Machine learning: lecture 6

Tommi S. Jaakkola
MIT CSAIL
tommi@csail.mit.edu

Topics

- Generalized linear models
  - logistic regression
  - gradient ascent, learning rate, convergence, examples
  - additive models, neural networks, back-propagation
- Regularization
  - basic idea
  - effective number of parameters

Review: logistic regression

- In a logistic regression model the conditional probability of the label y given the input example $\mathbf{x}$ is expressed as

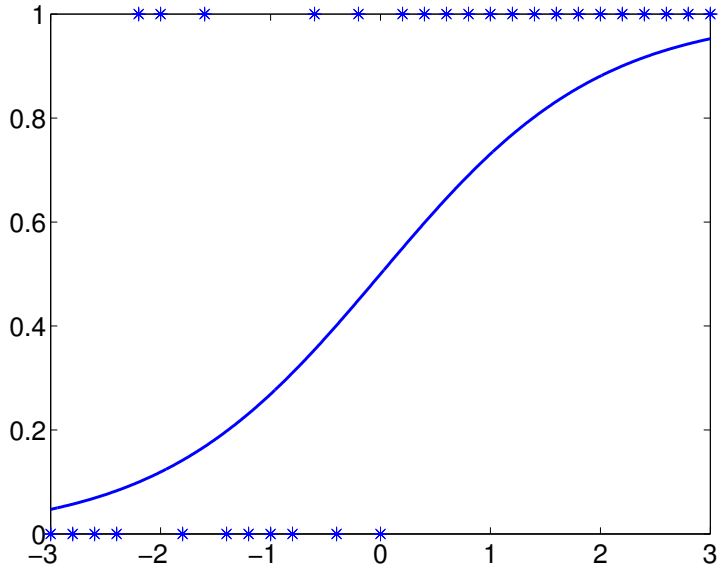
$$P(y = 1|\mathbf{x}, \mathbf{w}) = g\left(\, w_0 + w_1 x_1 + \ldots + w_d x_d \,\right)$$

where the coefficients $\mathbf{w}$ are the adjustable parameters.

The "squashing function"

$$g(z) = (1 + \exp(-z))^{-1}$$

known as the logistic function turns linear predictions into probabilities



Example problem

- The problem: classification of radar returns from the ionosphere (data is available from the UCI ML repository)
  - binary class label
  - 34 input "features" (2 values per radar pulse) defining the input vector $\mathbf{x} = [x_1, \ldots, x_{34}]^T$.
  - 200 training and 150 testing examples
- We would like to estimate a simple logistic regression model for this classification task

$$P(y = 1|\mathbf{x}, \mathbf{w}) = g\left(w_0 + w_1 x_1 + \ldots + w_d x_d \right)$$

where $d = 34$.

Fitting logistic regression models

- As in the case of linear regression models we can fit the logistic models using the maximum (conditional) log-likelihood criterion

$$l(D; \mathbf{w}) = \sum_{i=1}^{n} \log P(y_i | \mathbf{x}_i, \mathbf{w})$$

where

$$P(y = 1 | \mathbf{x}, \mathbf{w}) = g\left(w_0 + w_1 x_1 + \ldots + w_d x_d \right)$$

- The log-likelihood function $l(D; \mathbf{w})$ is a *jointly concave* function of the parameters $\mathbf{w}$; a number of optimization techniques are available for finding the maximizing parameters

Example method: gradient ascent

- We can, for example, maximize the log-likelihood by iteratively adjusting the parameters in small increments

  In each iteration we adjust $\mathbf{w}$ in the direction that increases the log-likelihood (towards the gradient):

