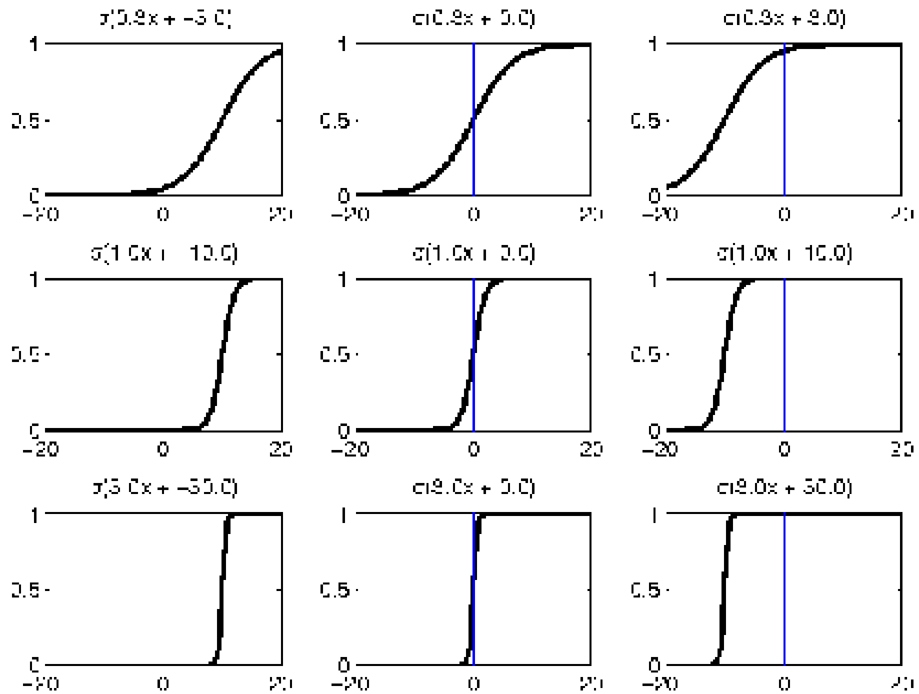
This time

- Logistic regression
- MLE
- Perceptron algorithm
- IRLS
- Multinomial logistic regression

