Lecture 18: Kernel perceptron, Logistic Regression, Cross Entropy Minimization, Gradient Descent

Instructor: Prof. Ganesh Ramakrishnan

Non-linear perceptron?

- Kernelized perceptron[1]: (Recall from non-param regression)

$$f(x) = \underbrace{\text{sign}}_{\text{categorical mapping}}\left[\sum_i y_i \alpha_i K(x^{(i)}, x) + b\right]$$

Q: What would updates for α_i look like.. let us try & inspire α_i updates from perceptron

Recall: $w^{(k+1)} = w^{(k)} + \eta \phi(x) y$ s.t $y (w^{(k)})^T \phi(x) < 0$

Now: Hint: Consider effect of stochastic update on

$$f(x): f^{(k+1)}(x) = \underbrace{(w^{(k+1)})^T}_{\text{Expanding in terms of updates so far}} \phi(x) = \left(w^{0^T}\phi(x) + \sum_j \eta y^{(j)} \underbrace{\phi^T(x^{(j)})\phi(x)}_{K(x^{(j)}, x)}\right)$$

$\alpha_j = \alpha_j + 1$

[1]The first kernel classification learner, was invented in 1964

Non-linear perceptron?

- Kernelized perceptron[1]: $f(\mathbf{x}) = sign\left(\sum_i \alpha_i y_i K(\mathbf{x}, \mathbf{x}_i) + b\right)$
  - INITIALIZE: α=zeroes()
  - REPEAT: for $< \mathbf{x}_i, y_i >$
    - If $sign\left(\sum_j \alpha_j y_j K(\mathbf{x}_j, \mathbf{x}_j) + b\right) \neq y_i$
    - then, $\alpha_i = \alpha_i + 1$
    - endif

$[w^{(0)} = [0 \quad 1]]$

$\underline{sign(0) = 0}$

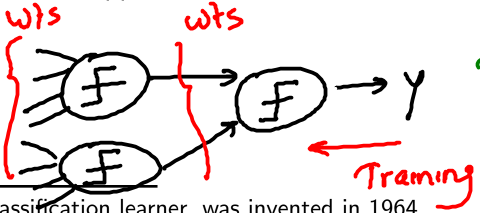
Alternative

$f(x) = sign\left(\sum_i \alpha_i K(x, x_i) + b\right)$

$\& \ \alpha_i = \alpha_i + y_i$

- Convergence is matter of Tutorial 6, Problem 3.
- Any other non-linear approach?

Neural nets?



For neural network training, we would like f to become soft, differentiable

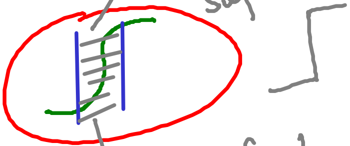
Sigmoidal

[1] The first kernel classification learner, was invented in 1964

Non-linear perceptron?

- Kernelized perceptron[1]: $f(\mathbf{x}) = sign\left(\underbrace{\sum_i \alpha_i y_i K(\mathbf{x}, \mathbf{x}_i) + b}_{S}\right)$
  - INITIALIZE: α=zeroes()
  - REPEAT: for $< \mathbf{x}_i, y_i >$
    - If $sign\left(\sum_j \alpha_j y_j K(\mathbf{x}_j, \mathbf{x}_j) + b\right) \neq y_i$
    - then, $\alpha_i = \alpha_i + 1$
    - endif
- Convergence is matter of Tutorial 6, Problem 3.
- Any other non-linear approach? **Ans:** Neural Networks: Cascade of layers of perceptrons giving you non-linearity.
- To handle cascades of perceptrons effectively, we need to make the sign fn soft & differentiable

so that better approximates step fn



Design the fn to reduce width of gray area

$\frac{1}{1+e^{-s}} \in (0,1)$, $\tanh \in (-1,1)$ $\frac{e^s - e^{-s}}{e^s + e^{-s}}$

Decays faster

$\frac{d}{dx} = \frac{e^{-s}}{(1+e^{-s})^2}$

[1]The first kernel classification learner, was invented in 1964

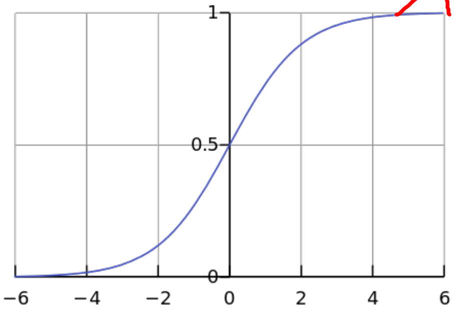
Non-linear perceptron?