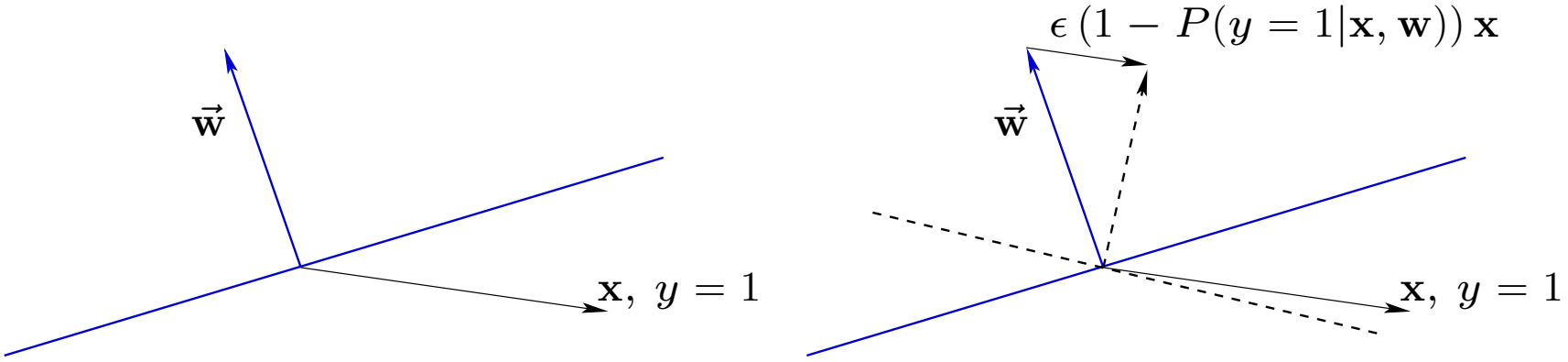
$$\begin{aligned}
\mathbf{w} &\leftarrow \mathbf{w} + \epsilon \frac{\partial}{\partial \mathbf{w}} \sum_{i=1}^{n} \log P(y_i | \mathbf{x}_i, \mathbf{w}) \\
&= \cdots \\
&= \mathbf{w} + \epsilon \sum_{i=1}^{n} \underbrace{\left(y_i - P(y_i | \mathbf{x}_i, \mathbf{w}) \right)}_{\text{prediction error}} \begin{bmatrix} 1 \\ \mathbf{x}_i \end{bmatrix}
\end{aligned}$$

  where ϵ is the *learning rate*.

Gradient ascent cont'd

- To understand the procedure graphically we can focus on a single example

$$\mathbf{w} \leftarrow \mathbf{w} + \epsilon \underbrace{\left(y_i - P(y_i|\mathbf{x}_i, \mathbf{w}) \right)}_{\text{prediction error}} \begin{bmatrix} 1 \\ \mathbf{x}_i \end{bmatrix}$$



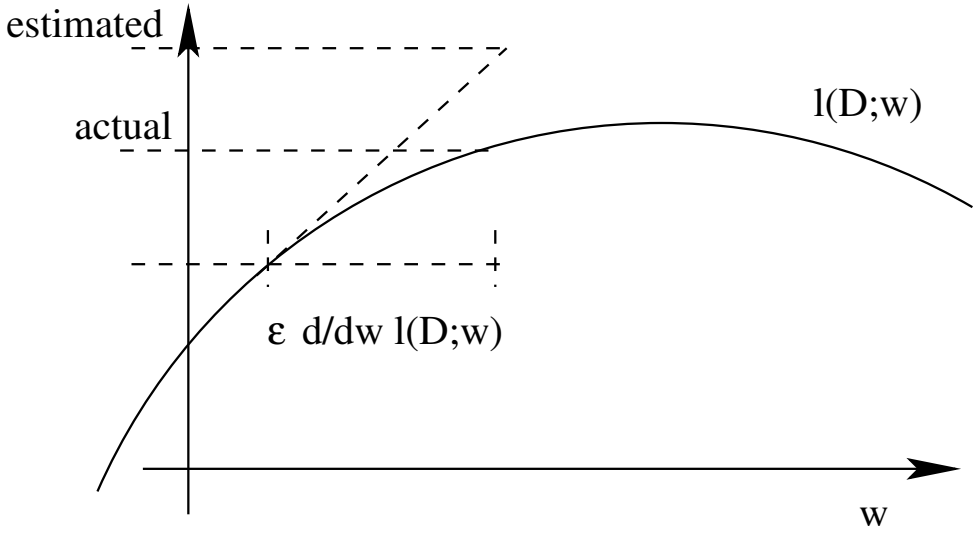
(no bias term included)

Setting the learning rate: Armijo rule

The learning rate in

$$\mathbf{w} \leftarrow \mathbf{w} + \epsilon \frac{\partial}{\partial \mathbf{w}} l(D; \mathbf{w})$$

