Logistic Regression

Machine Learning – CSEP546

Carlos Guestrin

University of Washington

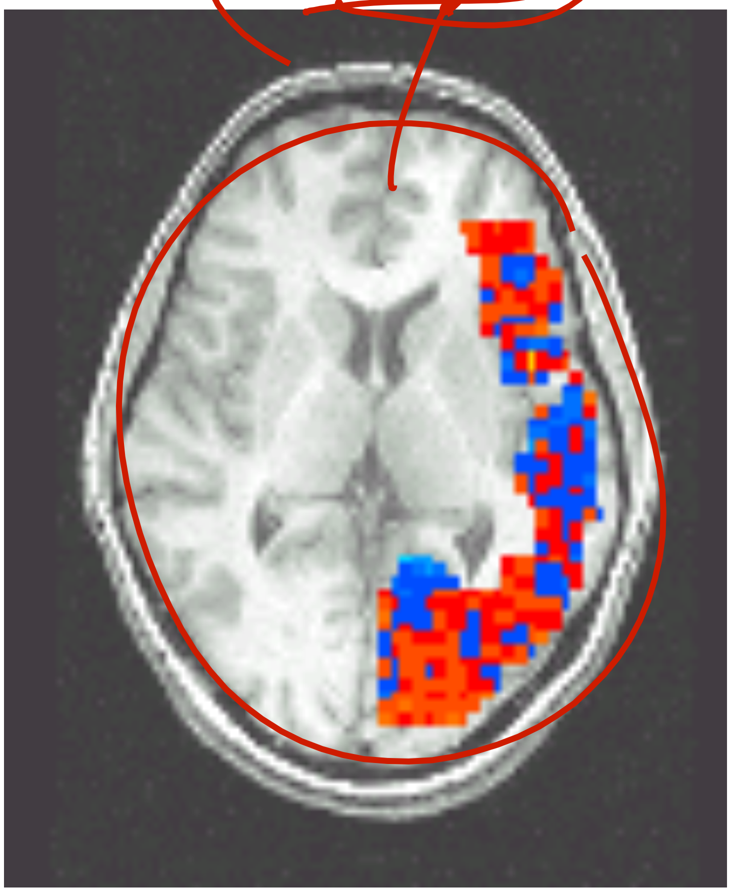
January 27, 2014

Reading Your Brain, Simple Example

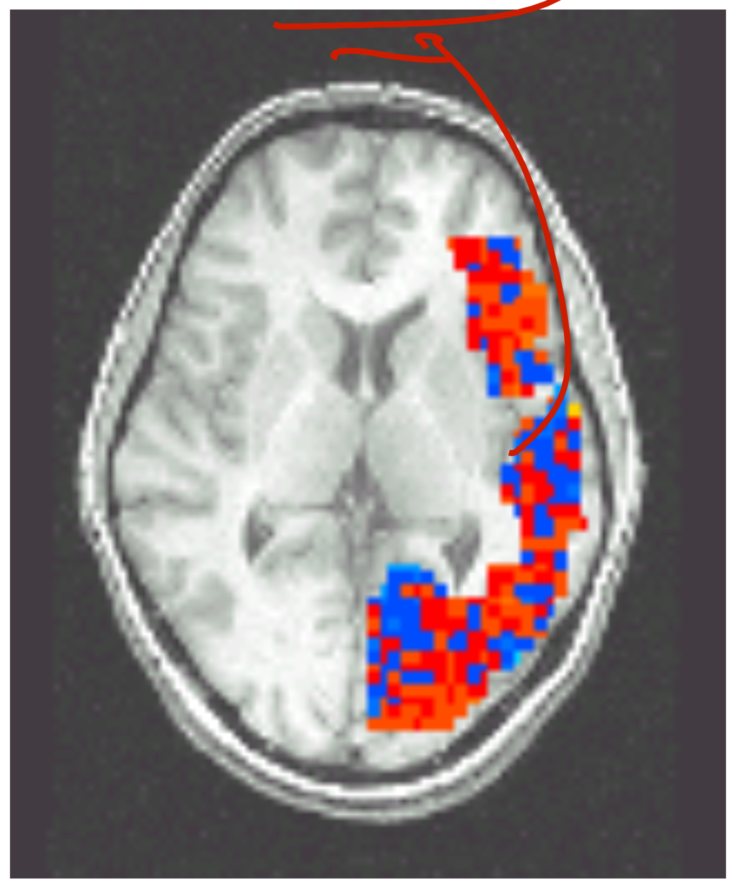
[Mitchell et al.]

Pairwise classification accuracy: 85%

Person



Animal



Classification

- **Learn**: h:**X** $\mapsto$ Y
  - **X** – features
  - Y – target classes

$X = (GPA, grade, resume, \ldots)$

$Y = \{hired, not\ hired\}$

- **Simplest case: Thresholding**

X: Load Computer

Y: alarm?

Load > 99% ⇒ alarm = true

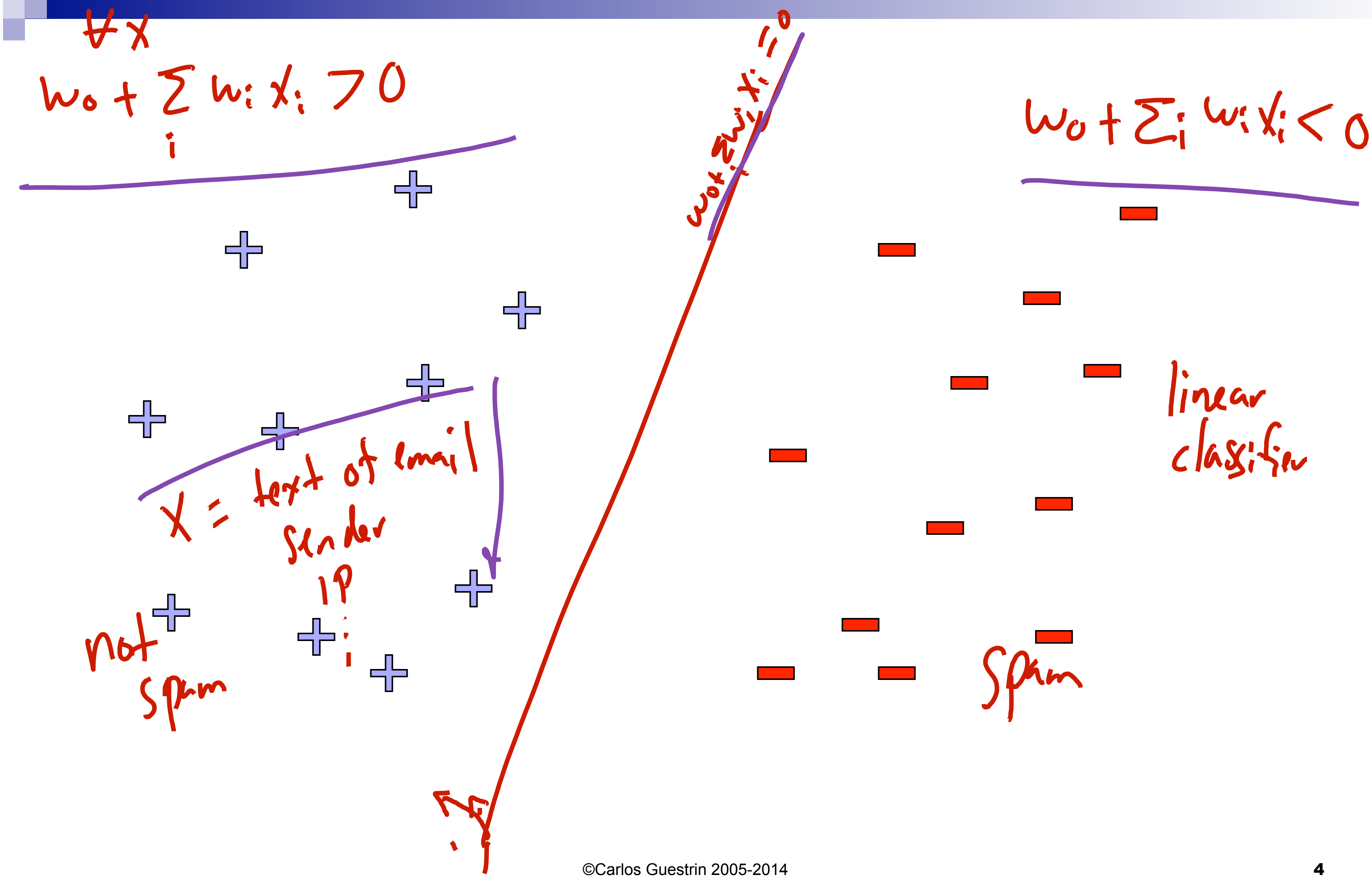
X_i

else ⇒ alarm = false

$X_j > 27°C$

Linear (Hyperplane) Decision Boundaries

$w_0 + \sum_i w_i x_i \overset{?}{>} 0$



Classification

- **Learn**: h:**X** $\mapsto$ Y
  - **X** – features
  - Y – target classes

- Thus far: just a decision boundary

$\hat{y} = \text{sign}(w \cdot x)$ ← yes/no decision

- What if you want probability of each class? P(Y|X)