- Kernelized perceptron[1]: $f(\mathbf{x}) = sign\left(\sum_i \alpha_i y_i K(\mathbf{x}, \mathbf{x}_i) + b\right)$
  - INITIALIZE: α=zeroes()
  - REPEAT: for $< \mathbf{x}_i, y_i >$
    - If $sign\left(\sum_j \alpha_j y_j K(\mathbf{x}_j, \mathbf{x}_i) + b\right) \neq y_i$
    - then, $\alpha_i = \alpha_i + 1$
    - endif
- Convergence is matter of Tutorial 6, Problem 3.
- Any other non-linear approach? **Ans:** Neural Networks: Cascade of layers of perceptrons giving you non-linearity.
- To handle cascades of perceptrons effectively, we need to make the perceptron and error (objective) function differentiable.
  We next discuss the specific sigmoidal percentron used most often in Neural Networks

[1]The first kernel classification learner, was invented in 1964

Sigmoidal (perceptron) Classifier

1. **(Binary) Logistic Regression**, abbreviated as **LR** is a single node perceptron-like classifier, but with....
   - *sign* $\left((\mathbf{w}^*)^T\phi(x)\right)$ replaced by $f_{\mathbf{w}}(\mathbf{x}) = f\left((\mathbf{w}^*)^T\phi(\mathbf{x})\right)$ where $f(s)$ is the sigmoid function: $f(s) = \frac{1}{1+e^{-s}}$
2. $f_{\mathbf{w}}(\mathbf{x}) = \frac{1}{1+e^{-(\mathbf{w}^*)^T\phi(\mathbf{x})}} \in [0,1]$ can be intepreted as $Pr(y=1|\mathbf{x})$ → Param p in Bernoulli
   - Then $Pr(y=0|x) = ?$ $\frac{e^{-w^T\phi(x)}}{1+e^{-w^T\phi(x)}}$



1. Estimator $\widehat{\mathbf{w}}$ is a function of the dataset $\mathcal{D} = \{(\phi(\mathbf{x}^{(1)}, y^{(1)}), (\phi(\mathbf{x}^{(2)}, y^{(2)}), \ldots, (\phi(\mathbf{x}^{(m)}, y^{(m)}))\}$
   - Estimator $\widehat{\mathbf{w}}$ is meant to approximate the parameter $\mathbf{w}$.
2. Maximum Likelihood Estimator: Estimator $\widehat{\mathbf{w}}$ that maximizes the likelihood $L(\mathcal{D}; \mathbf{w})$ function.

   Likelihood of w given $\mathcal{D}$

   - Assumes that all the instances $(\phi(\mathbf{x}^{(1)}, y^{(1)}), (\phi(\mathbf{x}^{(2)}, y^{(2)}), \ldots, (\phi(\mathbf{x}^{(m)}, y^{(m)}))$ in $\mathcal{D}$ are all independent and identically distributed (iid)
   - Thus, Likelihood is the probability of $\mathcal{D}$ under iid assumption:

$$\hat{\mathbf{w}} = \operatorname*{argmax}_{\mathbf{w}} L(\mathcal{D}, \mathbf{w}) = \operatorname*{argmax}_{w} \prod_{i=1}^{m} \left[Pr(y^{(i)}=1 \mid x^{(i)})\right]^{y^{(i)}} * \left[Pr(y^{(i)}=0 \mid x^{(i)})\right]^{(1-y^{(i)})}$$

$$= \operatorname*{argmax}_{w} \prod_{i=1}^{m} \left[f_w(x^{(i)})\right]^{y^{(i)}} \left[1-f_w(x^{(i)})\right]^{(1-y^{(i)})}$$

$$= \operatorname*{argmin}_{w} - \sum_{i=1}^{m} y^{(i)} \log\left(f_w(x^{(i)})\right) + (1-y^{i}) \log\left[1-f_w(x^{i})\right]$$

