CS 4824/ECE 4424: Gradient-based Optimization

Acknowledgement:
Many of these slides are derived from Tom Mitchell, Pascal Poupart, Pieter Abbeel, Eric Eaton, Carlos Guestrin, William Cohen, and Andrew Moore.

Training logistic regression: MCLE

- We need to choose W = <$w_0,\ldots,w_n$> to
  maximize the conditional likelihood of training data

  - where $P(Y=0\,|\,X,W) = \dfrac{1}{1+exp(w_0+\sum_{i=1}^{n} w_i x_i)}$

  - and $P(Y=1\,|\,X,W) = \dfrac{exp(w_0+\sum_{i=1}^{n} w_i x_i)}{1+exp(w_0+\sum_{i=1}^{n} w_i x_i)}$

- Training data D = {<X^1, Y^1>,…, <X^L, Y^L>}
- Data likelihood is $\prod_l \boxed{P(<X^l, Y^l>\,|\,W)}$
- Data conditional likelihood is $\prod_l \boxed{P(Y^l\,|\,X^l, W)}$
- Therefore we need to estimate $W_{MCLE} = \arg\max_W \boxed{\prod_l P(Y^l\,|\,X^l, W)}$

Expressing conditional log likelihood

- $l(W) = \ln \prod_l P(Y^l | X^l, W) = \sum_l \ln P(Y^l | X^l, W)$
  - where $P(Y = 0 | X, W) = \dfrac{1}{1 + exp(w_0 + \sum_{i=1}^n w_i x_i)}$
  - and $P(Y = 1 | X, W) = \dfrac{exp(w_0 + \sum_{i=1}^n w_i x_i)}{1 + exp(w_0 + \sum_{i=1}^n w_i x_i)}$

- $$l(W) = \sum_l Y^l \ln P(Y^l = 1 | X^l, W) + (1 - Y^l) \ln P(Y^l = 0 | X^l, W)$$
$$= \sum_l Y^l \ln \frac{P(Y^l = 1 | X^l, W)}{P(Y^l = 0 | X^l, W)} + \ln P(Y^l = 0 | X^l, W)$$
$$= \boxed{\sum_l Y^l (w_0 + \sum_i^n w_i X_i^l) - \ln(1 + exp(w_0 + \sum_i^n w_i X_i^l))}$$

