Announcements

HW 2 will be posted tonight or tomorrow. **DUE 11/2**

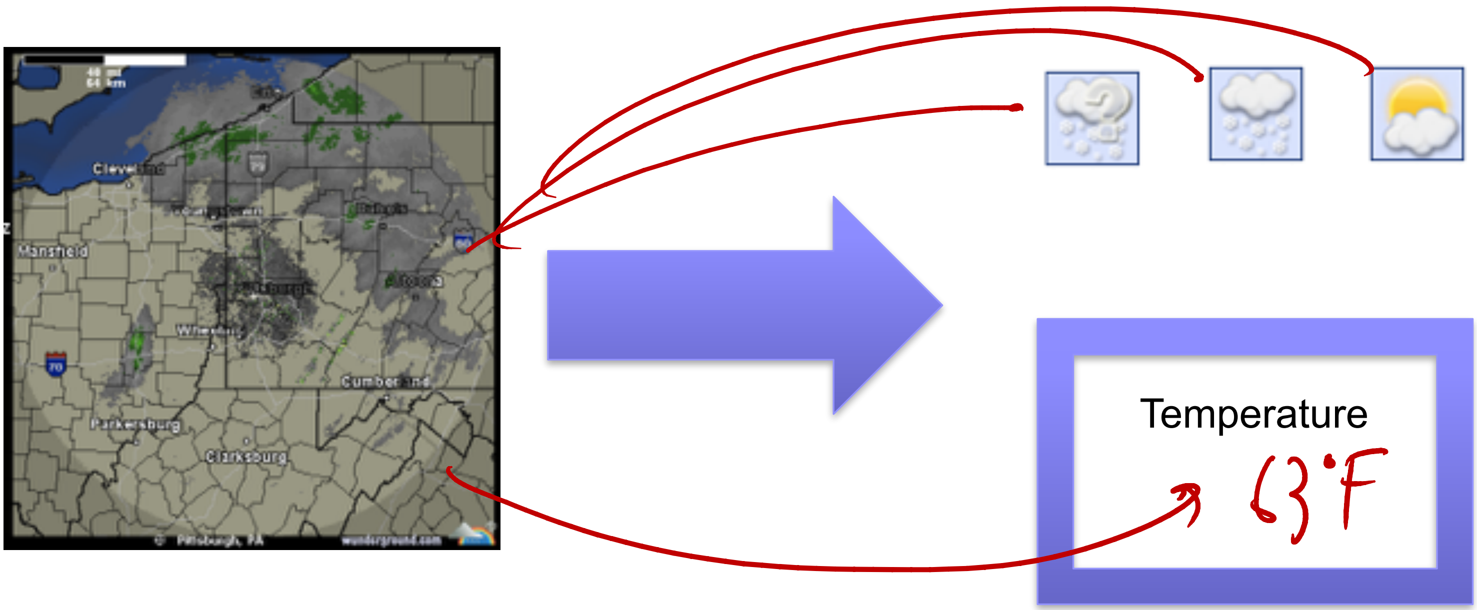
Classification
Logistic Regression

Machine Learning – CSE546
Kevin Jamieson
University of Washington

October 16, 2016

THUS FAR, REGRESSION: PREDICT A CONTINUOUS VALUE GIVEN SOME INPUTS

Weather prediction revisted



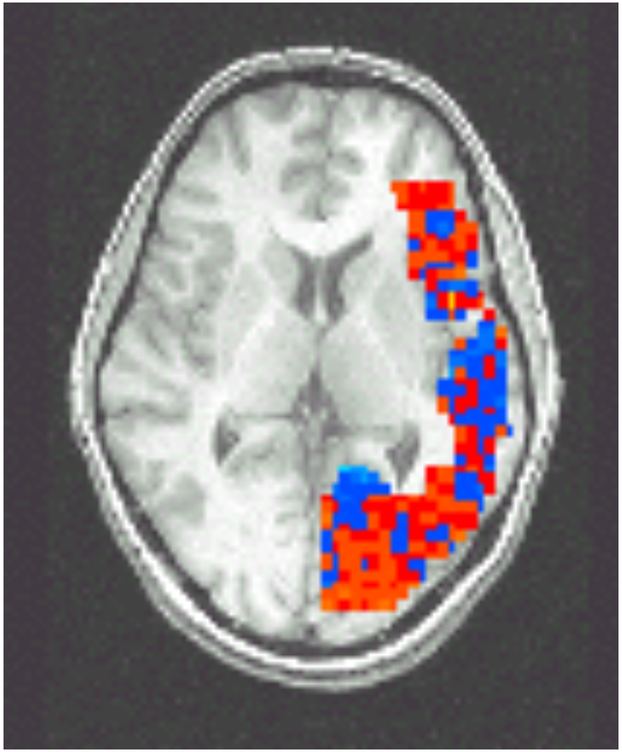
Reading Your Brain, Simple Example

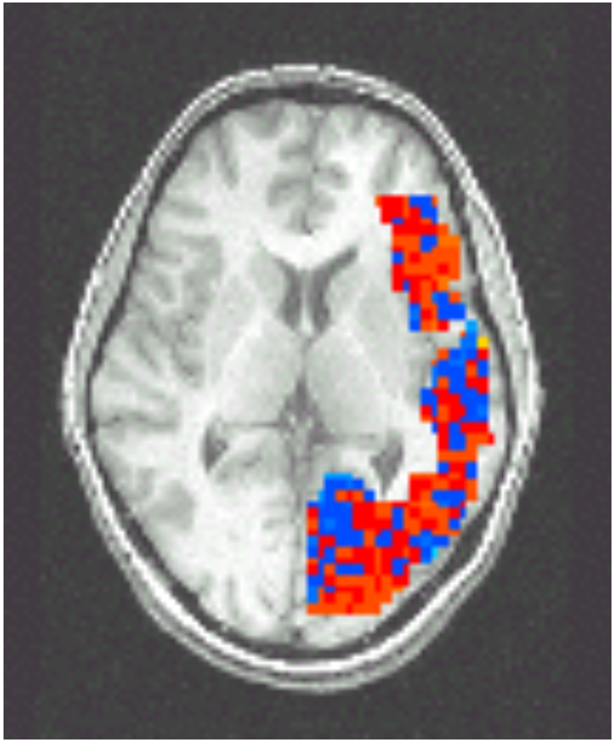
[Mitchell et al.]