$P(Y = \text{spam} \mid X = \text{text of email})$

$$\hat{y} = \arg\max_{y} P(Y = y \mid X = \text{text of email})$$

Ad Placement Strategies

- Companies bid on ad prices

$c_1 \to \$10$

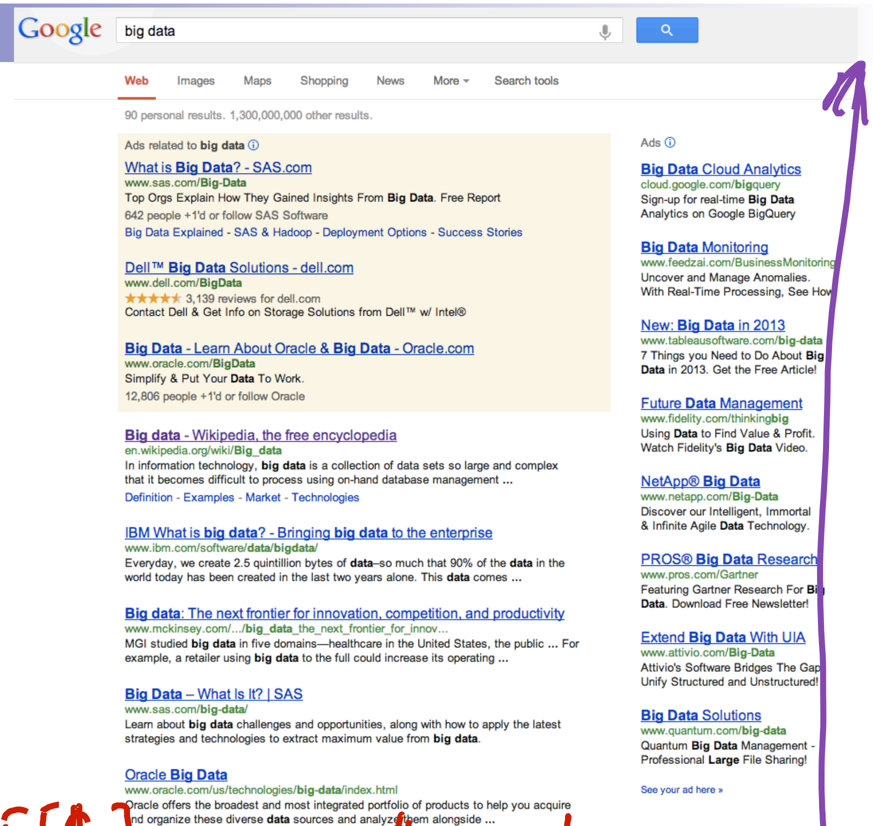
$c_2 \to \$20$

$c_3 \to \$100$

- Which ad wins? (many simplifications here)
  - Naively: $c_3 \to \$100$
  - But: paid on click only
  - Instead:

e.g. $p(\text{click} \mid c_3, \text{'bigdata'}) = 0.01 \Rightarrow E[\$_3] = 0.01 * \$100 = \$1$

$p(\text{click} \mid c_1, \text{'bigdata'}) = 0.5 \Rightarrow E[\$_1] = 0.5 * \$10 = \$5$



Link Functions

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

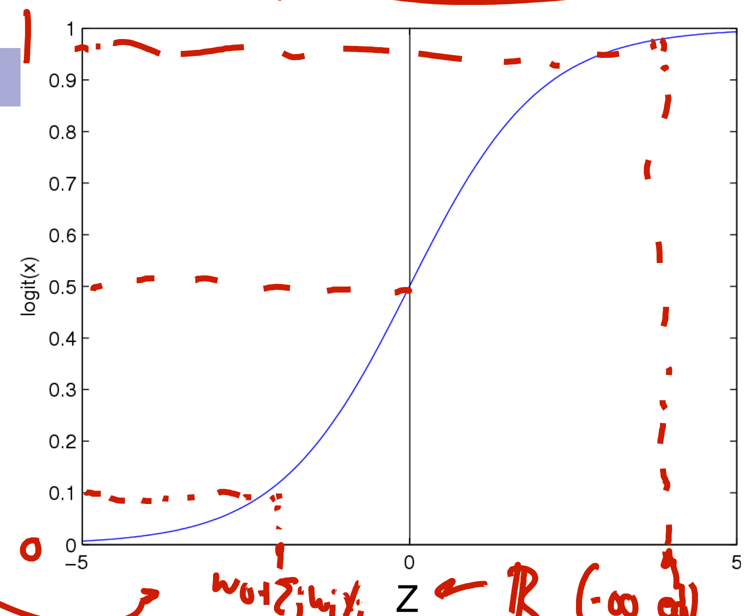
$$\underbrace{P(Y|X)}_{[0,1]} = \underbrace{w_0 + \sum_i w_i X_i}_{\mathbb{R}}$$

$\mathbb{R} \rightarrow$ maps $[0,1]$

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

Logistic Regression

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



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

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

$$P(Y=1|X,W) = 1 - P(Y=0|X,W)$$

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

Features can be discrete or continuous!

Understanding the sigmoid

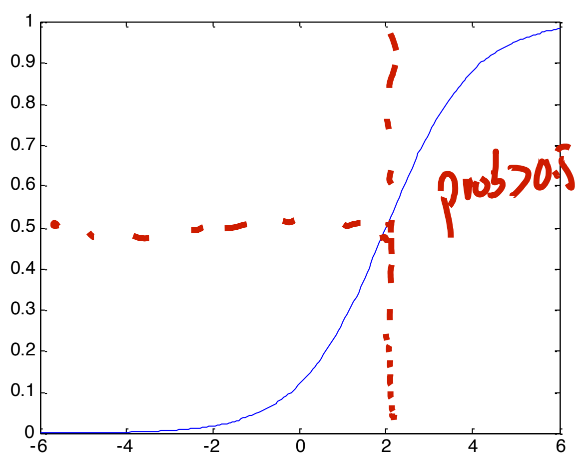
$w_2 \dots w_n = 0$

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

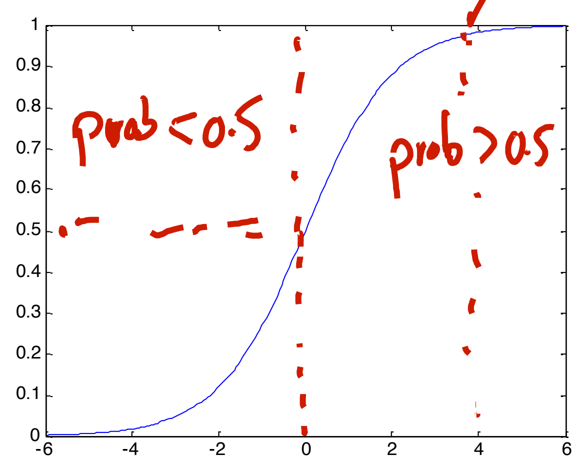
less steep

Shift

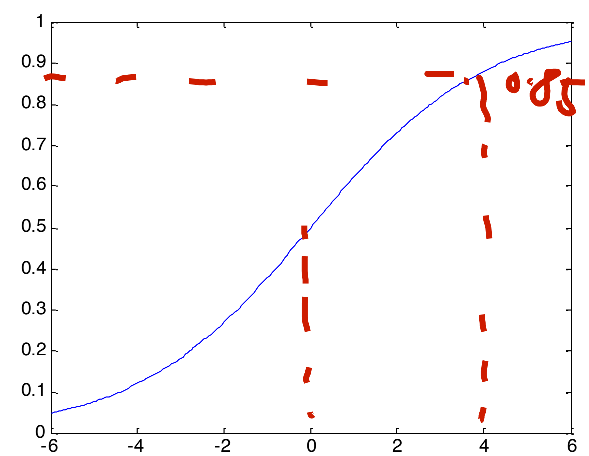
w_0=-2, w_1=-1



w_0=0, w_1=-1

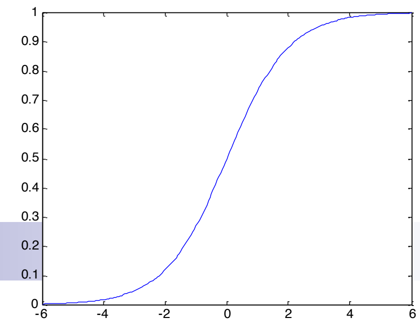


w_0=0, w_1=-0.5



Logistic Regression – a Linear classifier

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



$P(Y=0|W,X)=$

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

$w_0 + \sum_i w_i x_i = 0$

$w_0 + \sum_i w_i x_i > 0$

$\Downarrow$

$g(w_0 + \sum_i w_i x_i) < 0.5$

$\Downarrow$

$P(Y=0|w,x) < 0.5$

$P(Y=1|w,x) > 0.5$

$\Rightarrow$ predict 1

$w_0 + \sum_i w_i x_i < 0$

$P(Y=0|w,x) > 0.5$

$\Downarrow$

predict 0

Very convenient!

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

implies

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

implies

predict 1 ⇐ $1 < \dfrac{P(Y=1\,|X)}{P(Y=0\,|X)} = exp(w_0+\sum_i w_i X_i)$

linear classification rule!

implies

predict 1 ⇐ $0 < \ln \dfrac{P(Y=1\,|X)}{P(Y=0\,|X)} = w_0 + \sum_i w_i X_i \overset{?}{>} 0$

$X_i^j \leftarrow$ ith dimension of jth datapoint

Loss function: Conditional Likelihood

- Have a bunch of iid data of the form: $D_X, D_Y \equiv (x^j, y^j)_{1:N}$

X — GPA, ML grade..