"should" satisfy



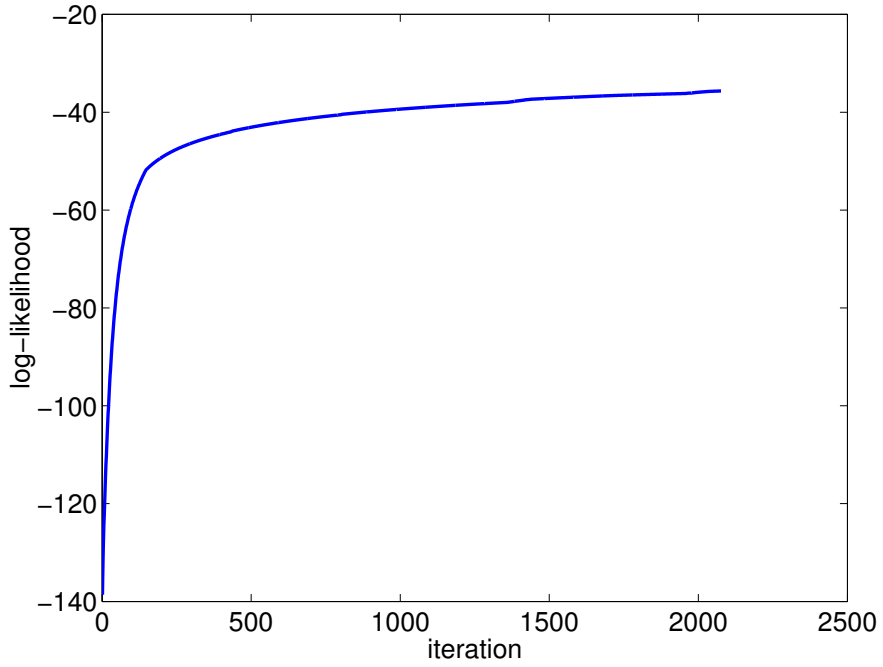
$$l\left(D;\, \mathbf{w} + \epsilon \frac{\partial}{\partial \mathbf{w}} l(D; \mathbf{w})\right) - l(D; \mathbf{w}) \geq \epsilon \cdot \frac{1}{2} \| \frac{\partial}{\partial \mathbf{w}} l(D; \mathbf{w}) \|^2$$

The Armijo rule suggests finding the smallest integer m such that $\epsilon = \epsilon_0 q^m,\ q < 1$ is a valid choice in this sense.

- Armijo rule is guaranteed to converge to a (local) maximum under certain technical assumptions

Example cont'd

- We get a monotonically increasing log-likelihood of the training labels as a function of the gradient ascent iterations



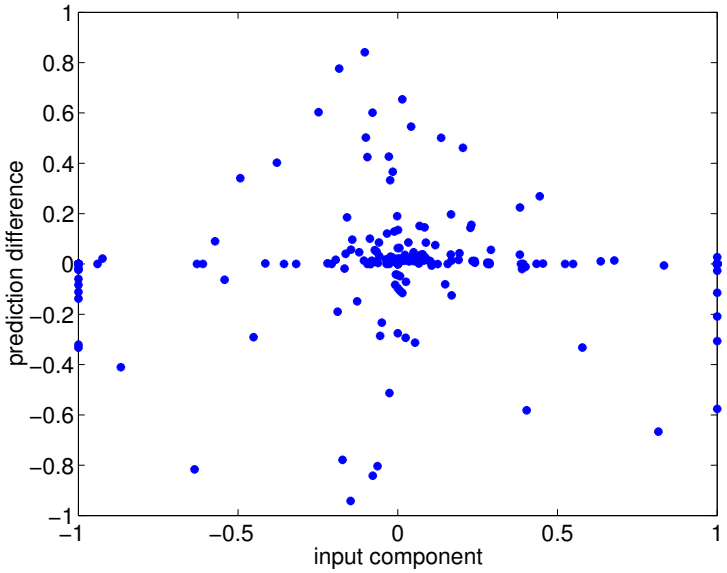
$$l(D; \mathbf{w}) = \sum_{i=1}^{n} \log P(y_i | \mathbf{x}_i, \mathbf{w})$$

Gradient ascent: convergence

- The gradient ascent learning method *converges* when there is no incentive to move the parameters in any particular direction:

$$\sum_{i=1}^{n} \underbrace{\left(y_i - P(y_i|\mathbf{x}_i, \hat{\mathbf{w}}) \right)}_{\text{prediction error}} \begin{bmatrix} 1 \\ \mathbf{x}_i \end{bmatrix} = 0$$