Logistic regression

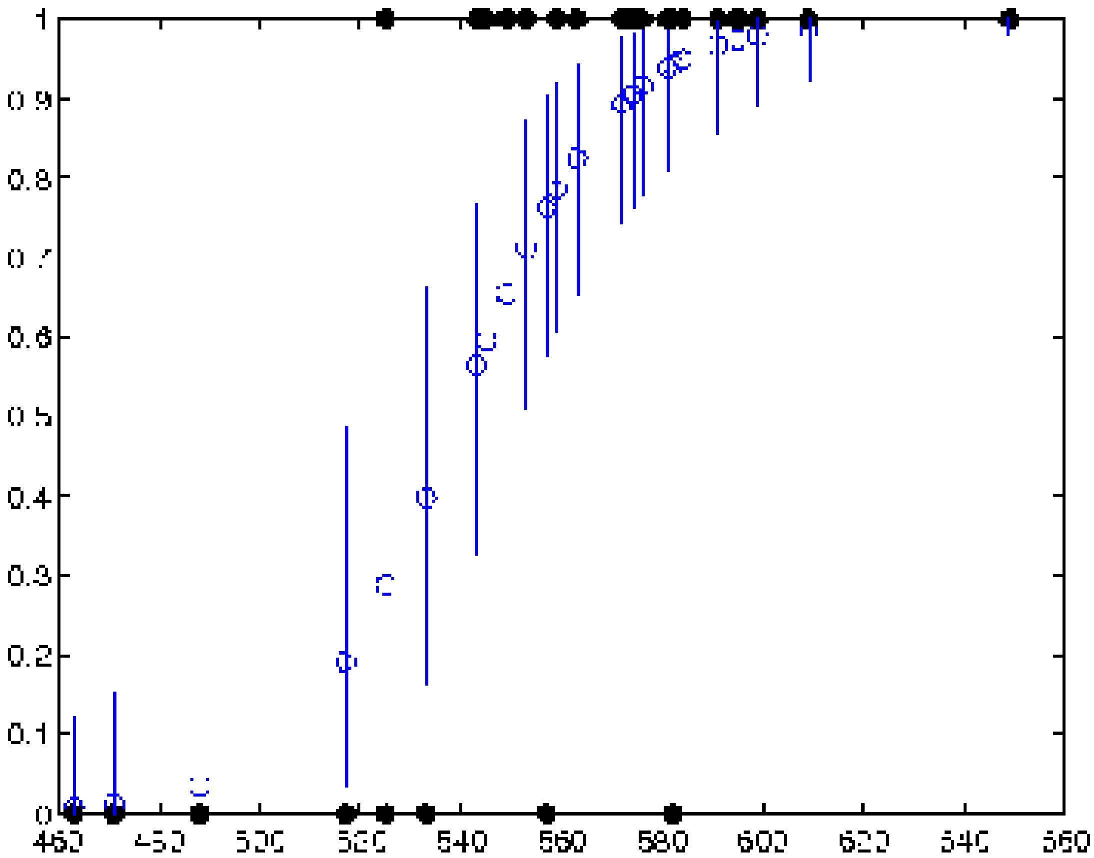
- Model for binary *classification*

$$
\begin{aligned}
p(y|\mathbf{x}, \mathbf{w}) &= \mathrm{Ber}(y|\sigma(\eta)) = \sigma(\eta)^y (1-\sigma(\eta))^{1-y} \\
\eta &= \mathbf{w}^T \boldsymbol{\phi}(\mathbf{x}) \\
\sigma(\eta) &\overset{\text{def}}{=} \frac{1}{1+\exp(-\eta)} = \frac{e^\eta}{e^\eta + 1}
\end{aligned}
$$