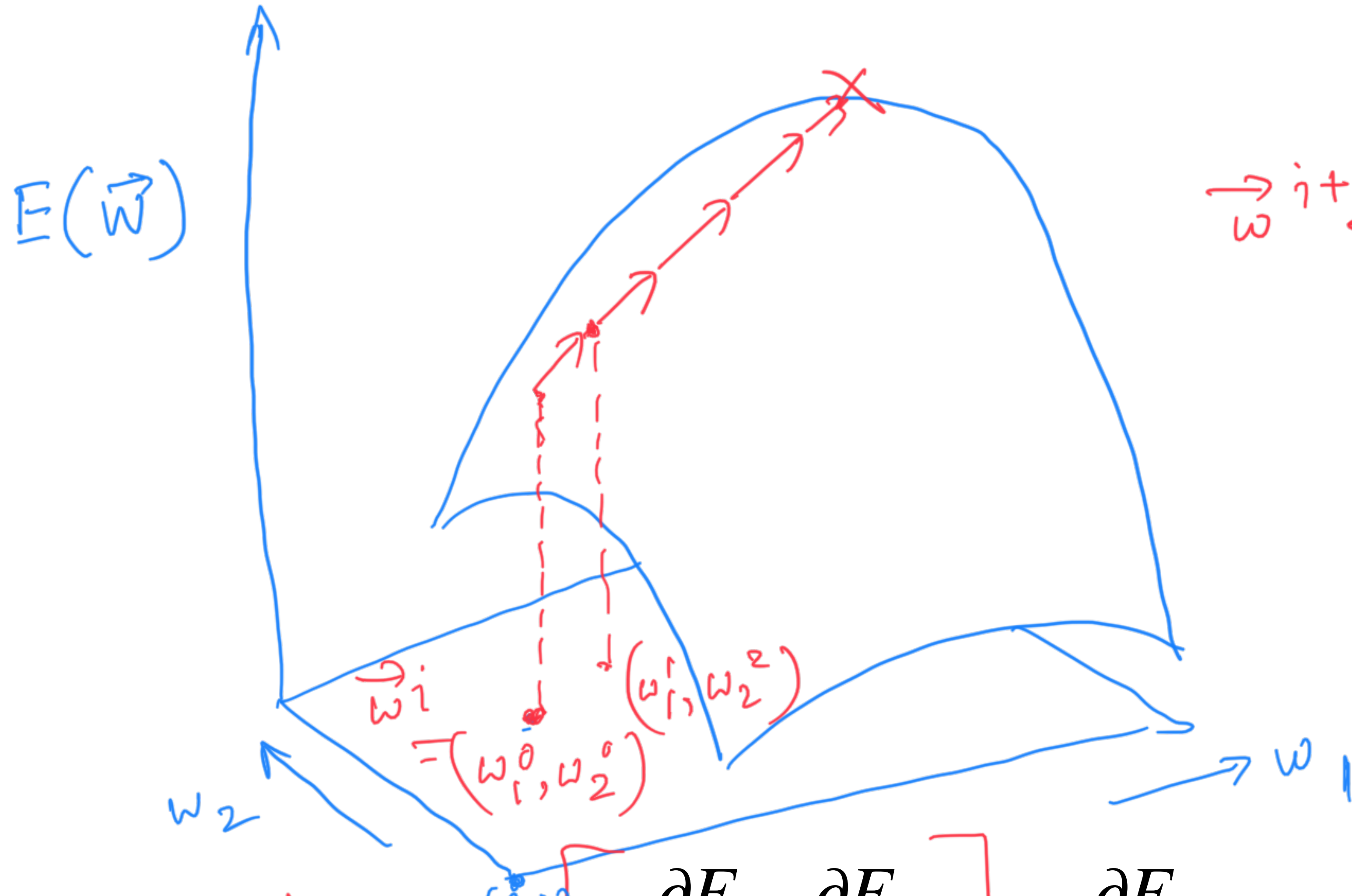
Maximizing conditional log likelihood

- $$l(W) = \ln \prod_l P(Y^l | X^l, W)$$

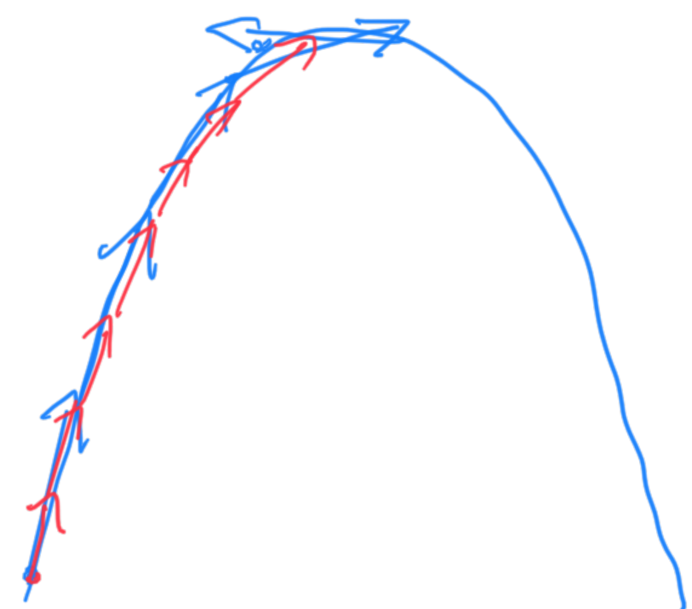
$$= \sum_l Y^l (w_0 + \sum_i^n w_i X_i^l) - \ln(1 + exp(w_0 + \sum_i^n w_i X_i^l))$$

- **Good news**: *l(W)* is a concave function of *W*
- **Bad news**: no closed-form solution to maximize *l(W)*