Y — hired, not hired

- Discriminative (logistic regression) loss function:
  Conditional Data Likelihood — $P(Y|X,w)$

$$\arg\max_w P(D_Y | D_X, w) \equiv \arg\max_w \prod_{j=1}^{N} P(y^j | x^j, w)$$

$P(1 | GPA=4.0, w) \; P(0 | GPA=3.0, w)$

$(Y^1=1, GPA=4.0)$
$(Y^2=0, GPA=3.0)$

true class, features, params

$$\arg\max_w \ln P(\mathcal{D}_Y \mid \mathcal{D}_{\mathbf{X}}, \mathbf{w}) = \sum_{j=1}^{N} \ln P(y^j \mid \mathbf{x}^j, \mathbf{w})$$

Expressing Conditional Log Likelihood

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

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

$y^j \in \{0,1\}$

$$l(\mathbf{w}) \equiv \sum_{j=1}^{N} \ln P(y^j|\mathbf{x}^j,\mathbf{w})$$

$$\equiv \sum_{j=1}^{N} y^j \ln P(Y=1|x^j,w) + (1-y^j) \ln P(Y=0|x^j,w)$$

$$\ell(\mathbf{w}) = \sum_j y^j \ln P(Y=1|\mathbf{x}^j,\mathbf{w}) + (1-y^j)\ln P(Y=0|\mathbf{x}^j,\mathbf{w})$$

$$= \sum_j y^j(w_0+\sum_i^n w_i x_i^j) - \ln(1+exp(w_0+\sum_i^n w_i x_i^j))$$

want to $\arg\max_w$

Maximizing Conditional Log Likelihood

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

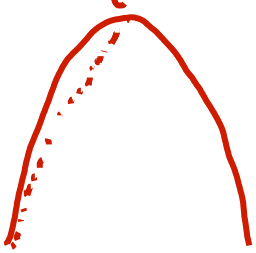
$$P(Y=1|X,W) = \frac{exp(w_0+\sum_i w_i X_i)}{1+exp(w_0+\sum_i w_i X_i)}$$

argmax w

$$l(\mathbf{w}) \equiv \ln \prod_j P(y^j|\mathbf{x}^j,\mathbf{w})$$

I want to get here

$$= \sum_j y^j(w_0+\sum_i^n w_i x_i^j) - \ln(1+exp(w_0+\sum_i^n w_i x_i^j))$$



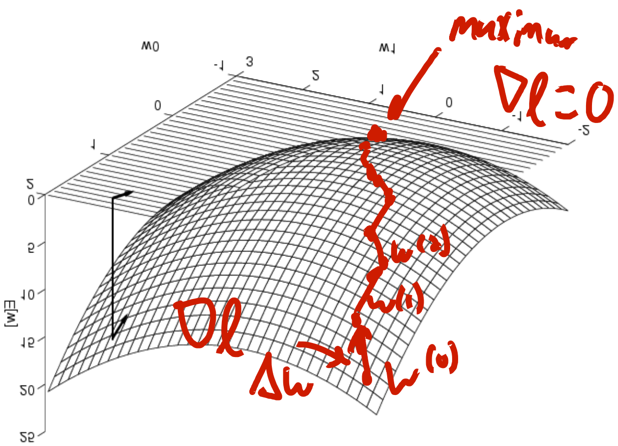
Good news: $l(\mathbf{w})$ is concave function of **w**, no local optima problems

Bad news: no closed-form solution to maximize $l(\mathbf{w})$

Good news: concave functions easy to optimize

Optimizing concave function – Gradient ascent

- Conditional likelihood for Logistic Regression is concave. Find optimum with gradient ascent



Gradient: $$\nabla_{\mathbf{w}} l(\mathbf{w}) = [\frac{\partial l(\mathbf{w})}{\partial w_0}, \ldots, \frac{\partial l(\mathbf{w})}{\partial w_n}]'$$

Step size, η>0

Update rule: $$\Delta \mathbf{w} = \eta \nabla_{\mathbf{w}} l(\mathbf{w})$$

$$w_i^{(t+1)} \leftarrow w_i^{(t)} + \eta \frac{\partial l(\mathbf{w})}{\partial w_i}$$

option

η = constant

often good

η_t e.g. $\frac{\text{constant}}{t}$

or $\frac{\text{constant}}{\sqrt{t}}$

- Gradient ascent is simplest of optimization approaches
  - e.g., Conjugate gradient ascent can be much better

Coordinate Descent v. Gradient Descent

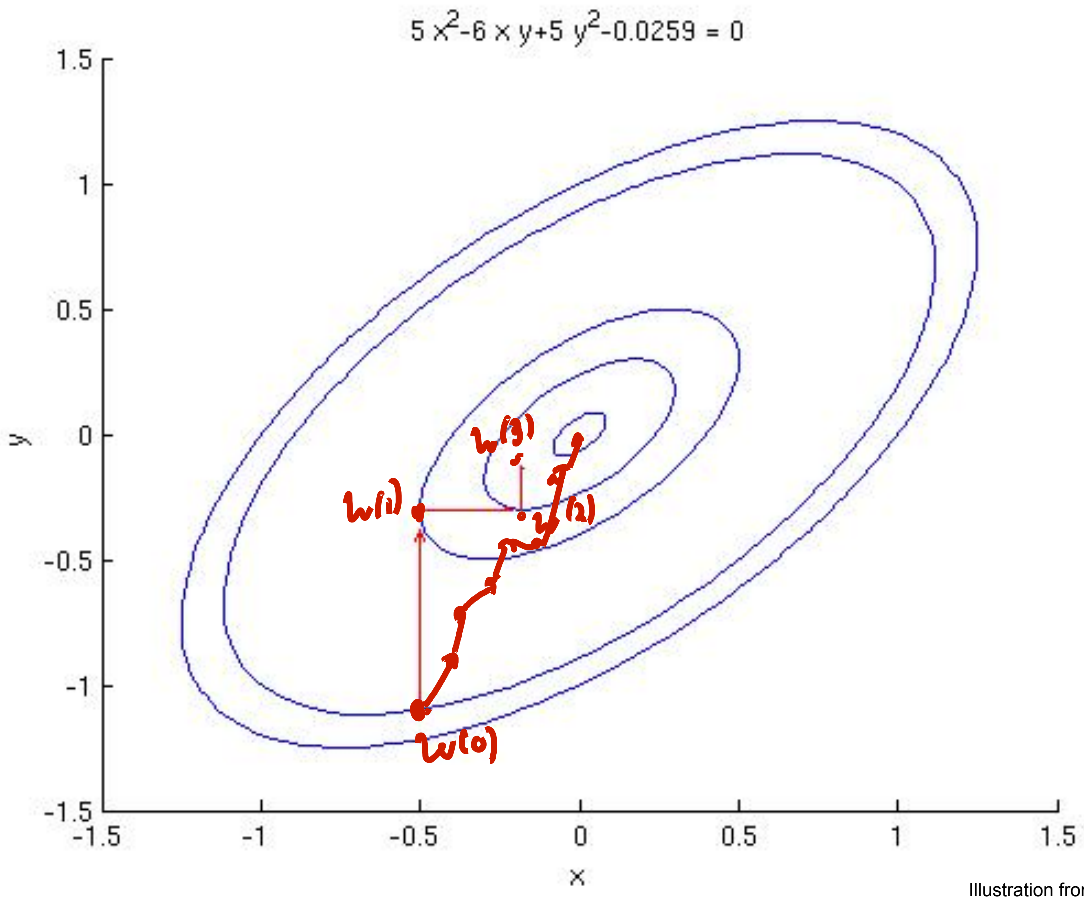


Illustration from Wikipedia

Maximize Conditional Log Likelihood: Gradient ascent

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

$$l(\mathbf{w}) \;=\; \sum_j y^j (w_0 + \sum_i^n w_i x_i^j) - \ln(1 + exp(w_0 + \sum_i^n w_i x_i^j))$$

Sum over data points

uphill

$$\frac{\partial \ell(\mathbf{w})}{\partial w_i} = \sum_{j=1}^{N} x_i^j \left(y^j - P(Y=1|x^j, \mathbf{w})\right)$$

if x_i positive

truth	prediction	
1	<1	more x_i
0	prediction >0	less x_i

Gradient Descent for LR: Intuition

typically normalize features e.g., 0 mean variance 1 ✓ subtract mean then divide by std.

Gender	Age	Location	Income	Referrer	New or Returning	Clicked?
F	Young	US	High	Google	New	N
M	Middle	US	Low	Direct	New	N
F	Old	BR	Low	Google	Returning	Y
M	Young	BR	Low	Bing	Returning	N

Gender (F=1, M=0)	Age (Young=0, Middle=1, Old=2)	Location (US=1, Abroad=0)	Income (High=1, Low=0)	Referrer: is Goo?	is Bing?	is direct?	New or Returning (New=1,, Returning =0)	Clicked? (Click=1, NoClick=0)
1	0	1	1	1	0	0	1	0
0	1	1	0	0	0	1	1	0
1	2	0	0	1	0	0	0	1
0	0	0	0	0	1	0	0	0

1. Encode data as numbers

   X_i is keyword 'big data'

   $x_{i,l}$ 0 0 0 1 0 0 0 — x_i, length of vocab

2. Until convergence: for each feature
   a. Compute ~~average~~ sum gradient over data points
   b. Update parameter

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

Gradient Ascent for LR

w or $\|\nabla \ell\|$

Gradient ascent algorithm: iterate until change < ε

$$w_0^{(t+1)} \leftarrow w_0^{(t)} + \eta \sum_{j=1}^{N} [y^j - \hat{P}(Y^j = 1 \mid \mathbf{x}^j, \mathbf{w}^{(t)})]$$

For i=1,…,k,

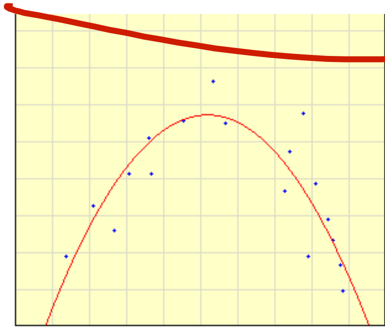
$$w_i^{(t+1)} \leftarrow w_i^{(t)} + \eta \sum_{j=1}^{N} x_i^j [y^j - \hat{P}(Y^j = 1 \mid \mathbf{x}^j, \mathbf{w}^{(t)})]$$