Sigmoid
or
Logistic
function

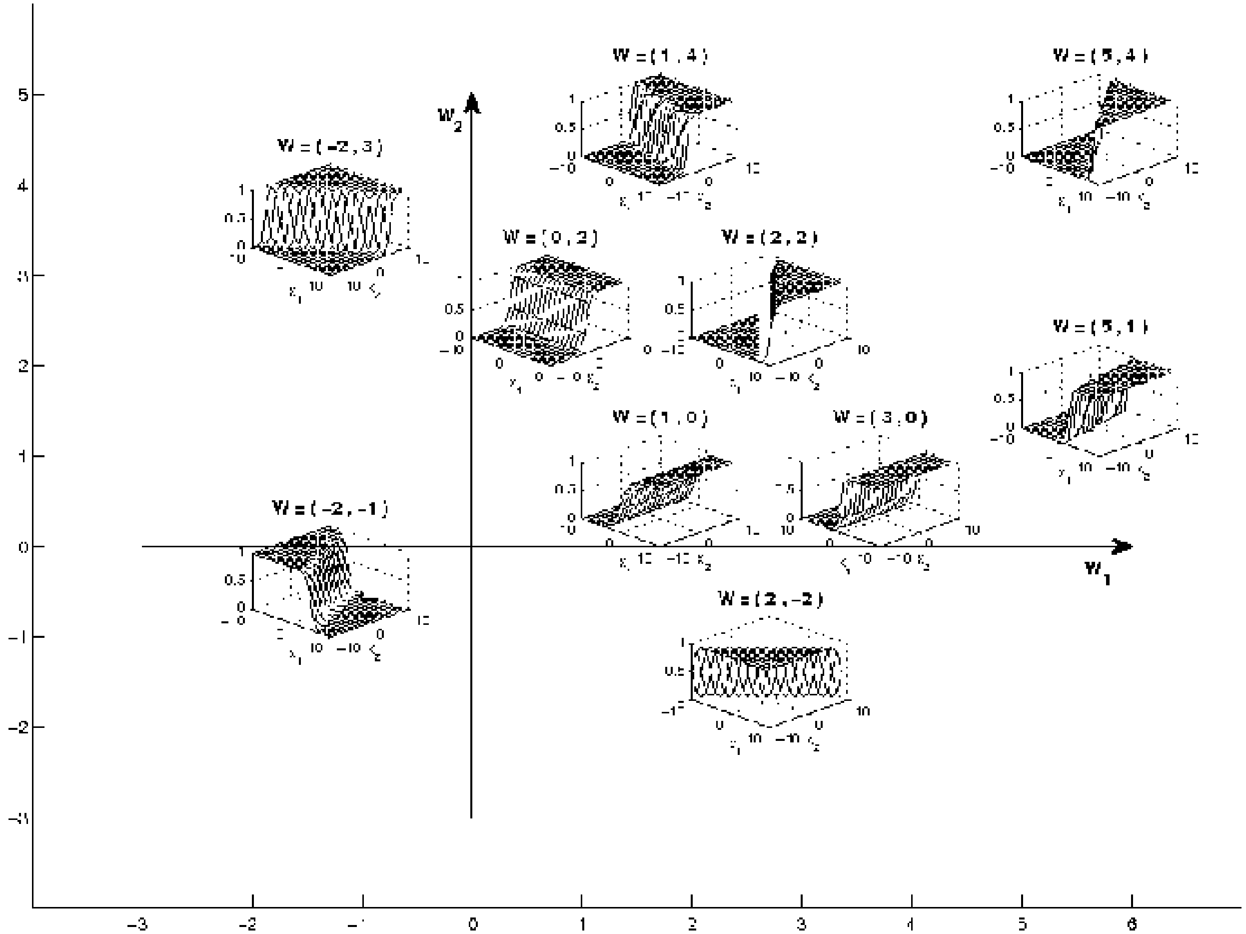
Logistic regression in 1d



SAT scores

Logistic regression in 2d

$$p(y = 1|\mathbf{x}, \mathbf{w}) = \sigma(w_0 + w_1 x_1 + w_2 x_2)$$



Notation

$$p(y|\mathbf{x}, \mathbf{w}) = \begin{cases} \sigma(\eta) & \text{if } y = 1 \\ 1 - \sigma(\eta) = \sigma(-\eta) & \text{if } y = 0 \end{cases}$$

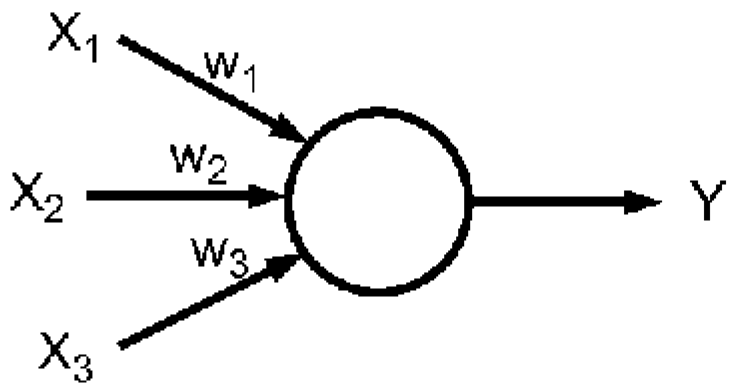
If we use $\tilde{y} \in \{-1, +1\}$ as the two labels instead of $y \in \{0, 1\}$:

$$p(\tilde{y}|\mathbf{x}, \mathbf{w}) \quad = \quad \sigma(\tilde{y}\eta)$$

$$\begin{array}{rcl} (-1, +1) & \overset{(y+1)/2}{\longrightarrow} & (0, 1) \\ (0, 1) & \overset{\text{sign}(y-0.5)}{\longrightarrow} & (-1, +1) \end{array}$$

Why the logistic function?

- McCulloch Pitts model of neuron



Log-odds ratio

$$\log \frac{p(y=1|\mathbf{x},\mathbf{w})}{p(y=0|\mathbf{x},\mathbf{w})} = \log \frac{e^\eta}{1+e^\eta} \frac{1+e^\eta}{1} = \log e^\eta = \eta$$

Thus if $w_j > 0$, then increasing x_j makes $y = 1$ more likely, and decreasing x_j makes $y = 0$ more likely; the opposite happens if $w_j < 0$. If $w_j = 0$, then x_j has no impact on the output, so feature j is irrelevant to predicting the output.