Pairwise classification accuracy: 85%

Person

Animal





Classification

- **Learn**: f:**X** —>Y
  - **X** – features
  - Y – target classes

- Conditional probability: P(Y|**X**)

- Suppose you know P(Y|**X**) exactly, how should you classify?
  - Bayes optimal classifier:

- **How do we estimate P(Y|X)?**

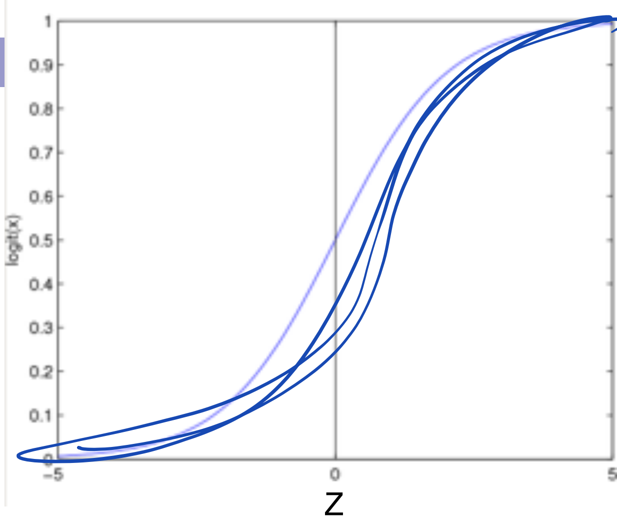
Link Functions

- Estimating P(Y|**X**): Why not use standard linear regression?

- Combining regression and probability?
  - Need a mapping from real values to [0,1]
  - A link function!

Logistic Regression

Logistic function (or Sigmoid): $\frac{1}{1+exp(-z)}$



- Learn P(Y|**X**) directly
  - Assume a particular functional form for link function
  - Sigmoid applied to a linear function of the input features:

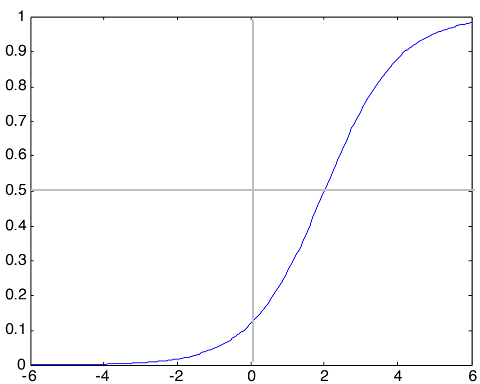
$$P(Y=0|X,W) = \frac{1}{1+exp(w_0+\sum_i w_i X_i)}$$

Features can be discrete or continuous!

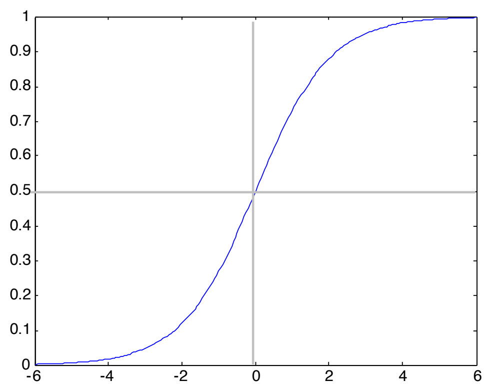
Understanding the sigmoid

$$g(w_0 + \sum_i w_i x_i) = \frac{1}{1 + e^{w_0 + \sum_i w_i x_i}}$$

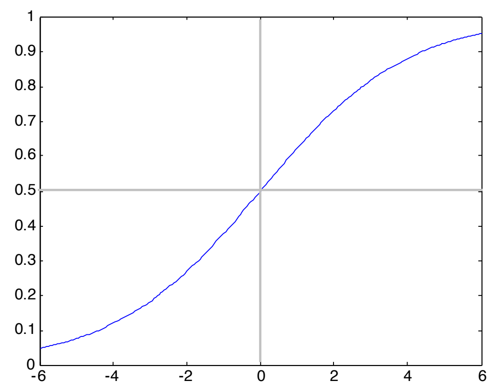
w_0=-2, w_1=-1



w_0=0, w_1=-1



w_0=0, w_1=-0.5



Very convenient!

$$P(Y=0|X=<X_1,...X_n>) = \frac{1}{1+exp(w_0+\sum_i w_i X_i)}$$

implies

$$P(Y=1|X=<X_1,...X_n>) = \frac{exp(w_0+\sum_i w_i X_i)}{1+exp(w_0+\sum_i w_i X_i)}$$

Very convenient!

$$P(Y=0|X=<X_1,...X_n>) = \frac{1}{1+exp(w_0+\sum_i w_i X_i)}$$

implies

$$P(Y=1|X=<X_1,...X_n>) = \frac{exp(w_0+\sum_i w_i X_i)}{1+exp(w_0+\sum_i w_i X_i)}$$

implies

$$\frac{P(Y=1|X)}{P(Y=0|X)} = exp(w_0+\sum_i w_i X_i)$$

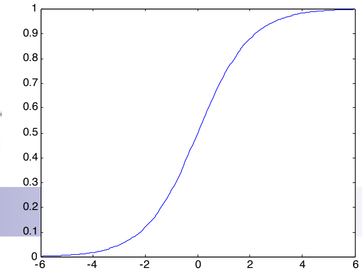
implies

$$\ln\frac{P(Y=1|X)}{P(Y=0|X)} = w_0+\sum_i w_i X_i$$

linear classification rule!