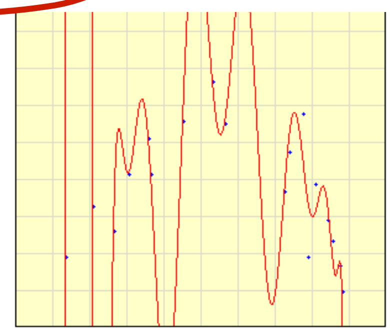
repeat

Regularization in linear regression

- Overfitting usually leads to very large parameter choices, e.g.:

  $-2.2 + 3.1\,X - 0.30\,X^2$ $\qquad\qquad$ $-1.1 + 4{,}700{,}910.7\,X - 8{,}585{,}638.4\,X^2 + \ldots$





- Regularized least-squares (a.k.a. ridge regression), for $\lambda>0$:

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} \sum_j \left(t(\mathbf{x}_j) - \sum_i w_i h_i(\mathbf{x}_j) \right)^2 + \lambda \underbrace{\sum_{i=1}^{k} w_i^2}_{\|w\|_2}$$

$\|w\|_1$

Linear Separability

$w_0 + \sum_i w_i x_i > 0$

$2w_0 + \sum_i 2w_i x_i > 0$

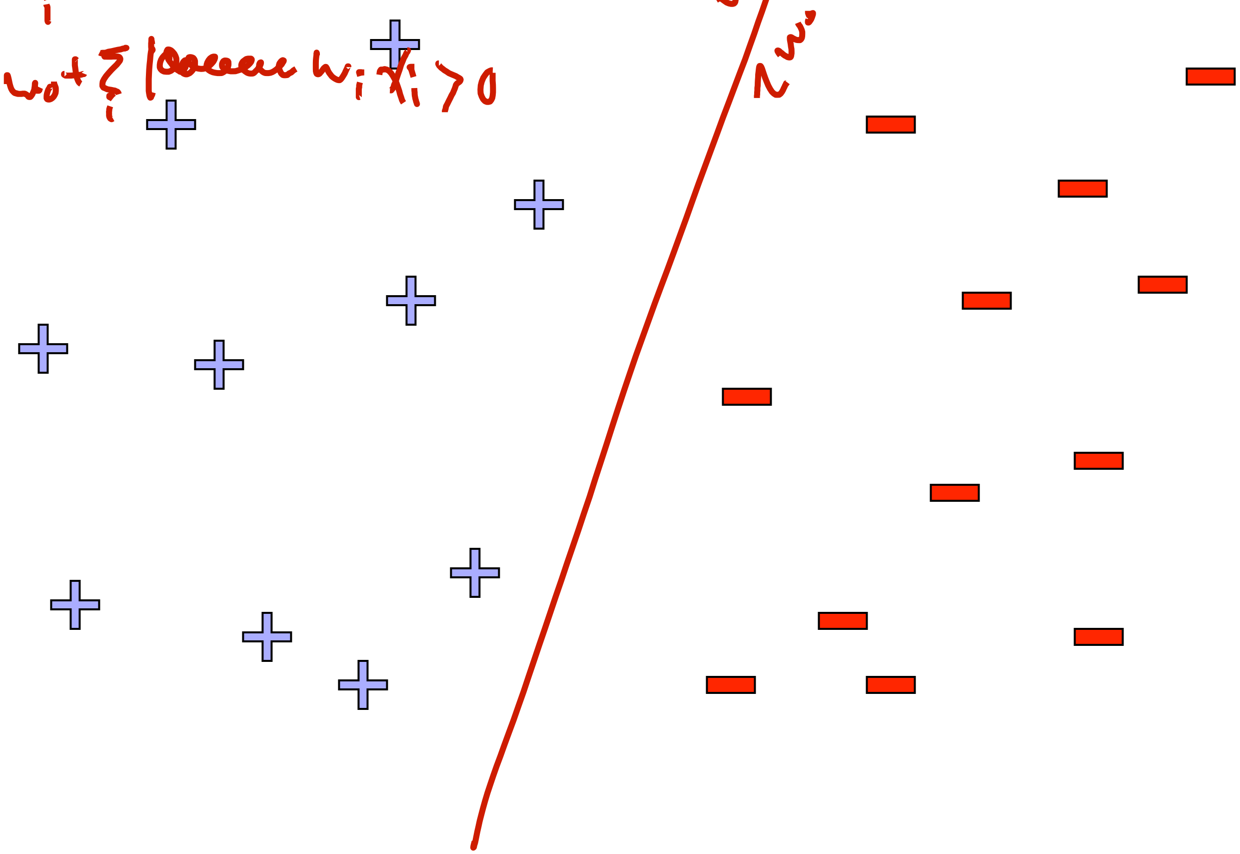
$1000000 w_0 + \sum_i 1000000 w_i x_i > 0$

$w_0 + \sum_i w_i x_i = 0$

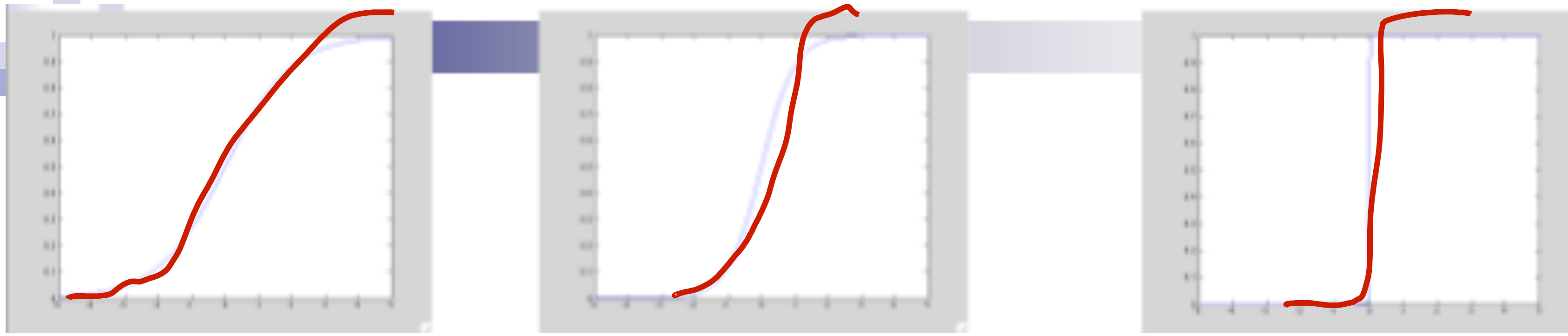
$2w_0 + \sum_i 2w_i x_i = 0$

$w_0 + \sum_i w_i x_i < 0$

$2w_0 + \sum_i 2w_i x_i < 0$



Large parameters → Overfitting



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

gradient will pick most "sure" answer

- If data is linearly separable, weights go to infinity

$P(Y=1|w,x) \rightarrow 1$

  - In general, leads to overfitting:
- Penalizing high weights can prevent overfitting…

Regularized Conditional Log Likelihood

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

$$\ell(\mathbf{w}) = \underbrace{\ln \prod_{j=1}^{N} P(y^j|\mathbf{x}^j, \mathbf{w})}_{\text{conditional log likelihood}} - \underbrace{\frac{\lambda}{2}||\mathbf{w}||_2^2}_{\text{regularization}}$$

$$\sum_{i=1}^{d} w_i^2$$

- Practical note about w_0:

don't regularize

- Gradient of regularized likelihood:

$$\frac{\partial \ell}{\partial w_i} = \frac{\partial}{\partial w_i}\left[\ln \prod_{j=1}^{N} P(y^j|x^j, w)\right] - \lambda w_i$$

Standard v. Regularized Updates

- Maximum conditional likelihood estimate

$$\mathbf{w}^* = \arg\max_{\mathbf{w}} \quad \ln \prod_{j=1}^{N} P(y^j|\mathbf{x}^j, \mathbf{w})$$

$$w_i^{(t+1)} \leftarrow w_i^{(t)} + \eta \sum_j x_i^j \underbrace{[y^j - \hat{P}(Y^j = 1 \mid \mathbf{x}^j, \mathbf{w}^{(t)})]}_{\text{margin}}$$

- Regularized maximum conditional likelihood estimate

$$\mathbf{w}^* = \arg\max_{\mathbf{w}} \quad \ln \prod_{j=1}^{N} P(y^j|\mathbf{x}^j, \mathbf{w}) - \frac{\lambda}{2} \sum_{i=1}^{k} w_i^2$$

extra term

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

Please Stop!! Stopping criterion

$$\ell(\mathbf{w}) = \ln \prod_j P(y^j|\mathbf{x}^j, \mathbf{w})) - \lambda||\mathbf{w}||_2^2$$

- When do we stop doing gradient descent?

w^* is optimal value of w

$\ell(w^*) - \ell(w^{(t)}) \leq \varepsilon$

$||\nabla \ell(w)||_2^2 = \sum_i \left(\frac{\partial \ell}{\partial w_i}\right)^2$

- Because *l*(**w**) is strongly concave:
  - i.e., because of some technical condition

$$\ell(\mathbf{w}^*) - \ell(\mathbf{w}^{(t)}) \leq \frac{1}{2\lambda}||\nabla\ell(\mathbf{w})||_2^2 \leq \varepsilon$$

regularization λ

- Thus, stop when: $||\nabla \ell(w^{(t)})||_2^2 \leq 2\lambda\varepsilon$

Digression: Logistic regression for more than 2 classes

- Logistic regression in more general case (C classes), where *Y in* {0,…,C-1}

$Y \in \{$ sunny, cloudy, rain, ... $\}$

Digression: Logistic regression more generally

- Logistic regression in more general case, where *Y in* {0,…,C-1}

for *c>0*

$$P(Y=c|\mathbf{x},\mathbf{w}) = \frac{\exp(w_{c0}+\sum_{i=1}^{k} w_{ci}x_i)}{1+\sum_{c'=1}^{C-1}\exp(w_{c'0}+\sum_{i=1}^{k} w_{c'i}x_i)}$$

for *c=0* (normalization, so no weights for this class)

$$P(Y=0|\mathbf{x},\mathbf{w}) = \frac{1}{1+\sum_{c'=1}^{C-1}\exp(w_{c'0}+\sum_{i=1}^{k} w_{c'i}x_i)}$$

Learning procedure is basically the same as what we derived!