Gradient ascent



$\vec{w}^{i+1} \leftarrow \vec{w}^i + \nabla E(\vec{w})$



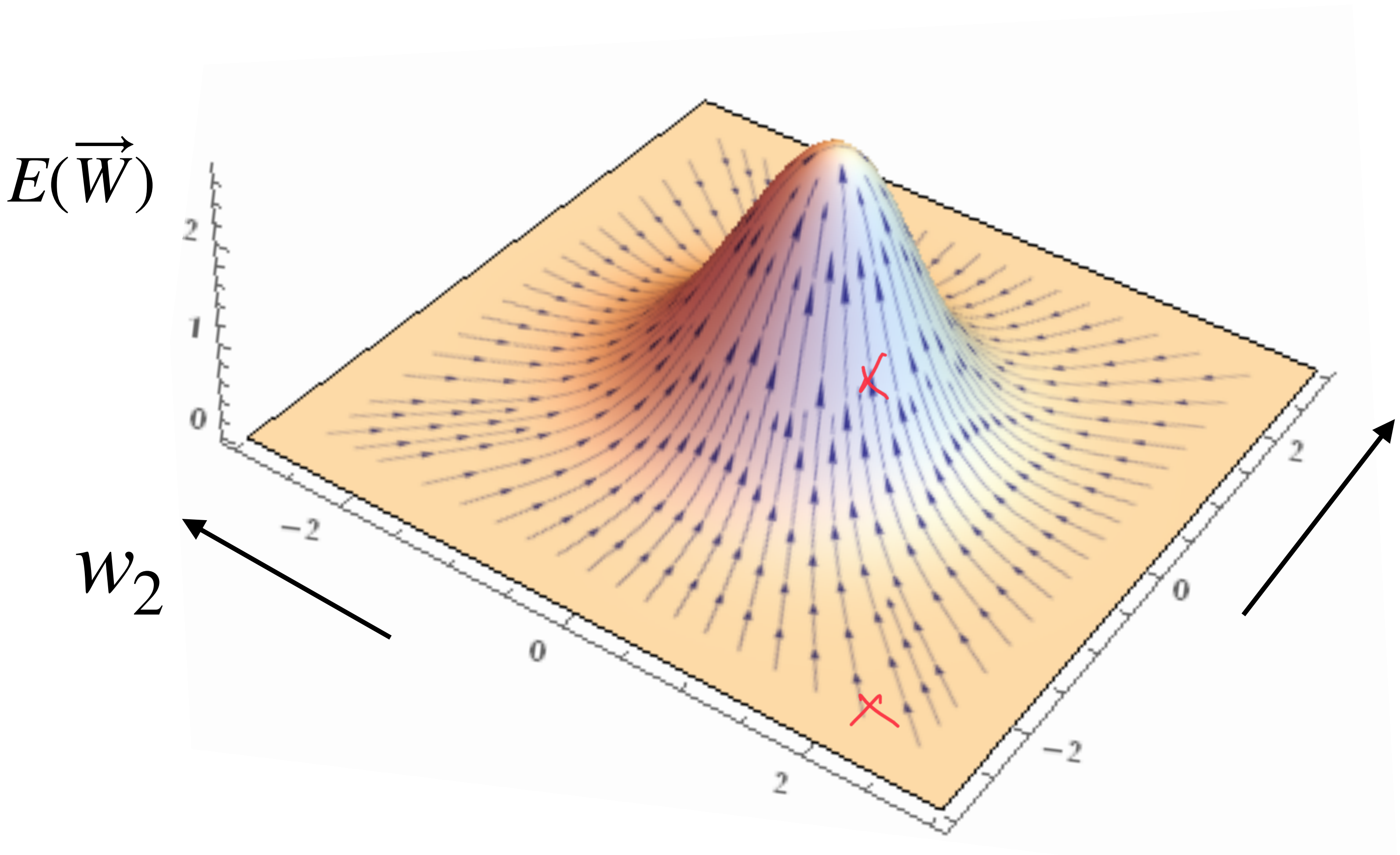
- **Gradient** $\nabla E(\vec{w}) = [\frac{\partial E}{\partial w_0}, \frac{\partial E}{\partial w_1}, \ldots, \frac{\partial E}{\partial w_n}]$