Logistic Regression – a Linear classifier

$$\frac{1}{1+exp(-z)}$$



$$g(w_0 + \sum_i w_i x_i) = \frac{1}{1+e^{w_0+\sum_i w_i x_i}}$$

$$\ln \frac{P(Y=0|X)}{P(Y=1|X)} = w_0 + \sum_i w_i X_i$$

Loss function: Conditional Likelihood

- Have a bunch of iid data of the form: $\{(x_i, y_i)\}_{i=1}^n \quad x_i \in \mathbb{R}^d, \quad y_i \in \{-1, 1\}$

$$P(Y=-1|x,w) = \frac{1}{1+\exp(w^T x)}$$

$$P(Y=1|x,w) = \frac{\exp(w^T x)}{1+\exp(w^T x)}$$

- This is equivalent to:

$$P(Y=y|x,w) = \frac{1}{1+\exp(-y\, w^T x)}$$

- So we can compute the maximum likelihood estimator:

$$\widehat{w}_{MLE} = \arg\max_w \prod_{i=1}^n P(y_i|x_i, w)$$

Loss function: Conditional Likelihood

- Have a bunch of iid data of the form: $\{(x_i, y_i)\}_{i=1}^n \quad x_i \in \mathbb{R}^d, \quad y_i \in \{-1, 1\}$

$$\widehat{w}_{MLE} = \arg\max_w \prod_{i=1}^n P(y_i|x_i, w) \qquad P(Y = y|x, w) = \frac{1}{1 + \exp(-y\, w^T x)}$$

Loss function: Conditional Likelihood

- Have a bunch of iid data of the form: $\{(x_i, y_i)\}_{i=1}^n \qquad x_i \in \mathbb{R}^d, \quad y_i \in \{-1, 1\}$

$$\widehat{w}_{MLE} = \arg\max_w \prod_{i=1}^n P(y_i|x_i, w) \qquad P(Y = y|x, w) = \frac{1}{1 + \exp(-y\, w^T x)}$$

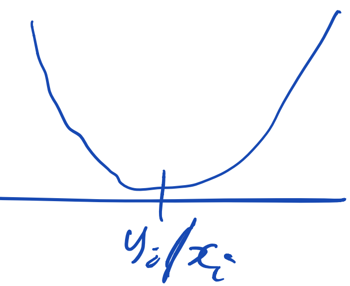
$$= \arg\min_w \sum_{i=1}^n \log(1 + \exp(-y_i\, x_i^T w))$$

Loss function: Conditional Likelihood

- Have a bunch of iid data of the form: $\{(x_i, y_i)\}_{i=1}^n \quad x_i \in \mathbb{R}^d, \quad y_i \in \{-1, 1\}$

$$\widehat{w}_{MLE} = \arg\max_w \prod_{i=1}^n P(y_i|x_i, w) \qquad P(Y = y|x, w) = \frac{1}{1 + \exp(-y\, w^T x)}$$

$$= \arg\min_w \sum_{i=1}^n \log(1 + \exp(-y_i\, x_i^T w))$$



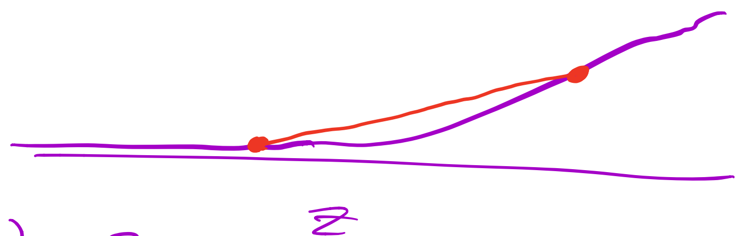
Logistic Loss: $\ell_i(w) = \log(1 + \exp(-y_i\, x_i^T w))$

Squared error Loss: $\ell_i(w) = (y_i - x_i^T w)^2$ (MLE for Gaussian noise)

$\log(1+\exp(z))$

For large z

$\log(1+\exp(z)) \simeq \log(\exp(z)) = z$



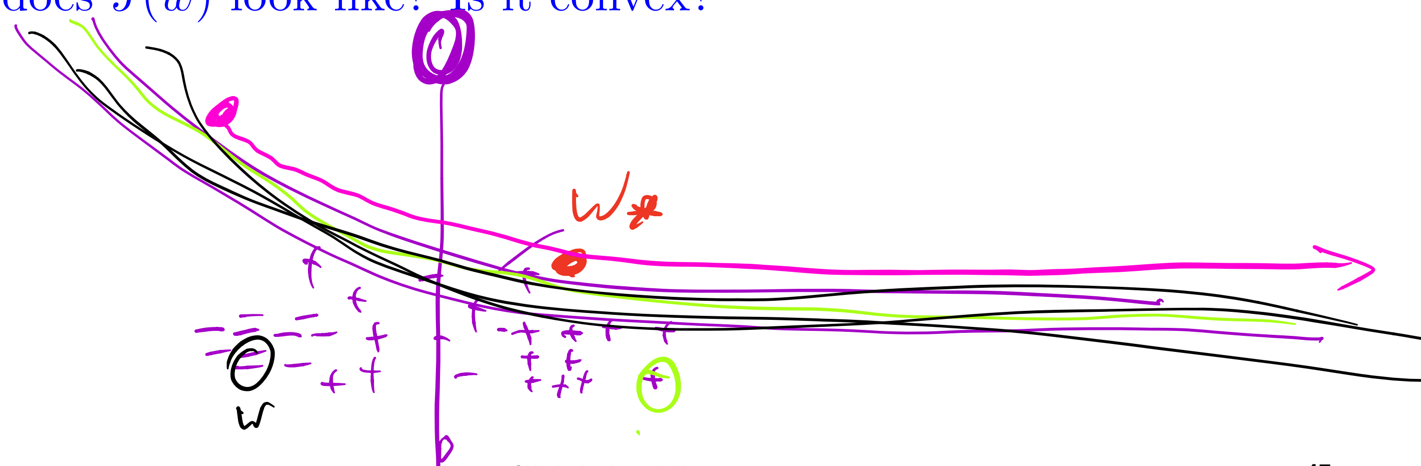
Loss function: Conditional Likelihood

- Have a bunch of iid data of the form: $\{(x_i, y_i)\}_{i=1}^n \quad x_i \in \mathbb{R}^d, \quad y_i \in \{-1, 1\}$

$$\widehat{w}_{MLE} = \arg\max_w \prod_{i=1}^n P(y_i | x_i, w) \qquad P(Y = y | x, w) = \frac{1}{1 + \exp(-y\, w^T x)}$$

$$= \arg\min_w \sum_{i=1}^n \log(1 + \exp(-y_i\, x_i^T w)) = J(w)$$

What does $J(w)$ look like? Is it convex?



