Linear Regression Cost

$$C = \sum_{n=1}^{N} \left(y^{(n)} - \underline{w}^T \underline{x}^{(n)} \right)^2$$

$$\nabla_{\underline{w}} C = -2 \sum_{n=1}^{N} \left(y^{(n)} - \underline{w}^T \underline{x}^{(n)} \right) \underline{x}^{(n)}$$

Logistic Regression Cost

Likelihood of parameters given by probability of data:

$$\mathcal{L}(\underline{w}) = \prod_n p(y^{(n)} \mid \underline{x}^{(n)}, \underline{w}) = \prod_n \sigma\left(z^{(n)} \underline{w}^T \underline{x}^{(n)} \right)$$

↑ Label $\in \{0, 1\}$ ↑ Label $\in \{-1, +1\}$, $z^{(n)} = 2y^{(n)} - 1$

⇓ cost function $-\log(\cdot)$

$$NLL = -\sum_n \log \underbrace{\sigma\left(z^{(n)} \underline{w}^T \underline{x}^{(n)} \right)}_{\sigma_n}$$

$$\nabla_{\underline{w}} NLL = -\sum_n (1 - \sigma_n) z^{(n)} \underline{x}^{(n)}$$

Gradient Descent

Initialize $\underline{w}$, e.g. to $\underline{0}$ (for these models)

for $t = 1$ to max_iters:

$$\underline{w} \leftarrow \underline{w} - \eta_t \nabla_{\underline{w}} NLL$$

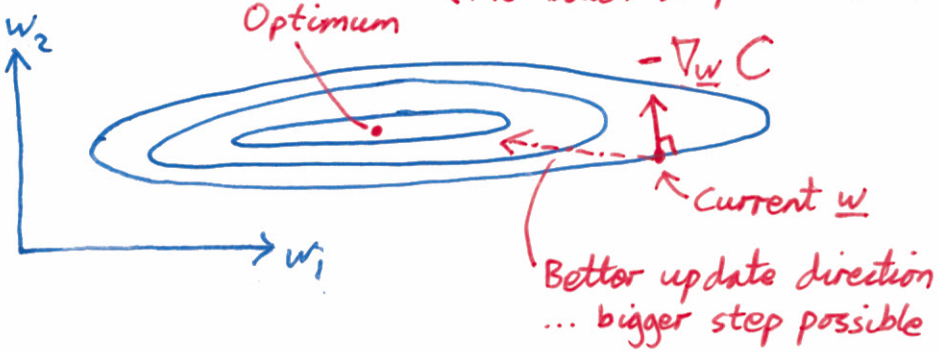
(or C, any cost function)

(η_t = small constant, or clever schedule)

Might terminate early if "converged."

Gradient Descent is awful

(The "batch" simple version.)



Eg. discussion §10.6 "Numerical Recipes in Fortran" (2nd Ed.)

⇒ Better "batch" methods

↳ C and $\nabla_{\underline{w}} C$ use whole dataset

⇒ "Stochastic"/"Online" or "Mini-batch"

Only look at one or some examples for each update of parameters $\underline{w}$

Stochastic Gradient Descent

Average Gradient of examples:

$$\frac{1}{N} \nabla_{\underline{w}} C = \frac{1}{N} \sum_{n=1}^{N} \nabla_{\underline{w}} C_n$$

eg. $-(1-\sigma_n)\, z^{(n)}\, \underline{x}^{(n)}$

C_n: Cost for n^{th} example

Monte Carlo Approximation:

Mini-batch of B examples at random:

$$\approx \frac{1}{B} \sum_{b=1}^{B} \nabla_{\underline{w}} C_b$$

Very approx. stochastic estimate, set $B = 1$

SGD stochastic gradient uses estimate in simple steepest gradient descent.

Often in practice we loop examples in order rather than sampling. Maybe randomly permute data every so often.

Softmax Regression

Multi-class classification

Target $y = [0\ 0\ 0 \cdots 0\ 1\ 0 \cdots 0]^T$

c^{th} location if the example has class c.

Fit $\underline{f}$, which is our model:

$$P(\text{class } c \mid \underline{x}) = f_c(\underline{x})$$

Each f_k has to be positive

$$f_k \propto e^{\underline{w}^{(k)T}\underline{x}}$$

We want $\underline{f}$ to be normalized:

$$\sum_{k=1}^{K} f_k = 1$$

$$f_k = \frac{e^{\underline{w}^{(k)T}\underline{x}}}{\sum_{k'} e^{\underline{w}^{(k')T}\underline{x}}}$$] Softmax

Model has parameters:

$$W = \{\underline{w}^{(k)}\}_{k=1}^{K}$$

P(Training data | W)

Maximize likelihood of W

If one example at $\underline{x}$ and we see its label: class c

Log prob. of this example under model

$$\log f_c = \underline{w}^{(c)T}\underline{x} - \log \sum_{k'} e^{\underline{w}^{(k')T}\underline{x}}$$

$$\nabla_{\underline{w}^{(k)}} \log f_c = \underbrace{\delta_{kc}}_{\text{Kronecker delta}} \underline{x} - \frac{1}{\sum_{k'} e^{\underline{w}^{(k')T}\underline{x}}} e^{\underline{w}^{(k)T}\underline{x}} \underline{x}$$

$$= (y_k - f_k)\underline{x}$$

Logistic Regression?

Two classes

$$p(y=1 \mid \underline{x}, W) = \frac{e^{\underline{w}^{(1)T}\underline{x}}}{e^{\underline{w}^{(1)T}\underline{x}} + e^{\underline{w}^{(0)T}\underline{x}}}$$

$$= \frac{1}{1 + e^{\underline{w}^{(0)T}\underline{x} - \underline{w}^{(1)T}\underline{x}}}$$

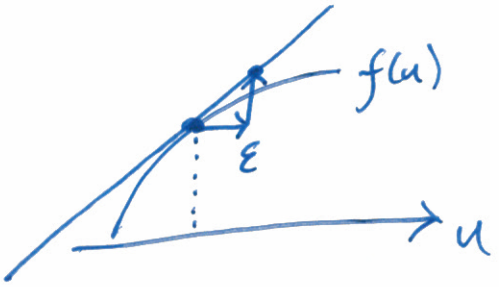
logistic sigmoid

$$= \sigma\left(\underbrace{(\underline{w}^{(1)} - \underline{w}^{(0)})^T}_{\text{"}\underline{w}\text{"}}\underline{x}\right)$$

$$\sigma(a) = \frac{1}{1+e^{-a}}$$

Same as logistic regression, but with redundant parameters.

Check your derivatives



$$f'(u) \approx \frac{f(u+\varepsilon) - f(u)}{\varepsilon}$$

correct $\lim \varepsilon \to 0$

approx $\varepsilon = 10^{-5}$