This time

- Logistic regression
- MLE
- Perceptron algorithm
- IRLS
- Multinomial logistic regression

MLE

Maximize log likelihood

$$\ell(\mathbf{w}) \stackrel{\text{def}}{=} \log p(D|\mathbf{w}) = \sum_{i=1}^{n} \log p(y_i|\mathbf{x}_i, \mathbf{w})$$

$$= \sum_{i=1}^{n} [y_i \log \mu_i + (1 - y_i) \log(1 - \mu_i)]$$

Minimize cross entropy

$$J(\mathbf{w}) = -\ell(\mathbf{w}) = -\sum_{i} [y_i \log \mu_i + (1 - y_i) \log(1 - \mu_i)]$$

Gradient

Gradient (homework)

$$\nabla J(\mathbf{w}) = \sum_{i=1}^{n} (\mu_i - y_i)\mathbf{x}_i = \mathbf{X}^T(\boldsymbol{\mu} - \mathbf{y})$$

Stochastic gradient descent

$$\mathbf{w} := \mathbf{w} - \alpha \mathbf{g}_i$$

Approximation

$$\mu_i \approx \hat{y}_i = \arg \max_{y \in \{0,1\}} p(y|\mathbf{x}_i, \mathbf{w})$$

Gives

$$\mathbf{g}_i = (\hat{y}_i - y_i)\mathbf{x}_i$$

Perceptron algorithm

$$\mathbf{g}_i = (\hat{y}_i - y_i)\mathbf{x}_i$$

Let $y \in \{-1, +1\}$.

$$\hat{y}_i = \text{sgn}(\mathbf{w}^T \mathbf{x}_i)$$

If $\hat{y}_i = y_i$, then $g_i = 0$. Otherwise

$$\mathbf{g}_i = -y_i \mathbf{x}_i$$

Hence

$$\mathbf{w} := \mathbf{w} + \alpha y_i \mathbf{x}_i$$

Set $\alpha = 1$.

Perceptron algorithm

```
function [w,b] = perceptron(X, y)
[n d] = size(X);
w = zeros(d,1);
b = zeros(1,1); % offset term
max_iter = 100;
for iter=1:max_iter
  errors = 0;
  for i=1:n
    xi = X(i,:)';
    yhati = sign(w'*xi + b);
    if ( y(i)*yhati <= 0 ) % made an error
      w = w + y(i) * xi;
      b = b + y(i);
      errors = errors + 1;
    end
  end
  fprintf('Iteration %d, errors = %d\n', iter, errors);
  if (errors==0)
    break;
  end
end
```

Convergence

- If linearly separable (so errors = 0), guaranteed to converge, but may do so slowly
- If not linearly separable, may not converge

This time

- Logistic regression
- MLE
- Perceptron algorithm
- IRLS
- Multinomial logistic regression

IRLS

- Iteratively reweighted least squares for finding the MLE for logistic regression
- Special case of Newton’s algorithm

Newton's method

- Consider a quadratic objective

$$f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T\mathbf{A}\mathbf{x} - \mathbf{b}^T\mathbf{x}$$

- Gradient methods may take many steps, but we can "hop" to the minimum in 1 step if we use

$$\begin{aligned} \mathbf{g}(\mathbf{x}) = \mathbf{A}\mathbf{x} - \mathbf{b} &= \mathbf{0} \\ \mathbf{x} &= \mathbf{A}^{-1}\mathbf{b} \end{aligned}$$

- In general, g(x) will not be linear in x, but we can linearize it. Alternatively we can approximate f by a quadratic.