- This condition means again that the prediction error is uncorrelated with the components of the input vector



Additive models and classification

- Similarly to linear regression models, we can extend the logistic regression models to additive (logistic) models

$$P(y = 1|\mathbf{x}, \mathbf{w}) = g\left(w_0 + w_1\phi_1(\mathbf{x}) + \ldots w_m\phi_m(\mathbf{x}) \right)$$

- We are again free to choose the basis functions $\phi_i(\mathbf{x})$

Additive models and classification

- Similarly to linear regression models, we can extend the logistic regression models to additive (logistic) models

$$P(y = 1|\mathbf{x}, \mathbf{w}) = g\left(w_0 + w_1\phi_1(\mathbf{x}) + \ldots w_m\phi_m(\mathbf{x}) \right)$$

- We are again free to choose the basis functions $\phi_i(\mathbf{x})$

  For example, we can make them adjustable "feature detectors"

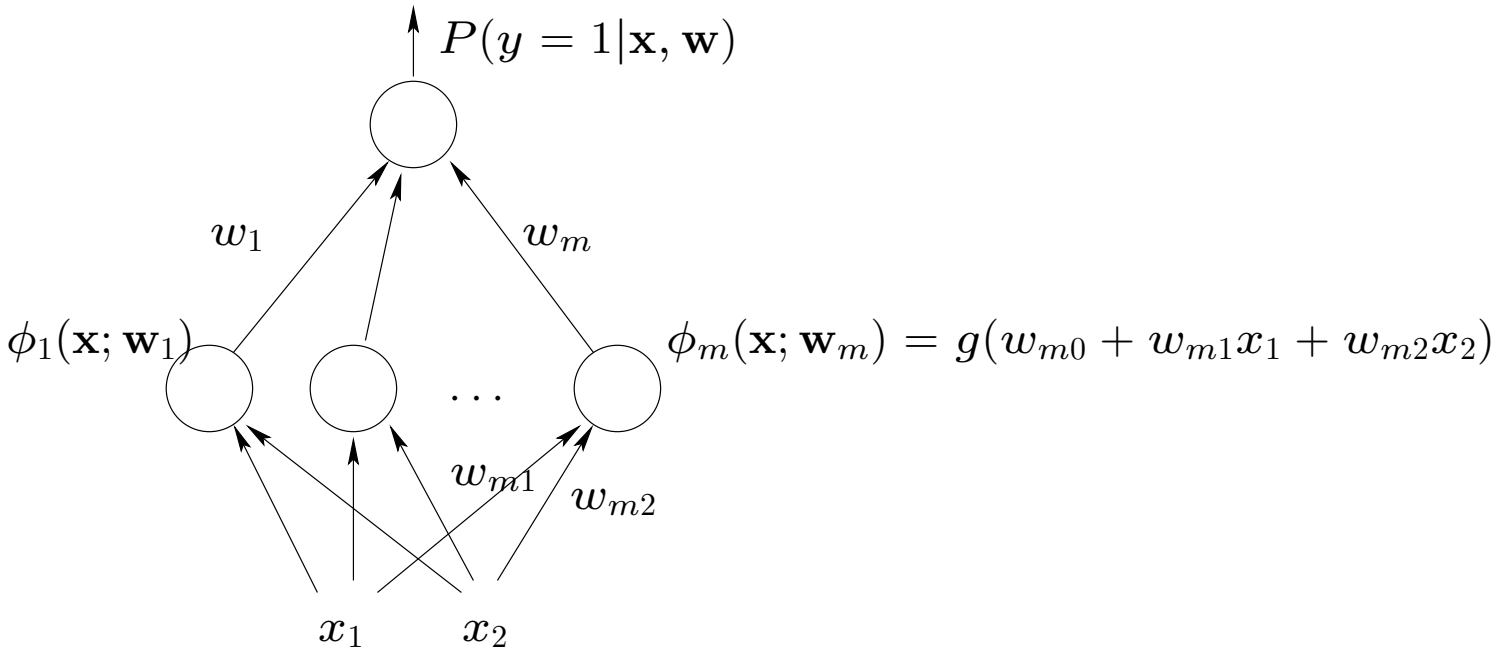
$$\phi_i(\mathbf{x}; \mathbf{w}_i) = g\left(w_{0i} + w_{i1}x_1 + w_{i2}x_2 \right)$$