Loss function: Conditional Likelihood

- Have a bunch of iid data of the form: $\{(x_i, y_i)\}_{i=1}^n \qquad x_i \in \mathbb{R}^d, \quad y_i \in \{-1, 1\}$

$$\widehat{w}_{MLE} = \arg\max_w \prod_{i=1}^n P(y_i|x_i, w) \qquad P(Y = y|x, w) = \frac{1}{1 + \exp(-y\, w^T x)}$$

$$= \arg\min_w \sum_{i=1}^n \log(1 + \exp(-y_i\, x_i^T w)) = J(w)$$

Good news: *J*(**w**) is convex function of **w**, no local optima problems

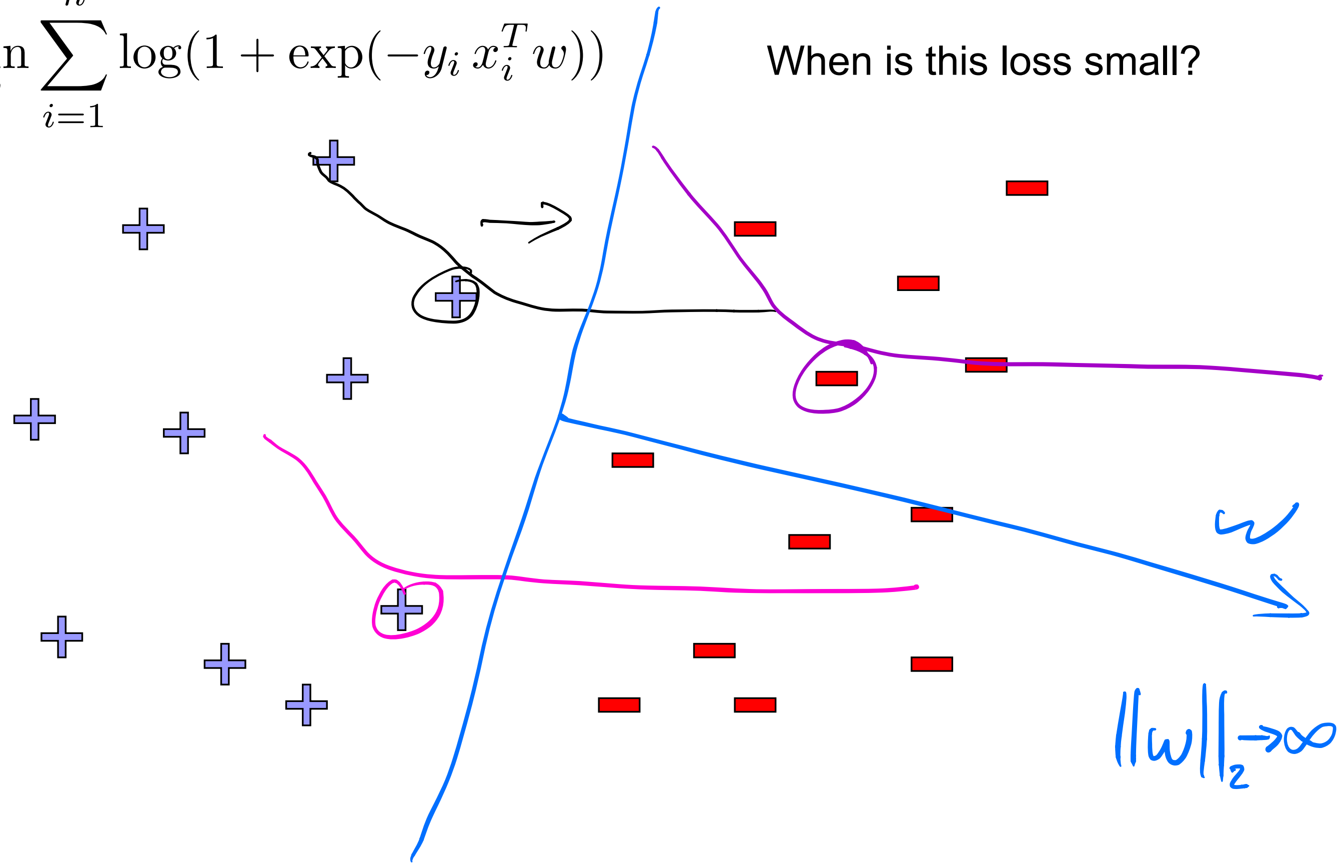
Bad news: no closed-form solution to maximize *J*(**w**)

Good news: convex functions easy to optimize ~~(next time)~~

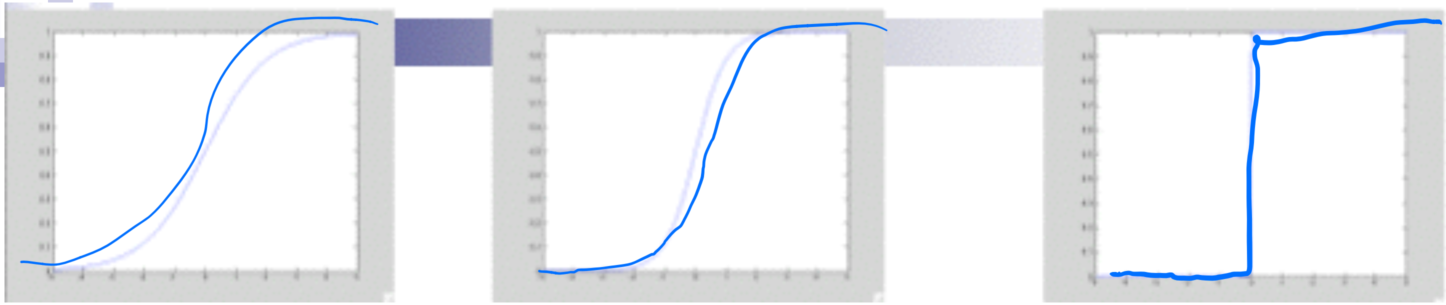
Linear Separability

$$\arg\min_{w} \sum_{i=1}^{n} \log(1 + \exp(-y_i \, x_i^T w))$$

When is this loss small?