Stochastic Gradient Descent

Machine Learning – CSEP546

Carlos Guestrin

University of Washington

January 27, 2014

The Cost, The Cost!!! Think about the cost… Say k dims

- What's the cost of a gradient update step for LR???

$$w_i^{(t+1)} \leftarrow w_i^{(t)} + \eta \left\{ -\lambda w_i^{(t)} + \sum_{j=1}^{N} x_i^j [y^j - \underbrace{\hat{P}(Y^j = 1 \mid \mathbf{x}^j, \mathbf{w}^{(t)})]}_{O(k)} \right\}$$

$\forall i$

$O(Nk)$ per parameter

total $O(Nk^2)$

cache $P(Y=1 \mid x^j, w^{(t)})$ per iteration

use $\forall i$

$O(Nk)$

Big Data: $N = 100B$

linear data size per η step

Learning Problems as Expectations

- Minimizing loss in training data:
  - Given dataset: $x^1, x^2, \ldots, x^N$
    - Sampled iid from some distribution p(**x**) on features:
  - Loss function, e.g., squared error, logistic loss,…
  - We often minimize loss in training data:

train error

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

error for least squares regression

$$\ell(w, x^j) = \left(t(x^j) - \left(w_0 + \sum_i w_i x_i^j\right)\right)^2$$

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

true error

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

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

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:

$$\nabla_w \ell(w) \approx \frac{1}{N} \sum_{j=1}^{N} \nabla_w \ell(w, x^j)$$

- 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!!!

at each step t estimate gradient by $\nabla_w \ell(w) \approx \nabla_w \ell(w, x^{(t)})$

Stochastic Gradient Ascent for Logistic Regression

mini batch
↳ 5 – 100 datapoints to estimate gradient

- 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:

sum over all points

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

- Stochastic gradient ascent updates: pick wrt example $(x^{(t)}, y^{(t)})$
  - Online setting:

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

O(k) per iteration

$\eta_t = \frac{\text{constant}}{t}$ or $\frac{\text{constant}}{\sqrt{t}}$

Stochastic Gradient Descent for LR: Intuition

Want order of data points to be random

Gender	Age	Location	Income	Referrer	New or Returning	Clicked?
F	Young	US	High	Google	New	N
M	Middle	US	Low	Direct	New	N
F	Old	BR	Low	Google	Returning	Y
M	Young	BR	Low	Bing	Returning	N

Gender (F=1, M=0)	Age (Young=0, Middle=1, Old=2)	Location (US=1, Abroad=0)	Income (High=1, Low=0)	Referrer G	B	D	New or Returning (New=1,, Returning =0)	Clicked? (Click=1, NoClick=0)
1	0	1	1	1	0	0	1	0
0	1	1	0	0	0	1	1	0

1. Until convergence: get a data point
   a. Encode data as numbers
   b. For each feature
      i. Compute gradient for this data point
      ii. Update parameter

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

Stochastic Gradient Ascent: general case

- Given a stochastic function of parameters: $\ell(w) \approx \frac{1}{N}\sum_{j=1}^{N} \ell(w, x^j)$
  - Want to find maximum

$$\arg\max_w \ell(w)$$

- Start from $\mathbf{w}^{(0)}$ $\leftarrow$ e.g. $w^{(0)} = 0$
- Repeat until convergence:
  - Get a sample data point $\mathbf{x}^t$
  - Update parameters:

$$w_i^{(t+1)} \leftarrow w_i^{(t)} + \eta_t \frac{\partial}{\partial w_i} \ell(w^{(t)}, x^t)$$

- Works on the online learning setting!
- Complexity of each gradient step is constant in number of examples!
- In general, step size changes with iterations

What you should know…

- Classification: predict discrete classes rather than real values
- Logistic regression model: Linear model
  - Logistic function maps real values to [0,1]
- Optimize conditional likelihood
- Gradient computation
- Overfitting
- Regularization
- Regularized optimization
- Cost of gradient step is high, use stochastic gradient descent

What's the Perceptron Optimizing?

Machine Learning – CSEP546
Carlos Guestrin
University of Washington

January 27, 2014

Remember our friend the Perceptron Algorithm

- At each time step:
  - Observe a data point:

$$x^t, y^t \qquad \hat{y}^t = \text{sign}(w \cdot x^t)$$

  - Update parameters if make a mistake:

$$\text{if } \hat{y}^t \neq y^t$$
$$w^{(t+1)} = w^{(t)} + y^t x^t$$
$$\text{else} \quad w^{(t+1)} = w^{(t)}$$

What is the Perceptron Doing???

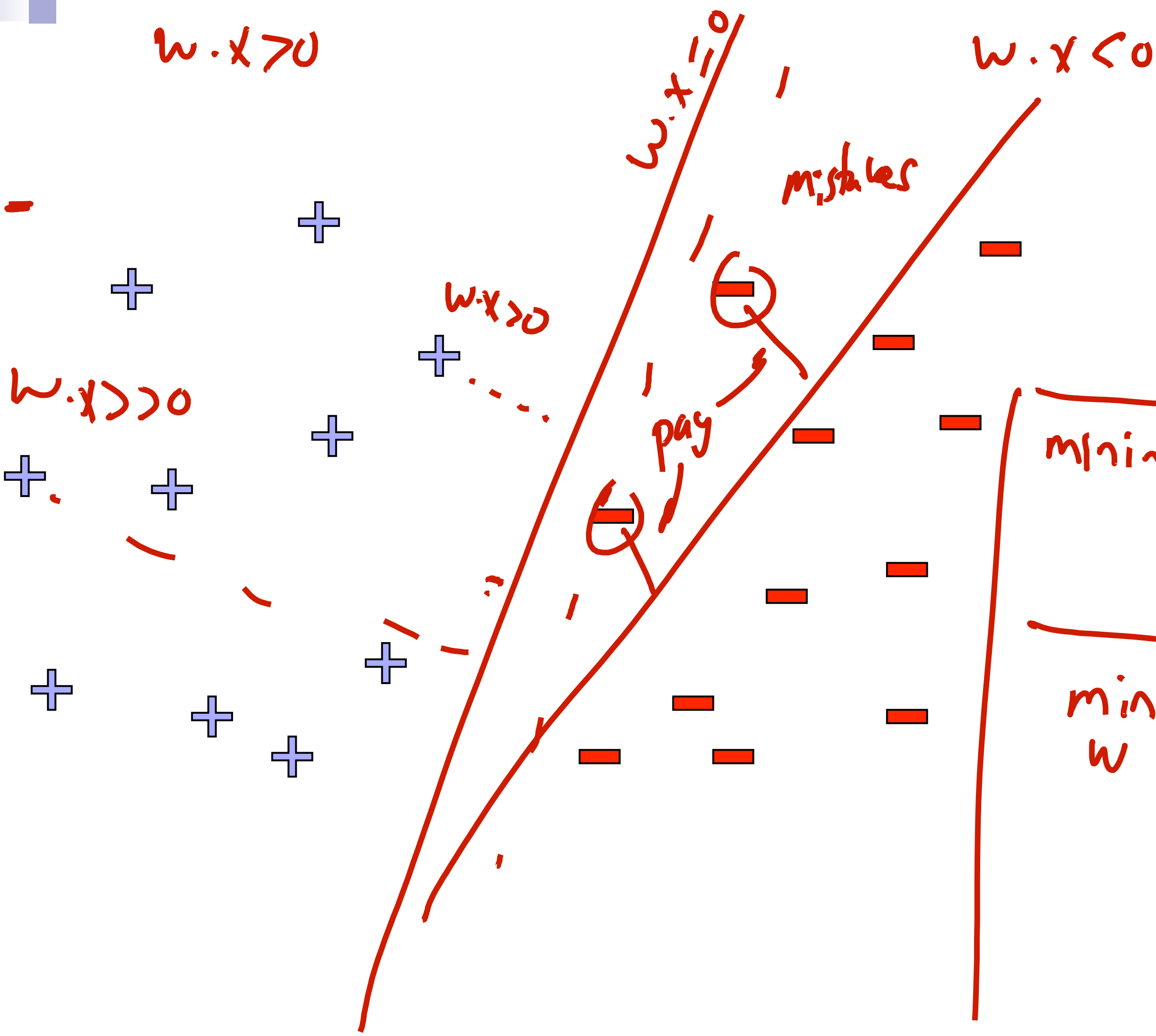
- When we discussed logistic regression:
  - Started from maximizing conditional log-likelihood

$$\arg\max_w \ \ln \prod_j P(y^j \mid x^j, w)$$

- When we discussed the Perceptron:
  - Started from description of an algorithm

- What is the Perceptron optimizing????