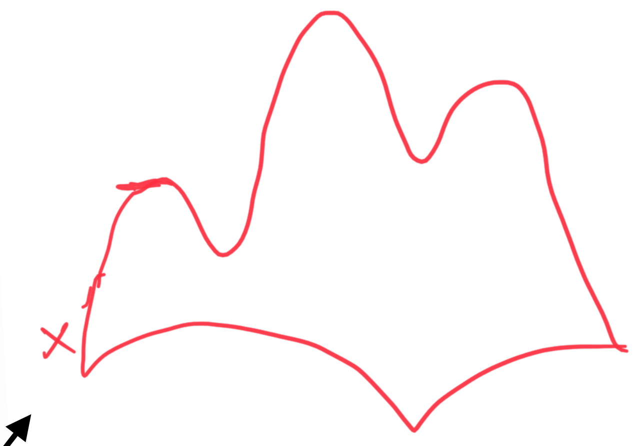
- Training Rule: $\vec{w}^{(i+1)} \leftarrow \vec{w}^i + \eta \nabla E(\vec{w})$

- i.e. $\boxed{\Delta w_i = \eta \frac{\partial E}{\partial w_i}}$

step size (aka. learnig rate)

Gradient ascent





- **Gradient** $\nabla E(\vec{w}) = [\frac{\partial E}{\partial w_0}, \frac{\partial E}{\partial w_1}, \ldots, \frac{\partial E}{\partial w_n}]$

- Training Rule: $\vec{w}^{(i+1)} \leftarrow \vec{w}^{i} + \eta \nabla E(\vec{w})$

- i.e. $\Delta w_i = \eta \frac{\partial E}{\partial w_i}$

Maximizing conditional log likelihood via gradient ascent

- $$l(W) = \ln \prod_l P(Y^l \mid X^l, W)$$

$$= \boxed{\sum_l Y^l (w_0 + \sum_i^n w_i X_i^l) - \ln(1 + exp(w_0 + \sum_i^n w_i X_i^l))}$$

- $$\frac{\partial l(W)}{\partial w_i} = \boxed{\sum_l X_i^l (Y^l - \hat{P}(Y^l = 1 \mid X^l, W))}$$

- **Gradient ascent algorithm**: iterate until change < ε
  - $\forall\, i$ repeat $\quad w_i \leftarrow w_i + \eta \boxed{\sum_l X_i^l (Y^l - \hat{P}(Y^l = 1 \mid X^l, W))}$
  - assume $\quad X_0 = 1$ for w_0

step size (a.k.a. learning rate)

L.R
Learning
algorithm

Demo Time 🙂

https://yihui.org/animation/example/grad-desc/

https://blog.skz.dev/gradient-descent

Batch vs. Stochastic gradient

- Batch gradient: use $\nabla E_D(\vec{w})$ over the entire training set D
  - Do until satisfied:
    - 1. Compute the gradient: $\nabla E_D(\vec{w}) = [\frac{\partial E_D}{\partial w_0}, \frac{\partial E_D}{\partial w_1}, \ldots, \frac{\partial E_D}{\partial w_n}]$
    - 2. Update the vector of parameters: $\vec{w}^{(i+1)} \leftarrow \vec{w}^i + \eta \nabla E_D(\vec{w})$
- Stochastic gradient: use $\nabla E_d(\vec{w})$ over a single example $d \in D$
  - Do until satisfied:
    - 1. Choose (with replacement) a random training example $d \in D$
    - 2. Compute the gradient just for d: $\nabla E_d(\vec{w}) = [\frac{\partial E_d}{\partial w_0}, \frac{\partial E_d}{\partial w_1}, \ldots, \frac{\partial E_d}{\partial w_n}]$
    - 2. Update the vector of parameters: $\vec{w}^{(i+1)} \leftarrow \vec{w}^i + \eta \nabla E_d(\vec{w})$
- Stochastic approximates Batch arbitrarily closely as $\eta \to 0$
- Stochastic is much faster than Batch when D is very large
- An intermediate approach is to use a subset of D instead of just one single example d

Hyperparameters in gradient-based optimization

- Epoch:
  - An epoch refers to a full pass over the dataset
  - One epoch means that each sample in the training dataset has had an opportunity to update the internal model parameters
  - The number of epochs is the number of complete passes through the training dataset
  - The number of epochs can be set to an integer value between one and infinity
  - You can run the algorithm for as long as you like and even stop it using other criteria besides a fixed number of epochs.