Large parameters → Overfitting



$$\frac{1}{1+e^{-x}} \qquad \frac{1}{1+e^{-2x}} \qquad \frac{1}{1+e^{-100x}}$$

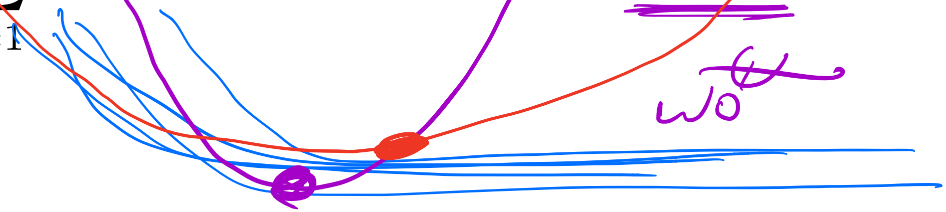
- If data is linearly separable, weights go to infinity
  - In general, leads to overfitting:
- Penalizing high weights can prevent overfitting…

Regularized Conditional Log Likelihood

- Add regularization penalty, e.g., L_2:

$$\arg\min_w \sum_{i=1}^n \log(1+\exp(-y_i\, x_i^T w)) + \lambda ||w||_2^2$$

$\lambda > 0$



- Practical note about w_0:

w_0 should not be regularized

$$\arg\min_{w, w_0} \sum_{i=1}^n \log(1+\exp(-y_i(x_i^T w + w_0)))$$

Gradient Descent

Machine Learning – CSE546
Kevin Jamieson
University of Washington

October 16, 2016

Machine Learning Problems

LS $\ell_i(w) = (y_i - x_i^T w)^2$

- Have a bunch of iid data of the form:

$$\{(x_i, y_i)\}_{i=1}^n \qquad x_i \in \mathbb{R}^d \qquad y_i \in \mathbb{R}$$

- Learning a model's parameters:

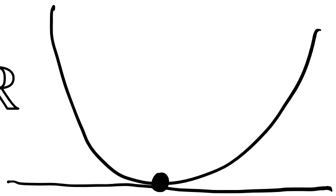
  Each $\ell_i(w)$ is convex.

$$\sum_{i=1}^n \ell_i(w)$$

Machine Learning Problems

- Have a bunch of iid data of the form:

$$\{(x_i, y_i)\}_{i=1}^n \qquad x_i \in \mathbb{R}^d \qquad y_i \in \mathbb{R}$$

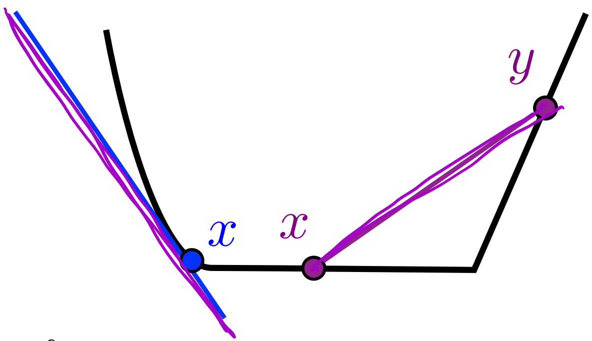


- Learning a model's parameters:

$$\sum_{i=1}^n \ell_i(w)$$

Each $\ell_i(w)$ is convex.

f(x) is a minimum of f if g is a sub-gradient at x



g is a subgradient at x if

$$f(y) \geq f(x) + g^T(y - x)$$

f convex:

$$f(\lambda x + (1-\lambda)y) \leq \lambda f(x) + (1-\lambda) f(y) \qquad \forall x, y, \lambda \in [0,1]$$

$$f(y) \geq f(x) + \nabla f(x)^T (y - x) \qquad \forall x, y$$

Machine Learning Problems

- Have a bunch of iid data of the form:

$$\{(x_i, y_i)\}_{i=1}^n \qquad x_i \in \mathbb{R}^d \qquad y_i \in \mathbb{R}$$

- Learning a model's parameters:

$$\sum_{i=1}^n \ell_i(w)$$

Each $\ell_i(w)$ is convex.

Logistic Loss: $\ell_i(w) = \log(1 + \exp(-y_i\, x_i^T w))$

Squared error Loss: $\ell_i(w) = (y_i - x_i^T w)^2$

Least squares

- Have a bunch of iid data of the form:

$$\{(x_i, y_i)\}_{i=1}^n \qquad x_i \in \mathbb{R}^d \qquad y_i \in \mathbb{R}$$

- Learning a model's parameters: $\sum_{i=1}^n \ell_i(w)$

  Each $\ell_i(w)$ is convex.

Squared error Loss: $\ell_i(w) = (y_i - x_i^T w)^2$

How does software solve: $\frac{1}{2}||\mathbf{X}w - \mathbf{y}||_2^2$

Least squares

- Have a bunch of iid data of the form:

$$\{(x_i, y_i)\}_{i=1}^n \qquad x_i \in \mathbb{R}^d \qquad y_i \in \mathbb{R}$$

- Learning a model's parameters: $\sum_{i=1}^n \ell_i(w)$

Each $\ell_i(w)$ is convex.

Squared error Loss: $\ell_i(w) = (y_i - x_i^T w)^2$

How does software solve: $\frac{1}{2}||\mathrm{X}w - \mathrm{y}||_2^2$

…its complicated:
(LAPACK, BLAS, MKL…)