Perceptron Prediction: Margin of Confidence



want pick w such that confidences are as large as possible

minimize $w.x^j<0 \quad y^j=+1$

$w.x^j>0 \quad y^j=-1$

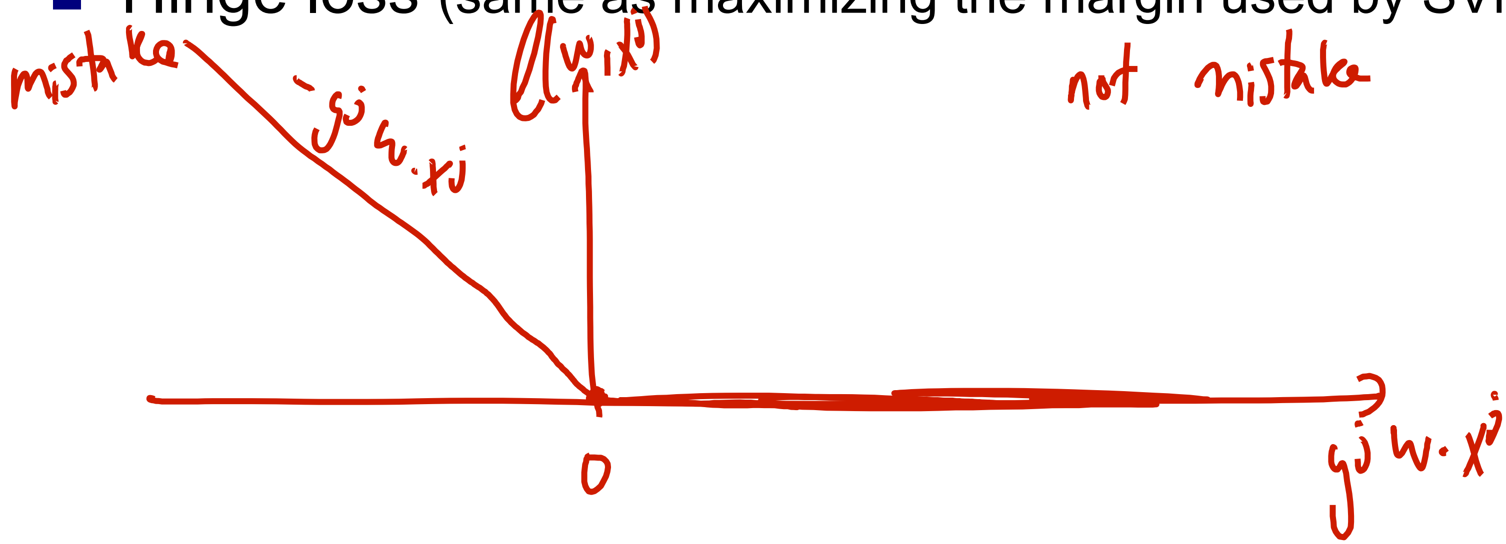
$$\min_w \sum_{\text{mistake}} -y^j \, w.x^j$$

Hinge Loss

- Perceptron prediction: $\text{sign}(w \cdot x)$
- Makes a mistake when:

$y^j \, w \cdot x^j < 0 \quad \Rightarrow \quad \ell(w, x^j) = \begin{cases} 0 & ; \; y^j \, w \cdot x^j \geq 0 \\ -y^j \, w \cdot x^j & ; \; \text{else} \end{cases}$

- Hinge loss (same as maximizing the margin used by SVMs)

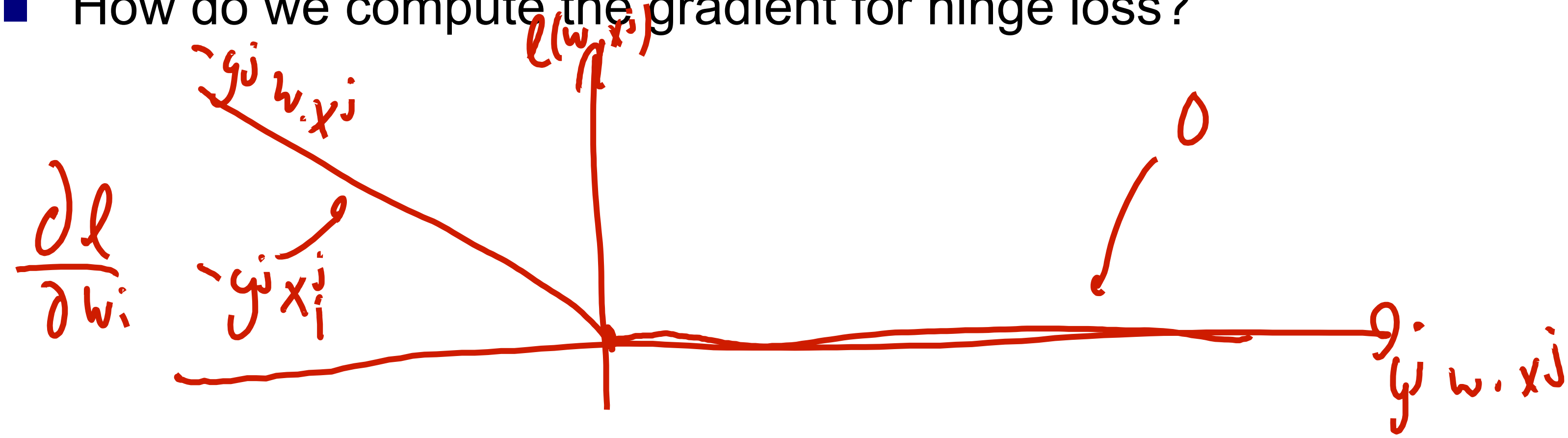


Stochastic Gradient Descent for Hinge Loss

- SGD: observe data point $x^{(t)}$, update each parameter

$$w_i^{(t+1)} \leftarrow w_i^{(t)} - \eta_t \frac{\partial \ell(\mathbf{w}^{(t)}, x^{(t)})}{\partial w_i}$$

- How do we compute the gradient for hinge loss?



(Sub)gradient of Hinge

- Hinge loss:

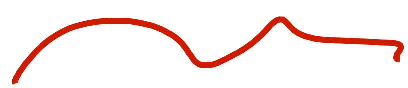
$$w_i^{(t+1)} \leftarrow w_i^{(t)} - \eta_t \frac{\partial \ell(\mathbf{w}^{(t)}, x^{(t)})}{\partial w_i}$$

- Subgradient of hinge loss:

$\nabla \ell$

  - If $y^{(t)} (w.\mathbf{x}^{(t)}) > 0$: 0
  - If $y^{(t)} (w.\mathbf{x}^{(t)}) < 0$: $-y^{(t)} x^{(t)}$
  - If $y^{(t)} (w.\mathbf{x}^{(t)}) = 0$: $[-y^{(t)} x^{(t)}, 0]$
  - In one line:

Mistake?



$$\nabla \ell(w^{(t)}, x^{(t)}) = \mathbb{1}(y^{(t)} w x^{(t)} \leq 0)(-y^{(t)} x^{(t)})$$

Stochastic Gradient Descent for Hinge Loss

- SGD: observe data point $x^{(t)}$, update each parameter

$$w_i^{(t+1)} \leftarrow w_i^{(t)} - \eta_t \frac{\partial \ell(\mathbf{w}^{(t)}, x^{(t)})}{\partial w_i}$$

- How do we compute the gradient for hinge loss?

Perceptron Revisited

SGD for hinge loss

- SGD for hinge loss:

$$\mathbf{w}^{(t+1)} \leftarrow \mathbf{w}^{(t)} + \eta_t \overbrace{\mathbb{1}\left[y^{(t)}(\mathbf{w}^{(t)} \cdot \mathbf{x}^{(t)}) \leq 0\right]}^{\text{Mistake}} y^{(t)}\mathbf{x}^{(t)}$$

- Perceptron update:

$$\mathbf{w}^{(t+1)} \leftarrow \mathbf{w}^{(t)} + \mathbb{1}\left[y^{(t)}(\mathbf{w}^{(t)} \cdot \mathbf{x}^{(t)}) \leq 0\right] y^{(t)}\mathbf{x}^{(t)}$$

- Difference? perceptron = SGD for hinge loss with $\eta_t = 1$

What you need to know

- Perceptron is optimizing hinge loss
- Subgradients and hinge loss
- (Sub)gradient decent for hinge objective