Logistic Regression: The Sigmoidal (perceptron) Classifier

1. Estimator $\widehat{\mathbf{w}}$ is a function of the dataset $\mathcal{D} = \{(\phi(\mathbf{x}^{(1)}, y^{(1)}), (\phi(\mathbf{x}^{(2)}, y^{(2)}), \ldots, (\phi(\mathbf{x}^{(m)}, y^{(m)}))\}$
   - Estimator $\widehat{\mathbf{w}}$ is meant to approximate the parameter $\mathbf{w}$.
2. Maximum Likelihood Estimator: Estimator $\widehat{\mathbf{w}}$ that maximizes the likelihood $L(\mathcal{D}; \mathbf{w})$ function.
   - Assumes that all the instances $(\phi(\mathbf{x}^{(1)}, y^{(1)}), (\phi(\mathbf{x}^{(2)}, y^{(2)}), \ldots, (\phi(\mathbf{x}^{(m)}, y^{(m)}))$ in $\mathcal{D}$ are all independent and identically distributed (iid)
   - Thus, Likelihood is the probability of $\mathcal{D}$ under iid assumption: $\hat{\mathbf{w}} = \underset{\mathbf{w}}{\text{argmax}}\ L(\mathcal{D}, \mathbf{w}) = \text{argmax}_{\mathbf{w}} \prod_{i=1}^{m} p(y^{(i)} | \phi(\mathbf{x}^{(i)})) =$
     $\text{argmax}_{\mathbf{w}} \prod_{i=1}^{m} \left(\frac{1}{1+e^{-(w)^T \phi(x^{(i)})}}\right)^{y^{(i)}} \left(\frac{e^{-(w)^T \phi(x^{(i)})}}{1+e^{-(w)^T \phi(x^{(i)})}}\right)^{1-y^{(i)}}$
     = argmax of w over bernoulli trials $y^{(i)}$ with parameter $f_{\mathbf{w}}\left(\mathbf{w}^T \phi(x^{(i)})\right)$

1. Thus, Maximum Likelihood Estimator for $\mathbf{w}$ is

$$\hat{\mathbf{w}} = \underset{\mathbf{w}}{\operatorname{argmax}}\ L(\mathcal{D}, w) = \underset{\mathbf{w}}{\operatorname{argmax}} \prod_{i=1}^{m} p(y^{(i)}|\phi(\mathbf{x}^{(i)}))$$

$$= \underset{\mathbf{w}}{\operatorname{argmax}} \prod_{i=1}^{m} \left(\frac{1}{1+e^{-\mathbf{w}^T\phi(\mathbf{x}^{(i)})}}\right)^{y^{(i)}} \left(\frac{e^{-\mathbf{w}^T\phi(\mathbf{x}^{(i)})}}{1+e^{-\mathbf{w}^T\phi(\mathbf{x}^{(i)})}}\right)^{1-y^{(i)}}$$

$$= \underset{\mathbf{w}}{\operatorname{argmax}} \underline{\prod_{i=1}^{m} \left(f_{\mathbf{w}}\left(\mathbf{x}^{(i)}\right)\right)^{y^{(i)}} \left(1 - f_{\mathbf{w}}\left(\mathbf{x}^{(i)}\right)\right)^{1-y^{(i)}}}$$

$$= \underset{w}{\operatorname{argmin}} - \sum_{i} y^{(i)} \log\left(f_w(x^{(i)})\right) + (1-y^{(i)}) \log\left[1-f_w(x^{(i)})\right]$$