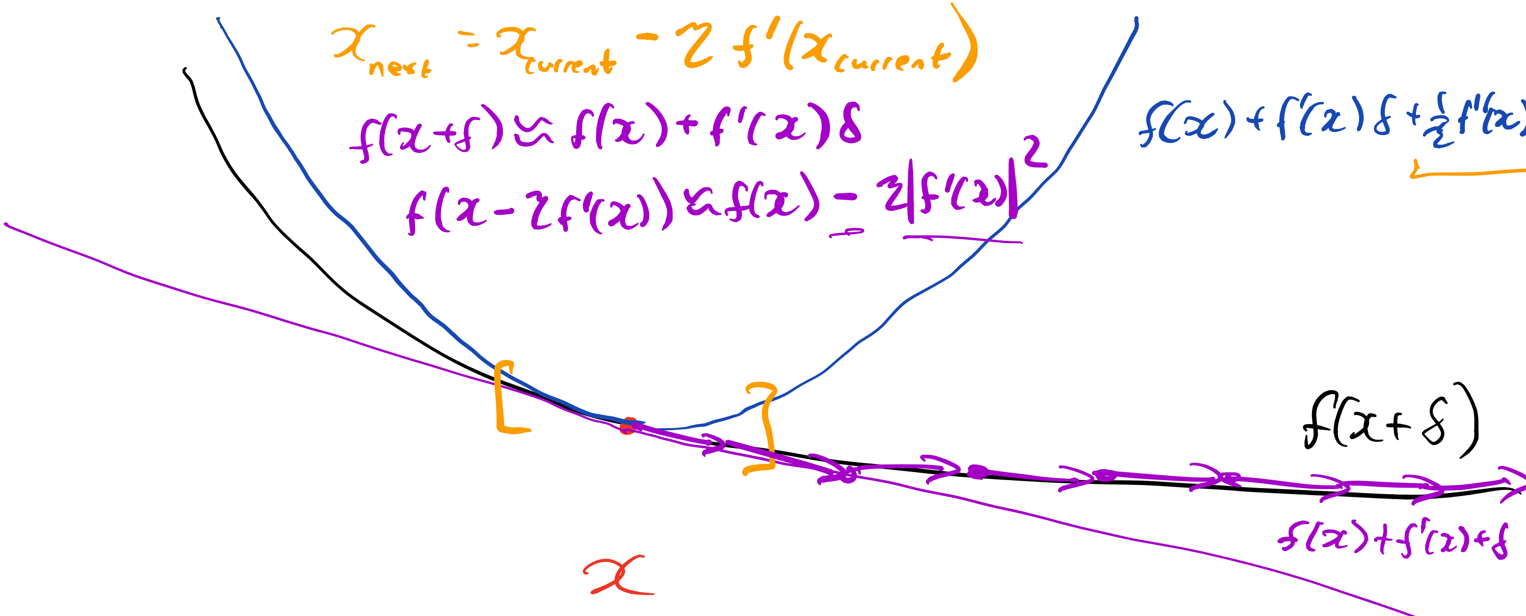
Do you need high precision?
Is X column/row sparse?
Is $\widehat{w}_{LS}$ sparse?
Is $\mathrm{X}^T\mathrm{X}$ "well-conditioned"?
Can $\mathrm{X}^T\mathrm{X}$ fit in cache/memory?

Taylor Series Approximation

- Taylor series in one dimension:

$$f(x+\delta) = f(x) + f'(x)\delta + \frac{1}{2}f''(x)\delta^2 + \dots$$

- Gradient descent:



Taylor Series Approximation

PSD

f is convex $\iff \nabla^2 f(x) \succeq 0 \;\; \forall x$

- Taylor series in d dimensions:

$$f(x+v) = f(x) + \nabla f(x)^T v + \frac{1}{2} v^T \nabla^2 f(x) v + \dots$$

Hessian of f at x

$\nabla^2 f(x) \in \mathbb{R}^{d \times d}$

- Gradient descent:

$$f(x+v) \simeq f(x) + \nabla f(x)^T v$$

$$f(x - \eta \nabla f(x)) \simeq f(x) - \eta \|\nabla f(x)\|_2^2$$

$f(x) + \nabla f(x)^T v$

Gradient Descent $f(w) = \frac{1}{2}||Xw - y||_2^2$

$$w_{t+1} = w_t - \eta \nabla f(w_t)$$

$$\nabla f(w) = X^T(Xw - y)$$

$$w_* = \min_w f(w) = (X^TX)^{-1}X^Ty$$

$$(X^TX)w_* = X^Ty$$

$$w_{t+1} = w_t - \eta(X^TXw_t - X^Ty)$$

$$(w_{t+1} - w_*) = (w_t - w_*) - \eta(X^TXw_t - X^Ty)$$

$$= (w_t - w_*) - \eta(X^TXw_t - X^TXw_*)$$

$$= (w_t - w_*) - \eta(X^TX)(w_t - w_*)$$

$$= (I - \eta(X^TX))(w_t - w_*)$$

$$= (I - \eta(X^TX))^{t+1}(w_0 - w_*)$$

$$A = V \operatorname{diag}(\alpha) V^T$$

$$A^2 = V \operatorname{diag} V^T V \operatorname{diag} V^T$$

$$= V \operatorname{diag}(\alpha)^2 V^T$$

$$A^t = V \operatorname{diag}(\alpha)^t V^T$$

$$a^t \xrightarrow{t\to\infty} 0 \text{ iff } |a|<1$$

$$\frac{1}{\eta} > \lambda_{max}(X^TX)$$

(max eigenvalue)

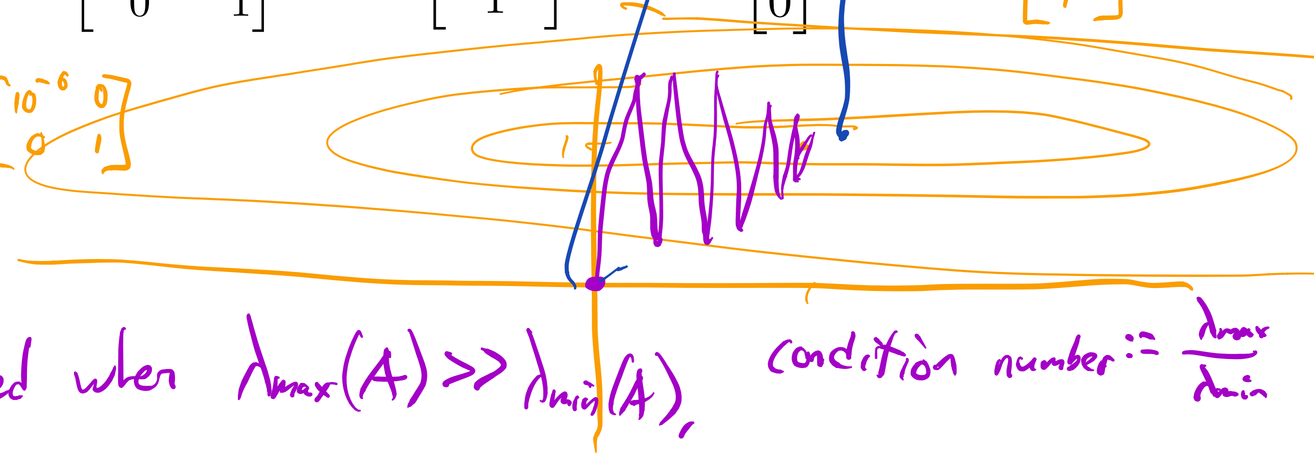
Gradient Descent $f(w) = \frac{1}{2}||Xw - y||_2^2$

$$w_{t+1} = w_t - \eta \nabla f(w_t)$$

$$(w_{t+1} - w_*) = (I - \eta X^T X)(w_t - w_*)$$
$$= (I - \eta X^T X)^{t+1}(w_0 - w_*)$$