Support Vector Machines

Machine Learning – CSEP546

Carlos Guestrin

University of Washington

January 27, 2014

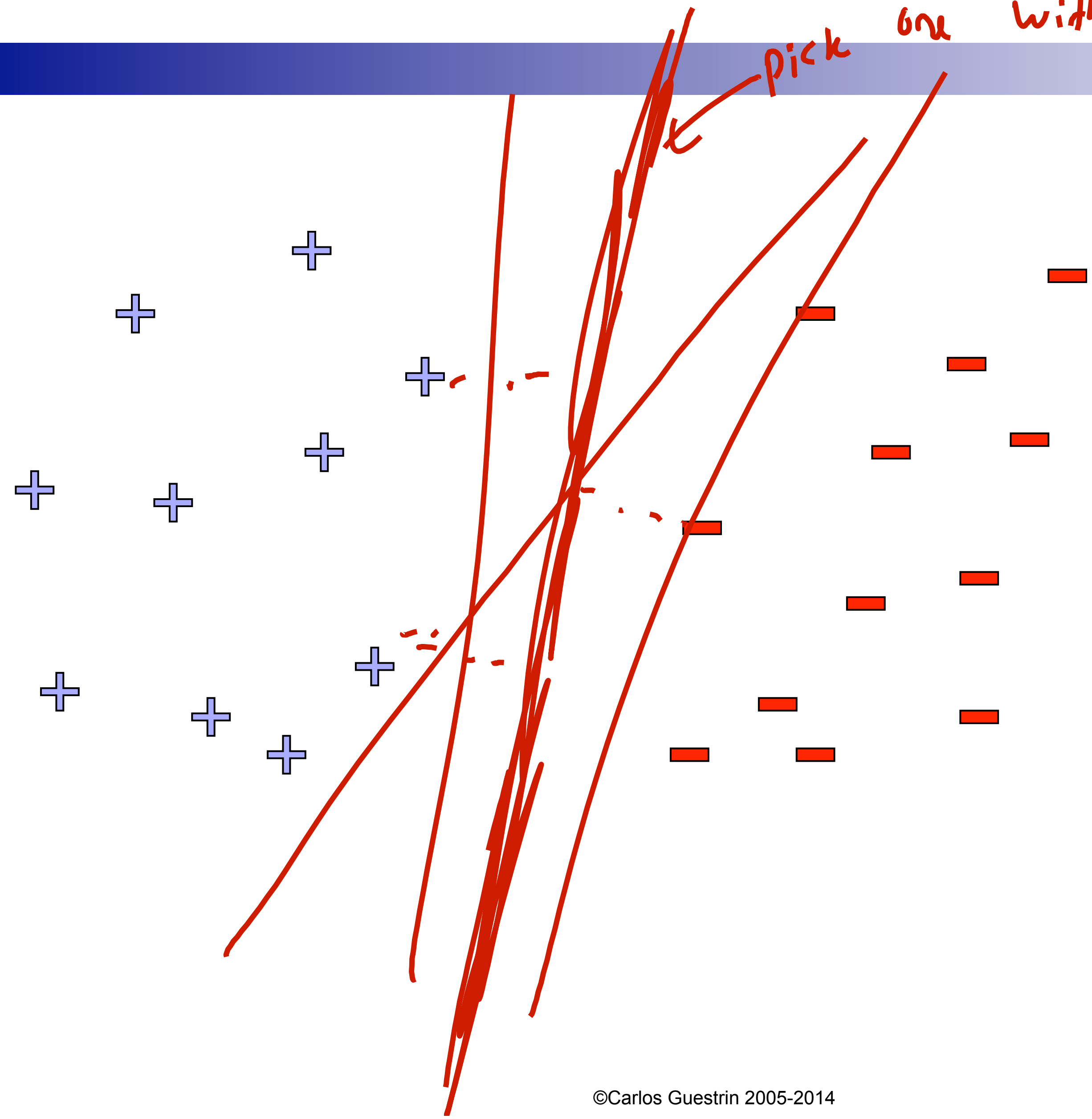
Support Vector Machines

- One of the most effective classifiers to date!
- Popularized kernels

- There is a complicated derivation, but…
- Very simple based on what you've learned thus far!

Linear classifiers – Which line is better?

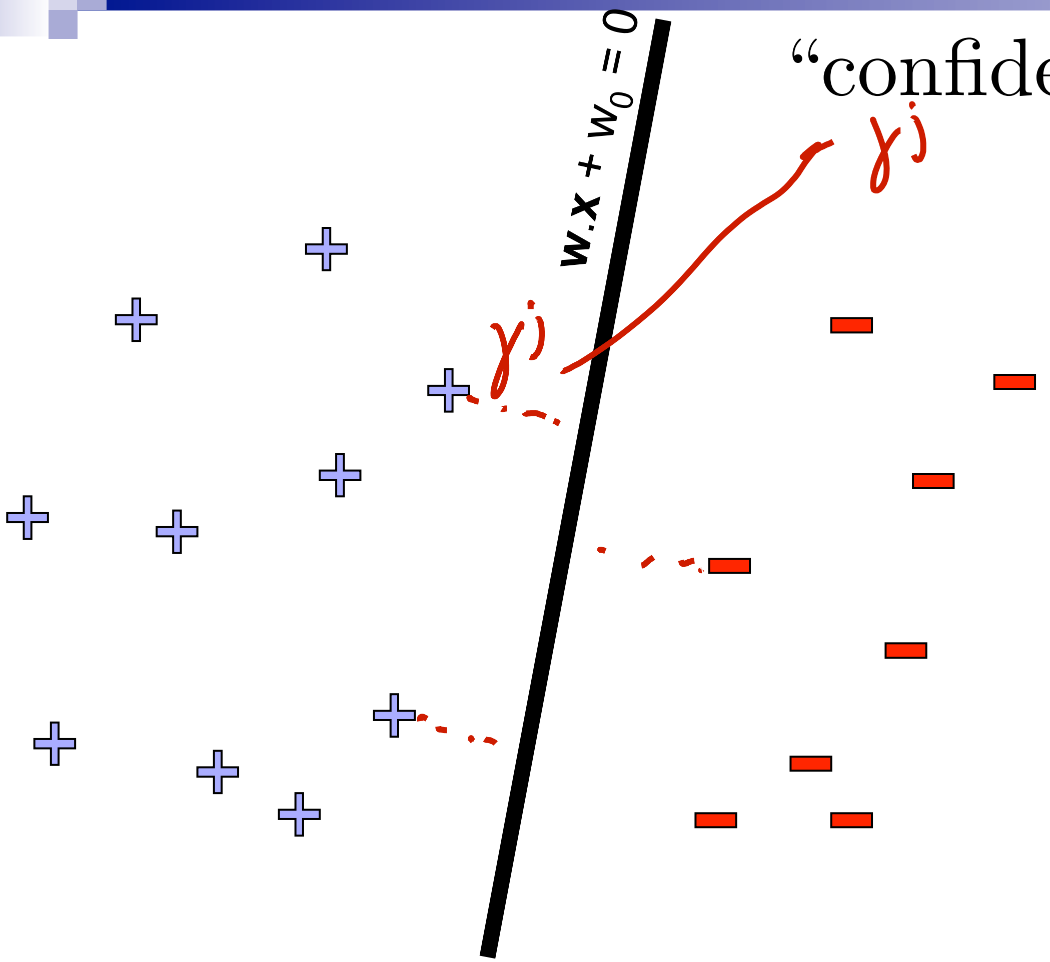
pick one with largest margin



Pick the one with the largest margin!

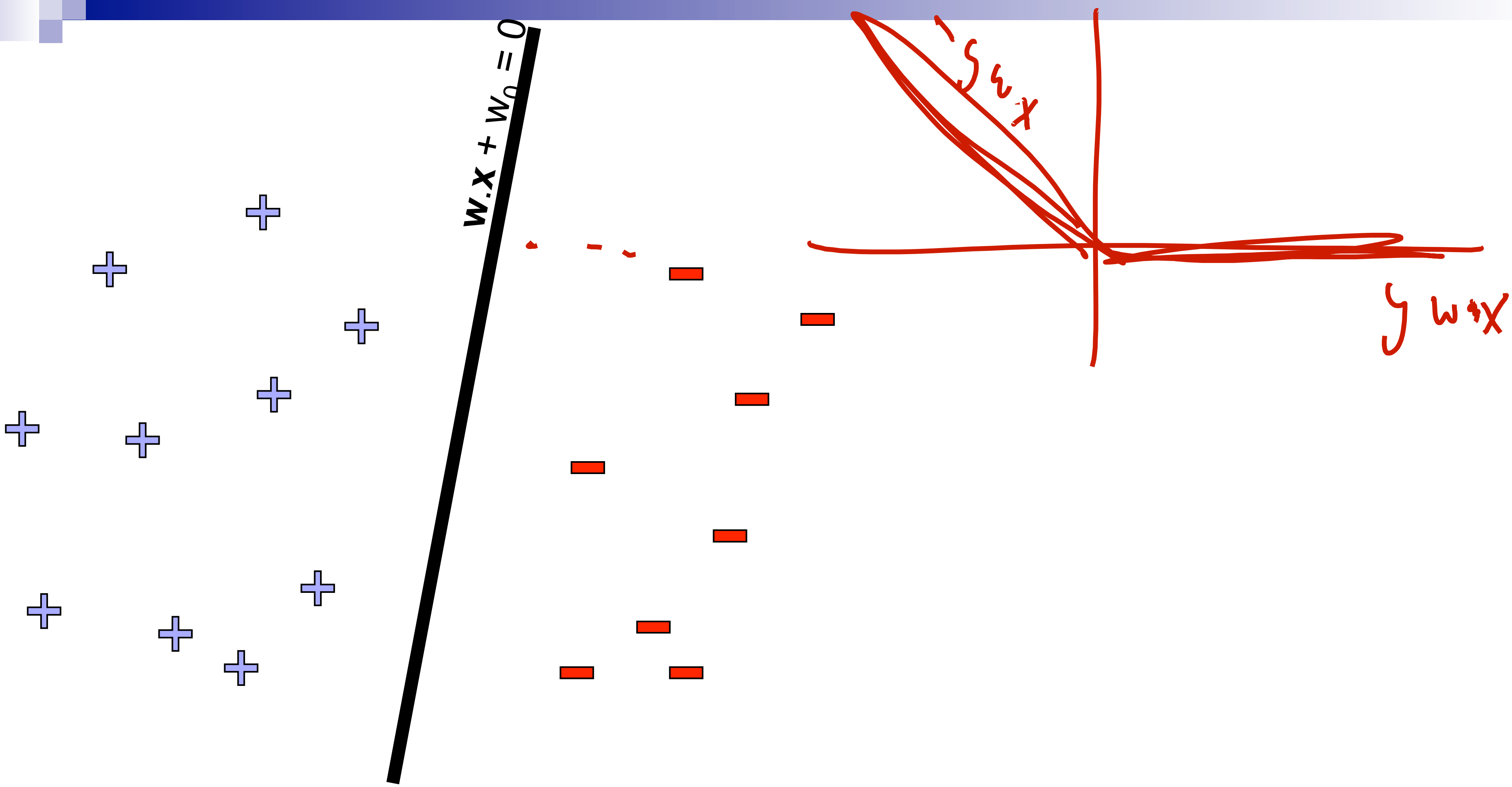
$$\text{“confidence”} = y^j(\mathbf{w} \cdot \mathbf{x}^j + w_0)$$