Training LR

1. Thus, Maximum Likelihood Estimator for $\mathbf{w}$ is

$$\hat{\mathbf{w}} = \underset{\mathbf{w}}{\operatorname{argmax}}\ L(\mathcal{D}, w) = \underset{\mathbf{w}}{\operatorname{argmax}} \prod_{i=1}^{m} p(y^{(i)}|\phi(\mathbf{x}^{(i)}))$$

$$= \underset{\mathbf{w}}{\operatorname{argmax}} \prod_{i=1}^{m} \left(\frac{1}{1+e^{-\mathbf{w}^T\phi(\mathbf{x}^{(i)})}}\right)^{y^{(i)}} \left(\frac{e^{-\mathbf{w}^T\phi(\mathbf{x}^{(i)})}}{1+e^{-\mathbf{w}^T\phi(\mathbf{x}^{(i)})}}\right)^{1-y^{(i)}}$$

$$= \underset{\mathbf{w}}{\operatorname{argmax}}\ \prod_{i=1}^{m} \left(f_{\mathbf{w}}\left(\mathbf{x}^{(i)}\right)\right)^{y^{(i)}} \left(1 - f_{\mathbf{w}}\left(\mathbf{x}^{(i)}\right)\right)^{1-y^{(i)}}$$

2. Maximizing the likelihood $\Pr(\mathcal{D}; \mathbf{w})$ w.r.t $\mathbf{w}$, is the same as minimizing the negative log-likelihood $E(\mathbf{w}) = -\frac{1}{m} \log \Pr(\mathcal{D}; \mathbf{w})$ w.r.t $\mathbf{w}$.
   - Derive the expression for $E(\mathbf{w})$. $\rightarrow$ Error fn
   - $E(\mathbf{w})$ is called the cross-entropy loss function

1. Cross-entropy[2] is the average number of bits needed to identify an event (example $\mathbf{x}$) drawn from the (data) set $\mathcal{D}$, if a coding scheme is used that is optimized for a modeled probability distribution $\Pr(y|\mathbf{w}, \phi(.))$, rather than the 'true' distribution $\Pr(y|\mathcal{D})$.

$$E(\mathbf{w}) = \mathbf{E}_{\Pr(y|\mathcal{D})}\left[-\log \Pr(y|\mathbf{w}, \phi(.))\right] \quad (1)$$

equality difficult for LR since $f_w \in (0,1)$

2. The Cross-entropy Loss function:

$$-\sum_i \underbrace{y^{(i)}}_{\Pr_O(Y=1|x^{(i)})} \log \big(\underbrace{f_w(x^{(i)})}_{\Pr_M(Y=1|x^{(i)})}\big) + \underbrace{(1-y^{(i)})}_{\Pr_O(Y=0|x^{(i)})} \log\big(\underbrace{1-f_w(x^{(i)})}_{\Pr_M(Y=0|x^{(i)})}\big) \geq \underline{H(\Pr_O)}$$

observed or Empirical distr

$\Pr_M(.)$ is model distr.

In some sense perceptron tries to achieve $H(\Pr_O)$

[2] https://en.wikipedia.org/wiki/Cross_entropy

$$CE(p,q) = -\sum_x p(x)\log(q(x))$$

Expected # of bits to specify through q (over p)

p = observed distr

q = hypothesized distr

$\forall p,q$

$$CE(p,q) = \underbrace{H(p)}_{-\sum_x p(x)\log p(x)} + \underbrace{KL(p\|q)}_{\sum_x p(x)\log\frac{p(x)}{q(x)}}$$

Expected # of bits required to specify an outcome driven by p

$$KL(p\|q) \geq 0$$

$$KL(p\|q) = 0 \text{ iff } p = q$$

ie $p(x) = q(x)\ \forall x$

$$\Rightarrow CE(p,q) \geq H(p)$$

equality holds iff $p(x) = q(x)\ \forall x$

Minimizing negative Log-likelihood for LR

1. Cross-entropy[2] is the average number of bits needed to identify an event (example $\mathbf{x}$) drawn from the (data) set $\mathcal{D}$, if a coding scheme is used that is optimized for a modeled probability distribution $\Pr(y|\mathbf{w}, \phi(.))$, rather than the 'true' distribution $\Pr(y|\mathcal{D})$.

$$\log(e^s) = s$$

$$E(\mathbf{w}) = \mathbf{E}_{\Pr(y|\mathcal{D})}\left[-\log \Pr(y|\mathbf{w}, \phi(.))\right] \qquad (1)$$