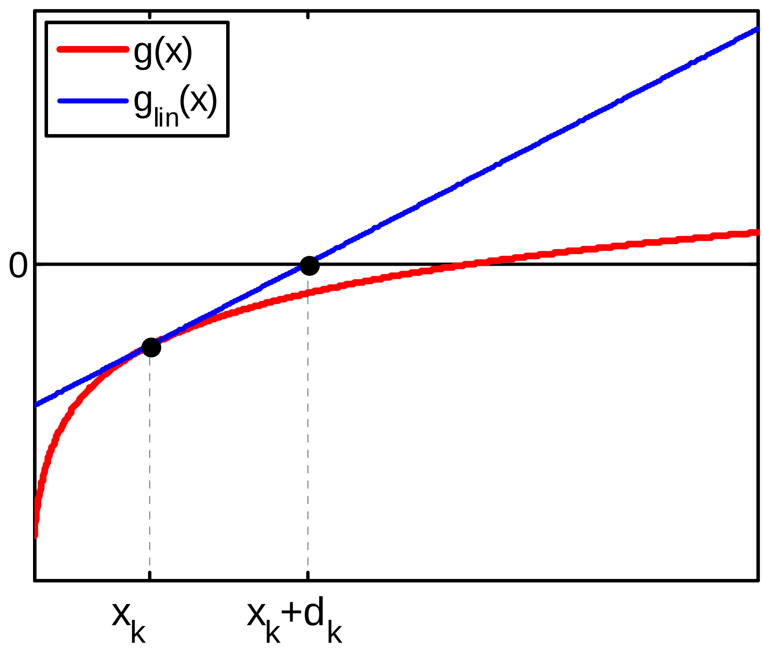
Linearize the gradient

- First order Taylor series approx of g(x) around x_k

$$\mathbf{g}(\mathbf{x}) \approx \mathbf{g}_k + \mathbf{H}_k(\mathbf{x} - \mathbf{x}_k)$$

$$\mathbf{g}(\mathbf{x}) = 0$$

$$\mathbf{x} = \mathbf{x}_k - \mathbf{H}_k^{-1}\mathbf{g}_k = \mathbf{x}_k + \mathbf{d}_k$$



$$g_{lin}(x) = g(x_k) + g'(x_k)(x - x_k)$$

Approximate the function

- Construct second order Taylor of f(x) around x_k

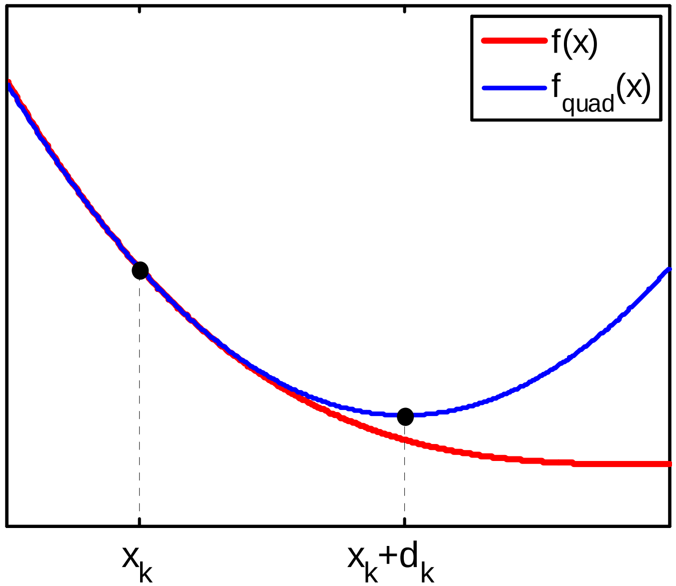
$$f_{quad}(\mathbf{x}) = f_k + \mathbf{g}_k^T(\mathbf{x} - \mathbf{x}_k) + \frac{1}{2}(\mathbf{x} - \mathbf{x}_k)^T \mathbf{H}_k (\mathbf{x} - \mathbf{x}_k)$$

$$f_{quad}(\mathbf{x}) = \mathbf{x}^T \mathbf{A} \mathbf{x} + \mathbf{b}^T \mathbf{x} + r$$

$$\mathbf{A} = \frac{1}{2}\mathbf{H}_k, \quad \mathbf{b} = \mathbf{g}_k - \mathbf{H}_k \mathbf{x}_k, \quad r = f_k - \mathbf{g}_k^T \mathbf{x}_k + \frac{1}{2}\mathbf{x}_k^T \mathbf{H}_k \mathbf{x}_k$$

Minimum

$$\mathbf{x} = -\frac{1}{2}\mathbf{A}^{-1}\mathbf{b} = \mathbf{x}_k - \mathbf{H}_k^{-1}\mathbf{g}_k$$