A two layer neural network model

- In a neural network model, the basis functions themselves are adjustable (e.g., squashed linear regression models) representing the probability that a "feature" is present in the input

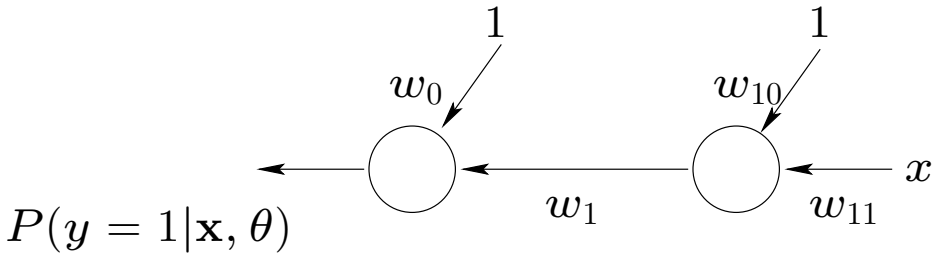
$$P(y = 1|\mathbf{x}, \theta) \;\; = \;\; g\left(w_0 + w_1\phi_1(\mathbf{x}; \mathbf{w}_1) + \ldots w_m\phi_m(\mathbf{x}; \mathbf{w}_m) \right)$$

where $\theta = \{\mathbf{w}, \mathbf{w}_1, \ldots, \mathbf{w}_m\}$ and



Parameter estimation

- We can estimate the parameters of the neural network model with a gradient ascent method as before.



$$P(y = 1|x, \theta) = g(w_0 + w_1\phi_1(x; \mathbf{w}_1)), \text{ where}$$
$$\phi_1(x; \mathbf{w}_1) = g(w_{10} + w_{11}x)$$

How do we evaluate the gradient or derivatives such as

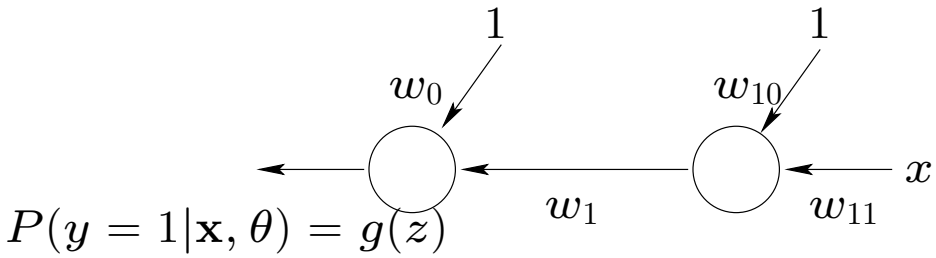
$$\frac{\partial}{\partial w_{11}} \log P(y|x, \theta) \, ?$$

Computing the gradient: back-propagation

- It is useful to think of the neural network model in terms of successive "inputs" and "outputs"

$$P(y=1|x,\theta) = g(\overbrace{w_0 + w_1\phi_1(x;\mathbf{w}_1)}^{\text{input } z}),$$

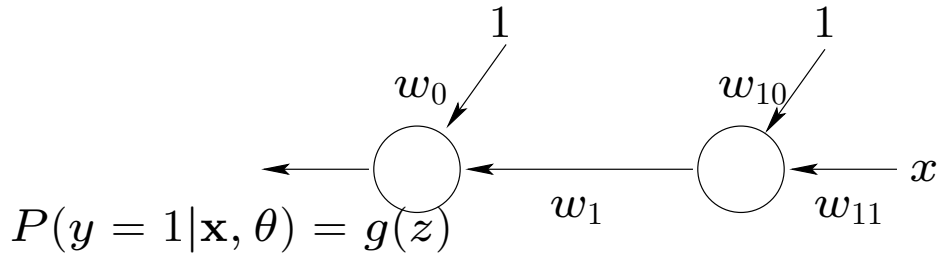
$$\phi_1(x;\mathbf{w}_1) = g(\underbrace{w_{10} + w_{11}x}_{\text{input } z_1})$$