2. The Cross-entropy Loss function:

$$\log(a/b) = \log a - \log b$$

$$E(\mathbf{w}) = -\left[\frac{1}{m}\sum_{i=1}^{m}\left(y^{(i)} \log \overbrace{f_{\mathbf{w}}\left(\mathbf{x}^{(i)}\right)}^{\frac{e^{w^T\phi(x^i)}}{1+e^{w^T\phi(x^i)}}} + \left(1-y^{(i)}\right) \log \overbrace{\left(1 - f_{\mathbf{w}}\left(\mathbf{x}^{(i)}\right)\right)}^{\frac{1}{1+e^{w^T\phi(x^i)}}}\right)\right] \qquad (2)$$

with some simplification,

$$E(w) = -\frac{1}{m}\sum_{i=1}^{m}\left[\underbrace{y^{(i)} w^T\phi(x^{(i)})}_{\text{Like perceptron loss}} - \underbrace{\log\left(1+e^{w^T\phi(x^{(i)})}\right)}_{\text{The new smoothing component!}}\right]$$

[2] https://en.wikipedia.org/wiki/Cross_entropy

Minimizing negative Log-likelihood for LR

1. Cross-entropy[2] is the average number of bits needed to identify an event (example $\mathbf{x}$) drawn from the (data) set $\mathcal{D}$, if a coding scheme is used that is optimized for a modeled probability distribution $\Pr\left(y|\mathbf{w},\phi(.)\right)$, rather than the 'true' distribution $\Pr(y|\mathcal{D})$.

$$E\left(\mathbf{w}\right) = \mathbf{E}_{\Pr(y|\mathcal{D})}\left[-\log\Pr\left(y|\mathbf{w},\phi(.)\right)\right] \tag{1}$$

2. The Cross-entropy Loss function:

$$E\left(\mathbf{w}\right) = -\left[\frac{1}{m}\sum_{i=1}^{m}\left(y^{(i)}\log f_{\mathbf{w}}\left(\mathbf{x}^{(i)}\right) + \left(1-y^{(i)}\right)\log\left(1-f_{\mathbf{w}}\left(\mathbf{x}^{(i)}\right)\right)\right)\right] \tag{2}$$

with some simplification,

$$E\left(\mathbf{w}\right) = -\left[\frac{1}{m}\sum_{i=1}^{m}\left(y^{(i)}\mathbf{w}^{T}\phi(\mathbf{x}^{(i)}) - \log\left(1+exp\left(\mathbf{w}^{T}\mathbf{x}^{(i)}\right)\right)\right)\right] \tag{3}$$

[2] https://en.wikipedia.org/wiki/Cross_entropy

Gradient descent for LR

1. No closed form solution to the cross-entropy loss

$$\widehat{\mathbf{w}}^{MLE} = \arg\min_{\mathbf{w}} -\left[\frac{1}{m}\sum_{i=1}^{m}\left(y^{(i)}\log f_{\mathbf{w}}\left(\mathbf{x}^{(i)}\right) + \left(1-y^{(i)}\right)\log\left(1-f_{\mathbf{w}}\left(\mathbf{x}^{(i)}\right)\right)\right)\right] \quad (4)$$

2. Apply gradient descent with $\mathbf{w}^{(k+1)} = \mathbf{w}^{k} - \eta\nabla E\left(\mathbf{w}^{k}\right)$

$$w^{(k+1)} = w^{(k)} - \eta \nabla\left[-\frac{1}{m}\sum_{i=1}^{m} y^{(i)} w^{(k)T}\phi(x^{(i)}) - \log\left(1+e^{w^{(k)T}\phi(x^{(i)})}\right)\right]$$

$$= w^{(k)} + \frac{\eta}{m}\sum_{i=1}^{m}\underbrace{\left(y^{(i)} - \Pr(Y=1 \mid x^{(i)}; w^{(k)})\right)}_{\text{Smoothed \& averaged version of perceptron update!!}}\phi(x^{(i)})$$

Gradient descent for LR

1. No closed form solution to the cross-entropy loss

$$\widehat{\mathbf{w}}^{MLE} = \arg\min_{\mathbf{w}} - \left[\frac{1}{m}\sum_{i=1}^{m}\left(y^{(i)}\log f_{\mathbf{w}}\left(\mathbf{x}^{(i)}\right) + \left(1-y^{(i)}\right)\log\left(1-f_{\mathbf{w}}\left(\mathbf{x}^{(i)}\right)\right)\right)\right] \quad (4)$$