γ^j



maximize avg margin of confidence over data points

Maximize the margin

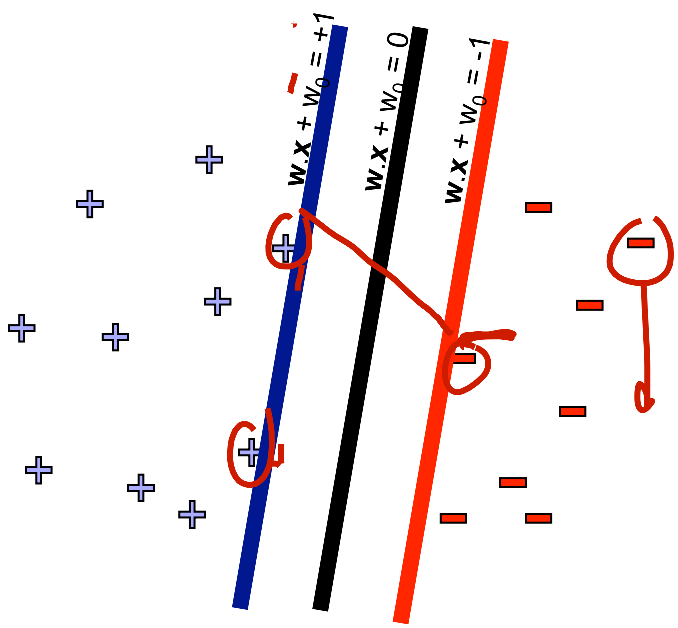


SVMs = Hinge Loss + L2 Regularization

$(a)_+ : \begin{cases} a & \text{if } a > 0 \\ 0 & \text{otherwise} \end{cases}$

- Maximizing Margin same as regularized hinge loss

- But, SVM "convention" is confidence has to be at least 1…



$$\ell(w,x) = \begin{cases} 0 & \text{, if } y\, w.x \geq 1 \\ 1 - y\, w.x & \text{otherwise} \end{cases}$$

$$= (1 - y\, w.x)_+$$

L2 Regularized Hinge Loss

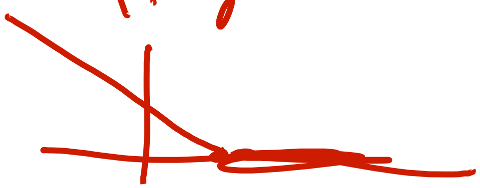
$\arg\min_w f(w) + \lambda g(w)$

$\equiv \arg\min_w \frac{1}{\lambda} f(w) + g(w)$

$C = \frac{1}{\lambda}$

- Final objective, adding regularization:

$\arg\min_w$ hinge



regularization

$$+ \frac{\lambda}{2} ||w||_2^2$$

- But, again, in SVMs, convention slightly different (but equivalent)

$$\arg\min_w \quad \frac{||\mathbf{w}||_2^2}{2} + C \sum_{j=1}^{N} \left(1 - y^j (\mathbf{w} \cdot \mathbf{x}^j + w_0)\right)_+$$

"penalty term" works like $\frac{1}{\lambda}$

SVMs for Non-Linearly Separable meet my friend the Perceptron…

- Perceptron was minimizing the hinge loss: like perceptron

$$\sum_{j=1}^{N} \left(-y^j(\mathbf{w}\cdot\mathbf{x}^j + w_0)\right)_+$$

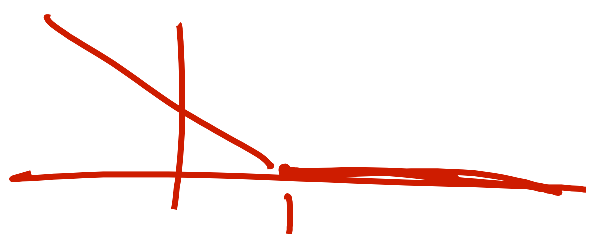
no regularization

- SVMs minimizes the regularized hinge loss!!

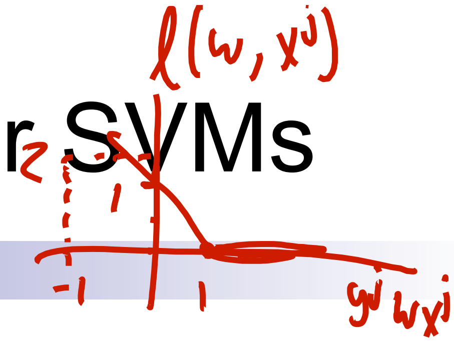
$$||\mathbf{w}||_2^2 + C\sum_{j=1}^{N} \left(1 - y^j(\mathbf{w}\cdot\mathbf{x}^j + w_0)\right)_+$$

2

regularization



Stochastic Gradient Descent for SVMs



- Perceptron minimization:

$$\sum_{j=1}^{N} \left(-y^j(\mathbf{w}\cdot\mathbf{x}^j + w_0)\right)_+$$

- SGD for Perceptron:

$$\mathbf{w}^{(t+1)} \leftarrow \mathbf{w}^{(t)} + \mathbb{1}\underbrace{\left[y^{(t)}(\mathbf{w}^{(t)}\cdot\mathbf{x}^{(t)}) \leq 0\right]}_{\text{mistake}} y^{(t)}\mathbf{x}^{(t)}$$

- SVMs minimization:

$$\frac{||\mathbf{w}||_2^2}{2} + C \sum_{j=1}^{N} \left(1 - y^j(\mathbf{w}\cdot\mathbf{x}^j + w_0)\right)_+$$

- SGD for SVMs:

for w_0, don't include this part

$$w^{(t+1)} \leftarrow w^{(t)} - \eta_t \left[w_i - C\, \underbrace{\mathbb{1}\left(y^{(t)}(w^{(t)}\cdot x^{(t)}) \leq 1\right)}_{\text{mistake}} y^{(t)} x^{(t)} \right]$$

What you need to know

- Maximizing margin
- Derivation of SVM formulation
- Non-linearly separable case
  - Hinge loss
  - A.K.A. adding slack variables
- SVMs = Perceptron + L2 regularization
- Can also use kernels with SVMs
- Can optimize SVMs with SGD
  - Many other approaches possible

Boosting

Machine Learning – CSEP546

Carlos Guestrin

University of Washington

January 27, 2014

Fighting the bias-variance tradeoff

- **Simple (a.k.a. weak) learners are good**
  - e.g., naïve Bayes, logistic regression, decision stumps (or shallow decision trees) thresholding
  - Low variance, don't usually overfit too badly
- **Simple (a.k.a. weak) learners are bad**
  - High bias, can't solve hard learning problems

- Can we make weak learners always good???
  - **No!!!**
  - **But often yes…**

The Simplest Weak Learner: Thresholding, a.k.a. Decision Stumps

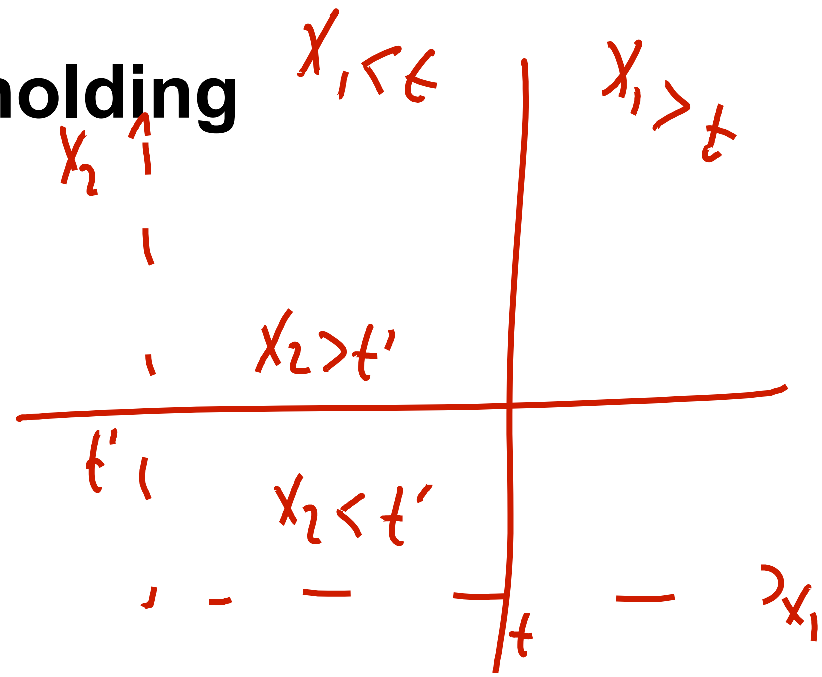
- **Learn**: h:**X** $\mapsto$ Y
  - **X** – features
  - Y – target classes

$X = (GPA, grade, \ldots$

$Y = \{hired, not hired)$

- **Simplest case: Thresholding**

$$h(x) = \begin{cases} \text{hired if } GPA > 3.9 \\ \text{not hired, otherwise} \end{cases}$$



Voting (Ensemble Methods)

- Instead of learning a single (weak) classifier, learn **many weak classifiers** that are **good at different parts of the input space** $h_i : X \to Y \in \{-1, +1\}$
- **Output class:** (Weighted) vote of each classifier
  - Classifiers that are most "sure" will vote with more conviction
  - Classifiers will be most "sure" about a particular part of the space
  - On average, do better than single classifier!

$$H(x) = \mathrm{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$$

classifier learned at iteration t

the weight of classifier

e.g. $h_t(x)$: GPA > 3.9?

- **But how do you ???**
  - force classifiers to learn about different parts of the input space?
  - weigh the votes of different classifiers?