$$z = w_0 + w_1 g(z_1)$$

$$z_1 = w_{10} + w_{11}x$$

Computing the gradient: back-propagation



$z = w_0 + w_1 g(z_1)$

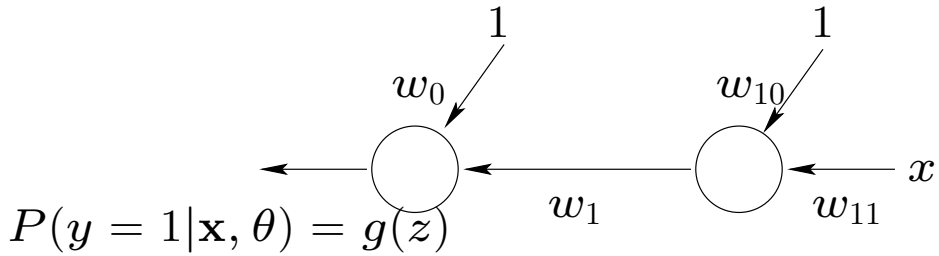
$z_1 = w_{10} + w_{11} x$

"input" derivatives:

$$\delta = \frac{\partial}{\partial z} \log P(y|x, \theta)$$

$$\delta_1 = \frac{\partial}{\partial z_1} \log P(y|x, \theta)$$

Computing the gradient: back-propagation



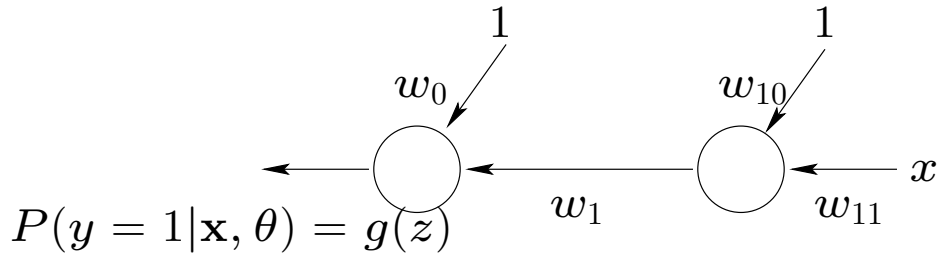
$z = w_0 + w_1 g(z_1)$

$z_1 = w_{10} + w_{11} x$

"input" derivatives:

$$\begin{aligned}
\delta &= \frac{\partial}{\partial z} \log P(y|x, \theta) \\
\delta_1 &= \frac{\partial}{\partial z_1} \log P(y|x, \theta) \\
&= \frac{\partial g(z_1)}{\partial z_1} \times
\end{aligned}$$

Computing the gradient: back-propagation



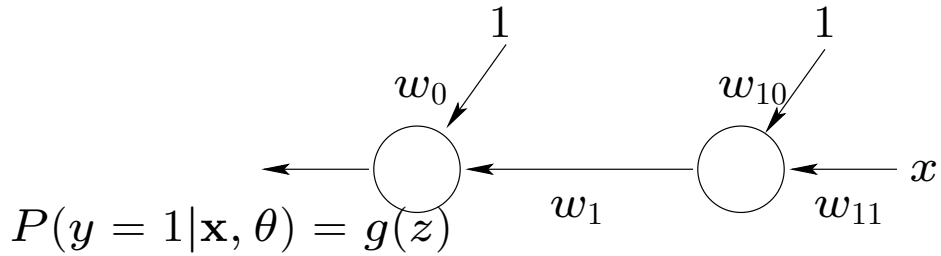
$z = w_0 + w_1 g(z_1)$

$z_1 = w_{10} + w_{11} x$

"input" derivatives:

$$\begin{aligned}
\delta &= \frac{\partial}{\partial z} \log P(y|x, \theta) \\
\delta_1 &= \frac{\partial}{\partial z_1} \log P(y|x, \theta) \\
&= \frac{\partial g(z_1)}{\partial z_1} \times \frac{\partial z}{\partial g(z_1)} \times
\end{aligned}$$

Computing the gradient: back-propagation



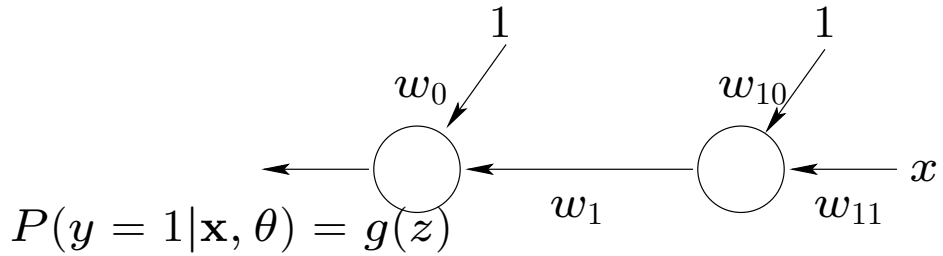
$z = w_0 + w_1 g(z_1)$

$z_1 = w_{10} + w_{11} x$

"input" derivatives:

$$\begin{aligned}
\delta &= \frac{\partial}{\partial z} \log P(y|x, \theta) \\
\delta_1 &= \frac{\partial}{\partial z_1} \log P(y|x, \theta) \\
&= \frac{\partial g(z_1)}{\partial z_1} \times \frac{\partial z}{\partial g(z_1)} \times \frac{\partial}{\partial z} \log P(y|x, \theta)
\end{aligned}$$

Computing the gradient: back-propagation



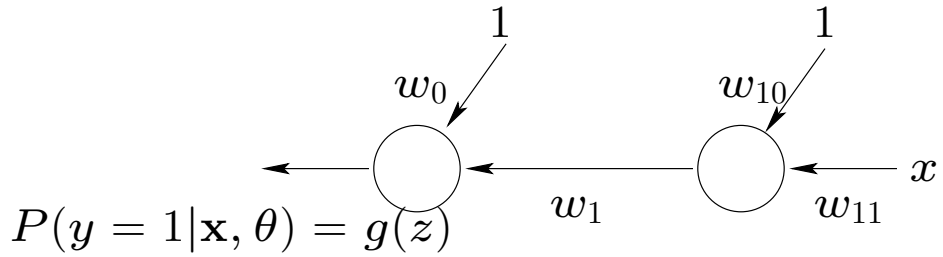
$z = w_0 + w_1 g(z_1)$

$z_1 = w_{10} + w_{11} x$

"input" derivatives:

$$
\begin{aligned}
\delta &= \frac{\partial}{\partial z} \log P(y|x, \theta) \\
\delta_1 &= \frac{\partial}{\partial z_1} \log P(y|x, \theta) \\
&= \frac{\partial g(z_1)}{\partial z_1} \times \frac{\partial z}{\partial g(z_1)} \times \frac{\partial}{\partial z} \log P(y|x, \theta) \\
&= g'(z_1) \times w_1 \times \delta
\end{aligned}
$$

Computing the gradient: back-propagation



"input" derivatives:

$$z = w_0 + w_1 g(z_1)$$

$$z_1 = w_{10} + w_{11} x$$

$$\begin{aligned}
\delta &= \frac{\partial}{\partial z} \log P(y|x, \theta) \\
\delta_1 &= \frac{\partial}{\partial z_1} \log P(y|x, \theta) \\
&= \frac{\partial g(z_1)}{\partial z_1} \times \frac{\partial z}{\partial g(z_1)} \times \frac{\partial}{\partial z} \log P(y|x, \theta) \\
&= g'(z_1) \times w_1 \times \delta
\end{aligned}$$

$$\frac{\partial}{\partial w_{11}} \log P(y|x, \theta) = \frac{\partial z_1}{\partial w_{11}} \times \frac{\partial}{\partial z_1} \log P(y|x, \theta) = x \times \delta_1$$

Topics

- Generalized linear models
  - logistic regression
  - gradient ascent, learning rate, convergence, examples
  - additive models, neural networks, back-propagation
- Regularization
  - basic idea
  - effective number of parameters

Regularization

- With more basis functions and parameters the models are more flexible to fit any relation (or noise) in the data.

$$P(y = 1|\mathbf{x}, \mathbf{w}) = g\left(w_0 + w_1\phi_1(\mathbf{x}) + \ldots w_m\phi_m(\mathbf{x}) \right)$$

- We need to find ways of constraining or "regularizing" such models in ways other than limiting the number of parameters

  We can limit the "effective" number of parameter choices, where the number depends on the resolution

Resolution: example

- The set of (0/1) coins parameterized by the probability p of getting "1"

  How many coins are there?

Resolution: example

- The set of (0/1) coins parameterized by the probability p of getting "1"

  How many coins are there?

  Case 1: uncountable number

Resolution: example

- The set of (0/1) coins parameterized by the probability p of getting "1"

  How many coins are there?