2. Apply gradient descent with $\mathbf{w}^{(k+1)} = \mathbf{w}^{k} - \eta\nabla E\left(\mathbf{w}^{k}\right)$
3. The descent update

$$-\eta\nabla E\left(\mathbf{w}\right) = -\eta\left[\frac{1}{m}\sum_{i=1}^{m}\left(y^{(i)}\nabla\log f_{\mathbf{w}}\left(\mathbf{x}^{(i)}\right) + \left(1-y^{(i)}\right)\nabla\log\left(1-f_{\mathbf{w}}\left(\mathbf{x}^{(i)}\right)\right)\right)\right] \quad (5)$$

Gradient descent for LR

1. No closed form solution to the cross-entropy loss

$$\widehat{\mathbf{w}}^{MLE} = \arg\min_{\mathbf{w}} -\left[\frac{1}{m}\sum_{i=1}^{m}\left(y^{(i)}\log f_{\mathbf{w}}\left(\mathbf{x}^{(i)}\right) + \left(1-y^{(i)}\right)\log\left(1-f_{\mathbf{w}}\left(\mathbf{x}^{(i)}\right)\right)\right)\right] \quad (4)$$

2. Apply gradient descent with $\mathbf{w}^{(k+1)} = \mathbf{w}^{k} - \eta\nabla E\left(\mathbf{w}^{k}\right)$
3. The descent update

$$-\eta\nabla E\left(\mathbf{w}\right) = -\eta\left[\frac{1}{m}\sum_{i=1}^{m}\left(y^{(i)}\nabla\log f_{\mathbf{w}}\left(\mathbf{x}^{(i)}\right) + \left(1-y^{(i)}\right)\nabla\log\left(1-f_{\mathbf{w}}\left(\mathbf{x}^{(i)}\right)\right)\right)\right] \quad (5)$$

4. $\nabla f_{\mathbf{w}}\left(\mathbf{x}^{(i)}\right) = \phi(\mathbf{x}^{(i)})\left(\frac{e^{-(\mathbf{w})^T\phi(\mathbf{x}^{(i)})}}{1+e^{-(\mathbf{w})^T\phi(\mathbf{x}^{(i)})}}\right)$
   $\Rightarrow$

} A different derivation (more complex)

Gradient descent for LR

1. No closed form solution to the cross-entropy loss

$$\widehat{\mathbf{w}}^{MLE} = \arg\min_{\mathbf{w}} - \left[\frac{1}{m}\sum_{i=1}^{m}\left(y^{(i)}\log f_{\mathbf{w}}\left(\mathbf{x}^{(i)}\right) + \left(1-y^{(i)}\right)\log\left(1-f_{\mathbf{w}}\left(\mathbf{x}^{(i)}\right)\right)\right)\right] \quad (4)$$

2. Apply gradient descent with $\mathbf{w}^{(k+1)} = \mathbf{w}^k - \eta\nabla E\left(\mathbf{w}^k\right)$
3. The descent update

$$-\eta\nabla E\left(\mathbf{w}\right) = -\eta\left[\frac{1}{m}\sum_{i=1}^{m}\left(y^{(i)}\nabla\log f_{\mathbf{w}}\left(\mathbf{x}^{(i)}\right) + \left(1-y^{(i)}\right)\nabla\log\left(1-f_{\mathbf{w}}\left(\mathbf{x}^{(i)}\right)\right)\right)\right] \quad (5)$$

4. $\nabla f_{\mathbf{w}}\left(\mathbf{x}^{(i)}\right) = \phi(\mathbf{x}^{(i)})\left(\frac{e^{-(\mathbf{w})^T\phi(\mathbf{x}^{(i)})}}{1+e^{-(\mathbf{w})^T\phi(\mathbf{x}^{(i)})}}\right)$
$\Rightarrow$
5. $\nabla\log f_{\mathbf{w}}\left(\mathbf{x}^{(i)}\right) = \phi(\mathbf{x}^{(i)})e^{-(\mathbf{w})^T\phi(\mathbf{x}^{(i)})}\left(\frac{1}{1+e^{-(\mathbf{w})^T\phi(\mathbf{x}^{(i)})}}\right)^2$ and
$\nabla\log\left(1-f_{\mathbf{w}}\left(\mathbf{x}^{(i)}\right)\right) = -\phi(\mathbf{x}^{(i)})\left(\frac{1}{1+e^{-(\mathbf{w})^T\phi(\mathbf{x}^{(i)})}}\right)^2$