Newton's algorithm

Algorithm 1: Newton's method for minimizing a convex function

Initialize $\mathbf{x}_0$
for $k = 1, 2, \ldots$ *until convergence* **do**
    Evaluate $\mathbf{g}_k = \nabla f(\mathbf{x}_k)$, $\mathbf{H}_k = \nabla^2 f(\mathbf{x}_k)$
    Solve $\mathbf{d}_k = -\mathbf{H}^{-1}\mathbf{g}_k$
    Use line search to find stepsize α_k along $\mathbf{d}_k$
    $\mathbf{x}_k = \mathbf{x}_k + \alpha_k \mathbf{d}_k$

Use QR to solve H d_k = -g_k for d_k

Gradient and Hessian

$$\mathbf{g}(\mathbf{w}) \quad = \quad \sum_{i=1}^{n}(\mu_i - y_i)\mathbf{x}_i = \mathbf{X}^T(\boldsymbol{\mu} - \mathbf{y})$$

$$\mathbf{H} = \nabla_{\mathbf{w}}(g(\mathbf{w})^T) = \sum_i (\nabla_{\mathbf{w}}\mu_i)\mathbf{x}_i^T = \sum_i \mu_i(1-\mu_i)\mathbf{x}_i\mathbf{x}_i^T$$

$$\mathbf{H} = \mathbf{X}^T\mathbf{S}\mathbf{X}$$

$$\mathbf{S} \stackrel{\text{def}}{=} \text{diag}(\mu_1(1-\mu_1), \ldots, \mu_n(1-\mu_n))$$

Generic solver

Listing 1: Listing of `logregNLLgradHess`logregNLLgradHess

```
function [f,g,H] = logregNLLgradHess(beta, X, y, lambda)
% gradient and hessian of negative log likelihood for logistic regression
%
% Rows of X contain data
% y(i) = 0 or 1
% lambda is optional strength of L2 regularizer

if nargin < 4,lambda = 0; end
mu = 1 ./ (1 + exp(-X*beta)); % mu(i) = prob(y(i)=1|X(i,:))
f = -sum( (y.*log(mu+eps) + (1-y).*log(1-mu+eps))) + lambda/2*sum(beta.^2);
g = []; H  = [];
if nargout > 1
  g = X'*(mu-y) + lambda*beta;
end
if nargout > 2
  W = diag(mu .* (1-mu)); %  weight matrix
  H = X'*W*X + lambda*eye(length(beta));
end
```

Listing 2: :

```
opts = optimset('fminunc');
opts = optimset(opts, 'GradObj', 'on', 'Hessian', 'on');
w = zeros(d,1);
[w fval] = fminunc(@logregNLLgradHess, w, opts);
```

IRLS

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \mathbf{H}^{-1}\mathbf{g}_t$$

$$\begin{aligned}
\mathbf{w}_{t+1} &= \mathbf{w}_t + (\mathbf{X}^T\mathbf{S}_t\mathbf{X})^{-1}\mathbf{X}^T(\mathbf{y} - \boldsymbol{\mu}_t) \\
&= (\mathbf{X}^T\mathbf{S}_t\mathbf{X})^{-1}\left[(\mathbf{X}^T\mathbf{S}_t\mathbf{X})\mathbf{w}_t + \mathbf{X}^T(\mathbf{y} - \boldsymbol{\mu}_t)\right] \\
&= (\mathbf{X}^T\mathbf{S}_t\mathbf{X})^{-1}\mathbf{X}^T\left[\mathbf{S}_t\mathbf{X}\mathbf{w}_t + \mathbf{y} - \boldsymbol{\mu}_t\right] \\
\mathbf{w}_{t+1} &= (\mathbf{X}^T\mathbf{S}_t\mathbf{X})^{-1}\mathbf{X}^T\mathbf{S}_t\mathbf{z}_t \\
\mathbf{z}_t &\stackrel{\text{def}}{=} \mathbf{X}\mathbf{w}_t + \mathbf{S}_t^{-1}(\mathbf{y} - \boldsymbol{\mu}_t)
\end{aligned}$$