Example: $X = \begin{bmatrix} 10^{-3} & 0 \\ 0 & 1 \end{bmatrix}$ $y = \begin{bmatrix} 10^{-3} \\ 1 \end{bmatrix}$ $w_0 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$ $w_* = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$

$X^T X = \begin{bmatrix} 10^{-6} & 0 \\ 0 & 1 \end{bmatrix}$



A is ill conditioned when $\lambda_{max}(A) >> \lambda_{min}(A)$, condition number := $\frac{\lambda_{max}}{\lambda_{min}}$

Taylor Series Approximation

- Taylor series in one dimension:

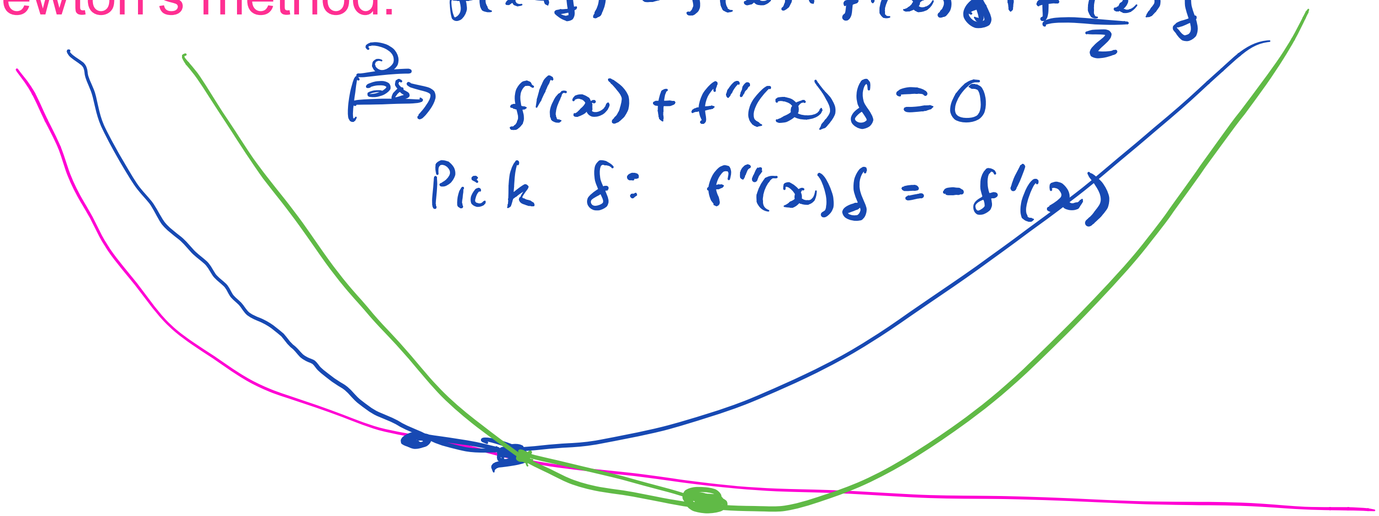
$$f(x+\delta) = f(x) + f'(x)\delta + \frac{1}{2}f''(x)\delta^2 + \dots$$

- Newton's method:

$$f(x+\delta) = f(x) + f'(x)\delta + \frac{f''(x)}{2}\delta^2$$

$$\frac{\partial}{\partial \delta} \Rightarrow \quad f'(x) + f''(x)\delta = 0$$

Pick δ: $\quad f''(x)\delta = -f'(x)$

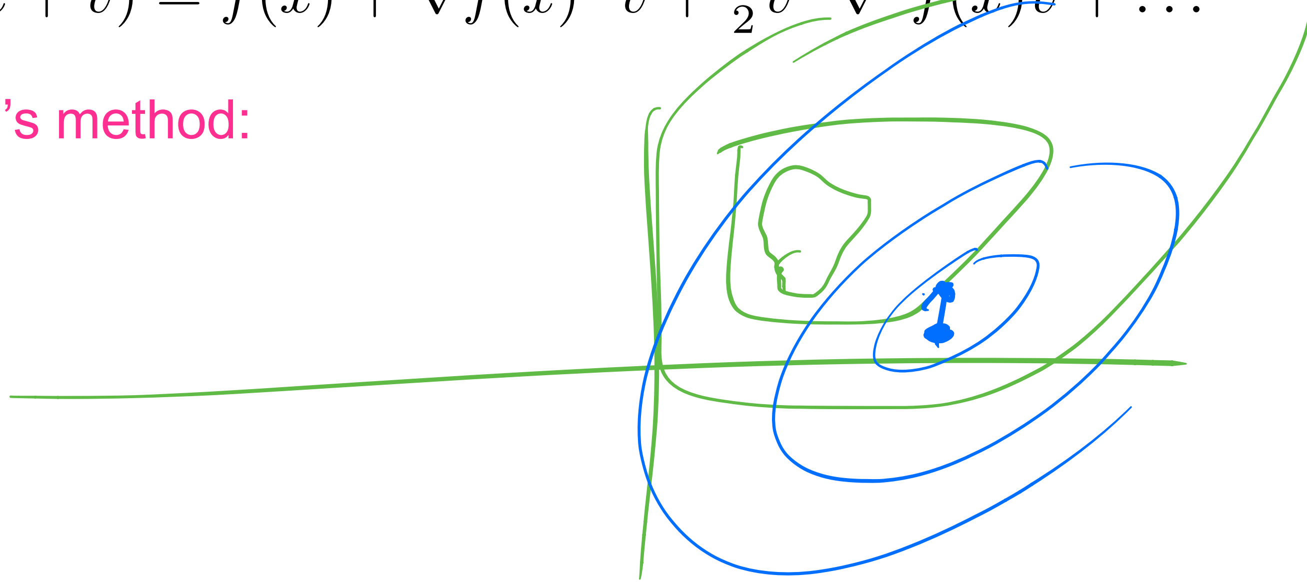


Taylor Series Approximation

- Taylor series in **d** dimensions:

$$f(x+v) = f(x) + \nabla f(x)^T v + \frac{1}{2} v^T \nabla^2 f(x) v + \ldots$$

- Newton's method:



Newton's Method $f(w) = \frac{1}{2}||\mathrm{X}w - \mathrm{y}||_2^2$

$$\nabla f(w) = X^T(Xw - y)$$

$$\nabla^2 f(w) = X^T X$$

$$v_t \text{ is solution to : } \nabla^2 f(w_t) v_t = -\nabla f(w_t)$$

$$w_{t+1} = w_t + \eta v_t$$

Newton's Method $f(w) = \frac{1}{2}||Xw - y||_2^2$

$$\nabla f(w) = X^T(Xw - y)$$

$$\nabla^2 f(w) = X^T X$$

v_t is solution to : $\nabla^2 f(w_t) v_t = -\nabla f(w_t)$

$$w_{t+1} = w_t + \eta v_t$$

$= X^T X$

$v_t = (X^T X)^{-1} X^T y$

For quadratics, Newton's method converges in one step! (Not a surprise, why?)

$$w_1 = w_0 - \eta (X^T X)^{-1} X^T (X w_0 - y) = w_*$$

$(X^T X)^{-1} X^T y$

General case

In general for Newton's method to achieve $f(w_t) - f(w_*) \leq \epsilon$:

$t \simeq O(\log(\log(1/\epsilon)))$

So why are ML problems overwhelmingly solved by gradient methods?

Hint: v_t is solution to : $\nabla^2 f(w_t) v_t = -\nabla f(w_t)$

General Convex case $f(w_t) - f(w_*) \leq \epsilon$

Clean convergence proofs: Bubeck

Newton's method:

$$t \approx \log(\log(1/\epsilon))$$

Gradient descent:

- f is *smooth* and *strongly convex:* $aI \preceq \nabla^2 f(w) \preceq bI$
- f is *smooth*: $\nabla^2 f(w) \preceq bI$
- f is potentially non-differentiable: $||\nabla f(w)||_2 \leq c$

Nocedal +Wright, Bubeck

Other: BFGS, Heavy-ball, BCD, SVRG, ADAM, Adagrad,…

Revisiting… Logistic Regression

Machine Learning – CSE546

Kevin Jamieson
University of Washington

October 16, 2016

Loss function: Conditional Likelihood

- Have a bunch of iid data of the form: $\{(x_i, y_i)\}_{i=1}^n \quad x_i \in \mathbb{R}^d, \quad y_i \in \{-1, 1\}$

$$\widehat{w}_{MLE} = \arg\max_w \prod_{i=1}^n P(y_i|x_i, w) \qquad P(Y = y|x, w) = \frac{1}{1 + \exp(-y\, w^T x)}$$

$$f(w) = \arg\min_w \sum_{i=1}^n \log(1 + \exp(-y_i\, x_i^T w))$$

$$\nabla f(w) =$$

Stochastic Gradient Descent: A Learning perspective

Machine Learning – CSE546

Kevin Jamieson
University of Washington

October 16, 2016

Learning Problems as Expectations

- Minimizing loss in training data:
  - Given dataset:
    - Sampled iid from some distribution p(**x**) on features:
  - Loss function, e.g., hinge loss, logistic loss,…
  - We often minimize loss in training data:

$$\ell_{\mathcal{D}}(\mathbf{w}) = \frac{1}{N}\sum_{j=1}^{N} \ell(\mathbf{w}, \mathbf{x}^j)$$

- However, we should really minimize expected loss on all data:

$$\ell(\mathbf{w}) = E_{\mathbf{x}}\left[\ell(\mathbf{w}, \mathbf{x})\right] = \int p(\mathbf{x})\ell(\mathbf{w}, \mathbf{x})d\mathbf{x}$$

- So, we are approximating the integral by the average on the training data

Gradient ascent in Terms of Expectations

- "True" objective function:

$$\ell(\mathbf{w}) = E_{\mathbf{x}}\left[\ell(\mathbf{w},\mathbf{x})\right] = \int p(\mathbf{x})\ell(\mathbf{w},\mathbf{x})d\mathbf{x}$$

- Taking the gradient:
- "True" gradient ascent rule:
- How do we estimate expected gradient?

SGD: Stochastic Gradient Ascent (or Descent)

- "True" gradient: $$\nabla \ell(\mathbf{w}) = E_{\mathbf{x}}\left[\nabla \ell(\mathbf{w}, \mathbf{x})\right]$$

- Sample based approximation:

- What if we estimate gradient with just one sample???
  - Unbiased estimate of gradient
  - Very noisy!
  - Called stochastic gradient ascent (or descent)
    - Among many other names
  - VERY useful in practice!!!

Stochastic Gradient Ascent for Logistic Regression

- Logistic loss as a stochastic function:

$$E_{\mathbf{x}}\left[\ell(\mathbf{w},\mathbf{x})\right] = E_{\mathbf{x}}\left[\ln P(y|\mathbf{x},\mathbf{w}) - \lambda||\mathbf{w}||_2^2\right]$$

- Batch gradient ascent updates:

$$w_i^{(t+1)} \leftarrow w_i^{(t)} + \eta \left\{ -\lambda w_i^{(t)} + \frac{1}{N}\sum_{j=1}^{N} x_i^{(j)}[y^{(j)} - P(Y=1|\mathbf{x}^{(j)},\mathbf{w}^{(t)})] \right\}$$

- Stochastic gradient ascent updates:
  - Online setting:

$$w_i^{(t+1)} \leftarrow w_i^{(t)} + \eta_t \left\{ -\lambda w_i^{(t)} + x_i^{(t)}[y^{(t)} - P(Y=1|\mathbf{x}^{(t)},\mathbf{w}^{(t)})] \right\}$$