Descent update for LR

$$-\eta \nabla E(\mathbf{w}) = -\eta \left[\frac{1}{m} \sum_{i=1}^{m} \left(y^{(i)} \nabla \log f_{\mathbf{w}}\left(\mathbf{x}^{(i)}\right) + \left(1 - y^{(i)}\right) \nabla \log \left(1 - f_{\mathbf{w}}\left(\mathbf{x}^{(i)}\right)\right)\right)\right] \tag{6}$$

1. $\nabla \log f_{\mathbf{w}}\left(\mathbf{x}^{(i)}\right) = \phi(\mathbf{x}^{(i)}) e^{-(\mathbf{w})^T \phi(\mathbf{x}^{(i)})} \left(\frac{1}{1+e^{-(\mathbf{w})^T \phi(\mathbf{x}^{(i)})}}\right)^2$ and
   $\nabla \log\left(1 - f_{\mathbf{w}}\left(\mathbf{x}^{(i)}\right)\right) = -\phi(\mathbf{x}^{(i)}) \left(\frac{1}{1+e^{-(\mathbf{w})^T \phi(\mathbf{x}^{(i)})}}\right)^2$
2. $\Rightarrow$ The final descent update is

Descent update for LR

$$-\eta\nabla E(\mathbf{w}) = -\eta\left[\frac{1}{m}\sum_{i=1}^{m}\left(y^{(i)}\nabla \log f_{\mathbf{w}}\left(\mathbf{x}^{(i)}\right) + \left(1 - y^{(i)}\right)\nabla \log\left(1 - f_{\mathbf{w}}\left(\mathbf{x}^{(i)}\right)\right)\right)\right] \tag{6}$$

1. $\nabla \log f_{\mathbf{w}}(\mathbf{x}^{(i)}) = \phi(\mathbf{x}^{(i)})e^{-(\mathbf{w})^T\phi(\mathbf{x}^{(i)})}\left(\frac{1}{1+e^{-(\mathbf{w})^T\phi(\mathbf{x}^{(i)})}}\right)^2$ and
   $\nabla \log(1 - f_{\mathbf{w}}(\mathbf{x}^{(i)})) = -\phi(\mathbf{x}^{(i)})\left(\frac{1}{1+e^{-(\mathbf{w})^T\phi(\mathbf{x}^{(i)})}}\right)^2$
2. $\Rightarrow$ The final descent update is

$$-\eta\nabla E(\mathbf{w}) = \eta\left[\frac{1}{m}\sum_{i=1}^{m}\left(y^{(i)} - f_{\mathbf{w}}\left(\mathbf{x}^{(i)}\right)\right)\phi(\mathbf{x}^{(i)})\right] \tag{7}$$

Gradient descent for LR

1. The final descent update

$$-\eta \nabla E(\mathbf{w}) = \eta \left[\frac{1}{m} \sum_{i=1}^{m} \left(y^{(i)} - f_{\mathbf{w}}\left(\mathbf{x}^{(i)}\right) \right) \phi(\mathbf{x}^{(i)}) \right] \tag{8}$$

2. The iterative update rule:

$$\mathbf{w}^{(k+1)} = \mathbf{w}^{k} + \eta \left[\frac{1}{m} \sum_{i=1}^{m} \left(y^{(i)} - f_{\mathbf{w}^k}\left(\mathbf{x}^{(i)}\right) \right) \phi(\mathbf{x}^{(i)}) \right] \tag{9}$$

3. Stochastic version of the same:

$$\mathbf{w}^{(k+1)} = \mathbf{w}^{k} + \eta \left(y^{(i)} - f_{\mathbf{w}^k}\left(\mathbf{x}^{(i)}\right) \right) \phi(\mathbf{x}^{(i)}) \tag{10}$$

4. How would you contrast the updates with sigmoid (LR) against those with the step function (perceptron)?

H/W
contrast!