  Case 1: uncountable number
  Case 2: 9 coins $(p_1, \ldots, p_9)$ so that predictions of any other coin (indexed by p) is no more than $\epsilon = 0.1$ away

  for any p, $|p - p_j| \leq \epsilon$ for at least one j

Resolution: example

- The set of (0/1) coins parameterized by the probability p of getting "1"

  How many coins are there?

  Case 1: uncountable number

  Case 2: 9 coins $(p_1, \ldots, p_9)$ so that predictions of any other coin (indexed by p) is no more than $\epsilon = 0.1$ away

  $$\text{for any } p,\ |p - p_j| \leq \epsilon \text{ for at least one } j$$

  Case 3: only 1 coin if $\epsilon = 0.5$

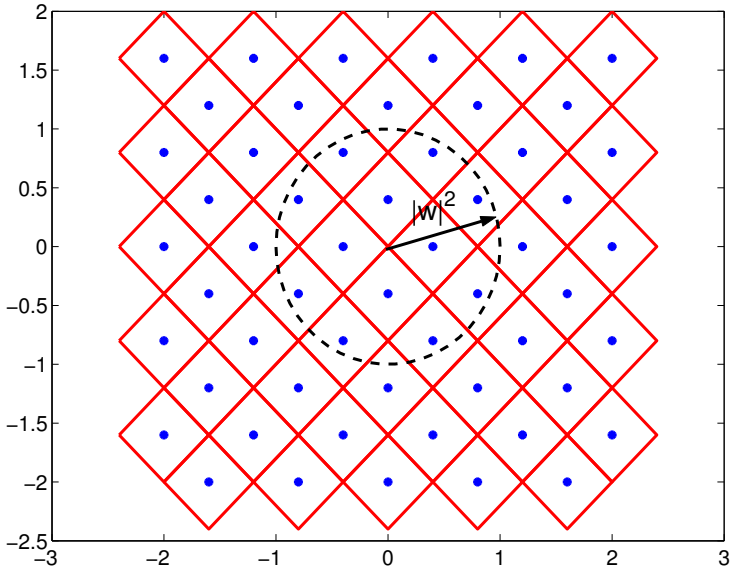
Logistic regression example

- Simple logistic regression model

$$P(y = 1|x, \mathbf{w}) = g(w_0 + w_1 x)$$

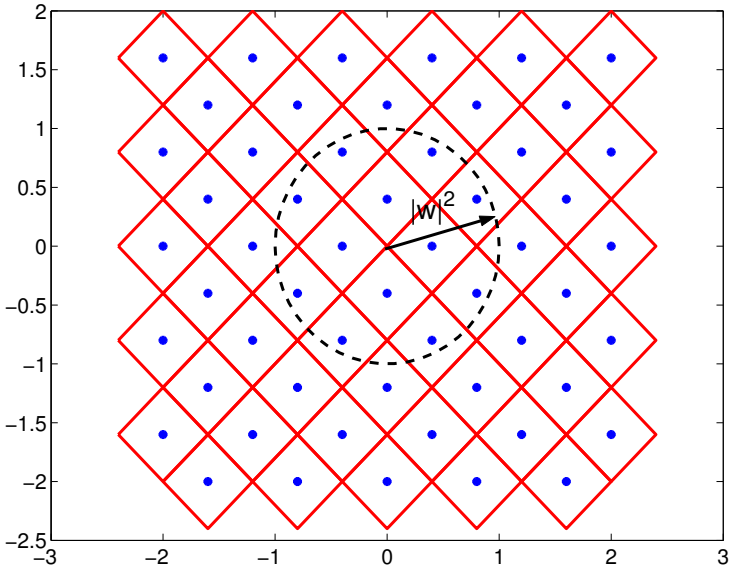
parameterized by $\mathbf{w} = (w_0, w_1)$. We assume that $x \in [-1, 1]$, i.e., that the inputs remain bounded.

- We can now divide the parameter space into regions with centers $\mathbf{w}_1, \mathbf{w}_2, \ldots$ such that the predictions of any $\mathbf{w}$ (for any $x \in [-1, 1]$) are close to those of one of the centers:



$$|\log P(y = 1|x, \mathbf{w}) - \log P(y = 1|x, \mathbf{w}_j)| \leq \epsilon$$

Limiting choices: regularization



- By constraining $\|\mathbf{w}\| \leq C$ for some *regularization parameter* C, we have fewer effective parameter choices in the logistic regression model

$$P(y = 1|x, \mathbf{w}) = g(w_0 + w_1 x)$$