L2 regularization

- Needed to prevent overfitting and w -> inf

$$J(\mathbf{w}, \lambda) = -\left[\sum_{i=1}^{n} y_i \log \mu_i + (1 - y_i) \log(1 - \mu_i)\right] + \frac{\lambda}{2}||\mathbf{w}||_2^2$$

$$\mathbf{g} = \mathbf{X}^T(\mu - \mathbf{y}) + \lambda \mathbf{w}, \quad \mathbf{H} = \mathbf{X}^T \mathbf{S} \mathbf{X} + \lambda \mathbf{I}_d$$

This time

- Logistic regression
- MLE
- Perceptron algorithm
- IRLS
- Multinomial logistic regression

Multinomial logistic regression

- Y in {1,…,C} categorical

$$p(y = c|\mathbf{x}, \mathbf{W}) = \mathcal{S}(\mathbf{W}^T\mathbf{x})_c$$

$$\mathcal{S}(\boldsymbol{\eta})_c = \frac{e^{\eta_c}}{\sum_{c'=1}^{C} e^{\eta_{c'}}}$$

softmax

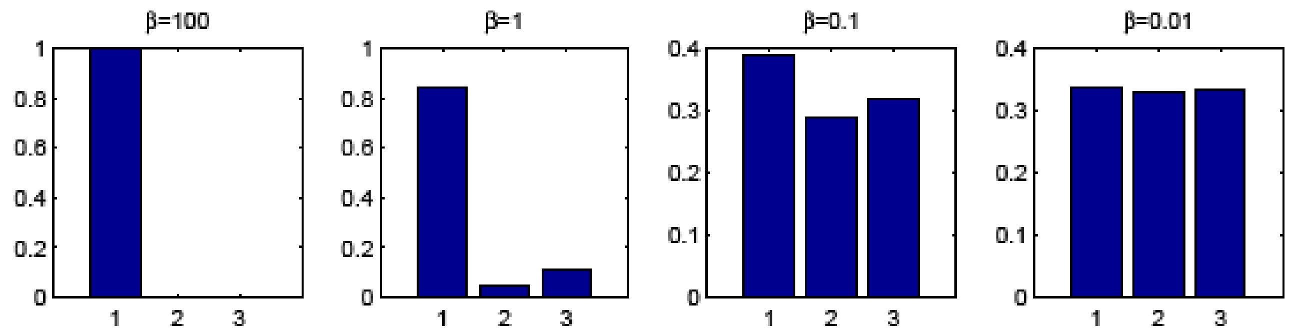
Binary case

$$\mathcal{S}(\mathbf{W}^T\mathbf{x})_1 = \frac{e^{\mathbf{w}_1^T\mathbf{x}}}{e^{\mathbf{w}_1^T\mathbf{x}} + e^{\mathbf{w}_0^T\mathbf{x}}} = \frac{1}{1 + e^{-(\mathbf{w}_1 - \mathbf{w}_0)^T\mathbf{x}}} = \sigma((\mathbf{w}_1 - \mathbf{w}_0)^T\mathbf{x})$$

Softmax function

$$\mathcal{S}(\boldsymbol{\eta})_c = \frac{e^{\eta_c}}{\sum_{c'=1}^{C} e^{\eta_{c'}}}$$

$$\mathcal{S}(\beta\boldsymbol{\eta})_c = \begin{cases} 1.0 & \text{if } c = \arg\max_{c'} \eta_{c'} \\ 0.0 & \text{otherwise} \end{cases}$$



MLE

$$\mu_{ik} = p(y = k|\mathbf{x}_i, \mathbf{W}) = \mathcal{S}(\boldsymbol{\eta}_i)_k$$
$$\boldsymbol{\eta}_i = \mathbf{W}^T \mathbf{x}_i$$
$$y_{ik} = I(y_i = k)$$

$$\ell(\mathbf{W}) = \sum_{i=1}^{n} \sum_{k=1}^{C} y_{ik} \log \mu_{ik} = \sum_{i=1}^{n} \left[\left(\sum_{k=1}^{C} y_{ik} \mathbf{w}_k^T \mathbf{x}_i \right) - \log \left(\sum_{j=1}^{C} \exp(\mathbf{w}_j^T \mathbf{x}_i) \right) \right]$$

Can compute gradient and Hessian and use Newton’s method

Can add L2 regularizer

Can use faster optimization methods eg bound optimization