- Batch size:
  - Batch size is a number of samples processed before the model is updated
  - An epoch is comprised of one or more batches
  - The size of a batch must be more than or equal to one and less than or equal to the number of samples in the training dataset

- There are no magic rules for how to configure these hyperparameters. You may try different values and see what works best for your problem.

We looked at M(C)LE, but what about MAP?

- One common approach is to define priors on W
  - Normal distribution, zero mean, identity covariance

- Helps avoid very large weights and overfitting

- Therefore we can estimate

$$W_{MAP} = \arg\max_{W} ln[P(W) \prod_{l} P(Y^l | X^l, W)]$$

- Let's assume Gaussian prior: W ~ $\mathcal{N}(0, \sigma I)$

zero-mean
Gaussian prior

M(C)LE vs MAP

- Maximum (conditional) likelihood estimate
  - $$W_{MCLE} = \arg\max_W \ln \prod_l P(Y^l \mid X^l, W)$$

$$w_i \leftarrow w_i + \eta \sum_l X_i^l (Y^l - \hat{P}(Y^l = 1 \mid X^l, W))$$

- Maximum a posteriori estimate with prior $W \sim \mathcal{N}(0, \sigma I)$
- $$W_{MAP} = \arg\max_W \ln[P(W) \prod_l P(Y^l \mid X^l, W)]$$

$x \in \{0,1\}$

$\eta = 0.001$ $\lambda = 0.0005$

$$w_i \leftarrow w_i \boxed{- \eta \lambda w_i} + \eta \sum_l X_i^l (Y^l - \hat{P}(Y^l = 1 \mid X^l, W))$$

MAP estimates and regularization

- Maximum a posteriori estimate with prior $W \sim \mathcal{N}(0, \sigma I)$
- $$W_{MAP} = \arg\max_W \ln[P(W) \prod_l P(Y^l | X^l, W)]$$

$$w_i \leftarrow w_i - \boxed{\eta \lambda w_i} + \eta \sum_l X_i^l (Y^l - \hat{P}(Y^l = 1 | X^l, W))$$

- Called a "**regularization**" term
- Helps reduce overfitting, especially for sparse data situations
- Keeps weights near zero with prior $W \sim \mathcal{N}(0, \sigma I)$, or whatever the prior suggests
- Used very frequently in logistic regression

The bottom line

- Consider learning f: $X \rightarrow Y$
  - X is a vector of real-valued features <X_1, X_2, …,X_n>
  - Y is boolean
  - Assume all X_i's are conditionally independent given Y
  - Model $P(X_i | Y = y_k)$ as Gaussian ~ $\mathcal{N}(\mu_{ik}, \sigma_i)$
  - Model P(Y) as Bernoulli (π)

- Given that, we can derive the parametric form of $P(Y|X)$:
    - where $P(Y = 0 | X, W) = \dfrac{1}{1 + exp(w_0 + \sum_{i=1}^{n} w_i x_i)}$
    - and $P(Y = 1 | X, W) = \dfrac{exp(w_0 + \sum_{i=1}^{n} w_i x_i)}{1 + exp(w_0 + \sum_{i=1}^{n} w_i x_i)}$
- And we can estimate W directly from the training data

So far…

- Training classifiers involve estimating $\boldsymbol{f}$: $\boldsymbol{X} \rightarrow \boldsymbol{Y}$ or $P(Y|X)$

- Naïve Bayes
  - Assumes some functional form for $P(X|Y)$, $P(Y)$
  - Estimates parameters of $P(X|Y)$, $P(Y)$ from training data
  - Use Bayes rule to calculate $P(Y|X)$

- Logistic Regression
  - Assumes some functional form for $P(Y|X)$
  - Estimates parameters of $P(Y|X)$ directly from training data

Use Naïve Bayes or Logistic Regression?