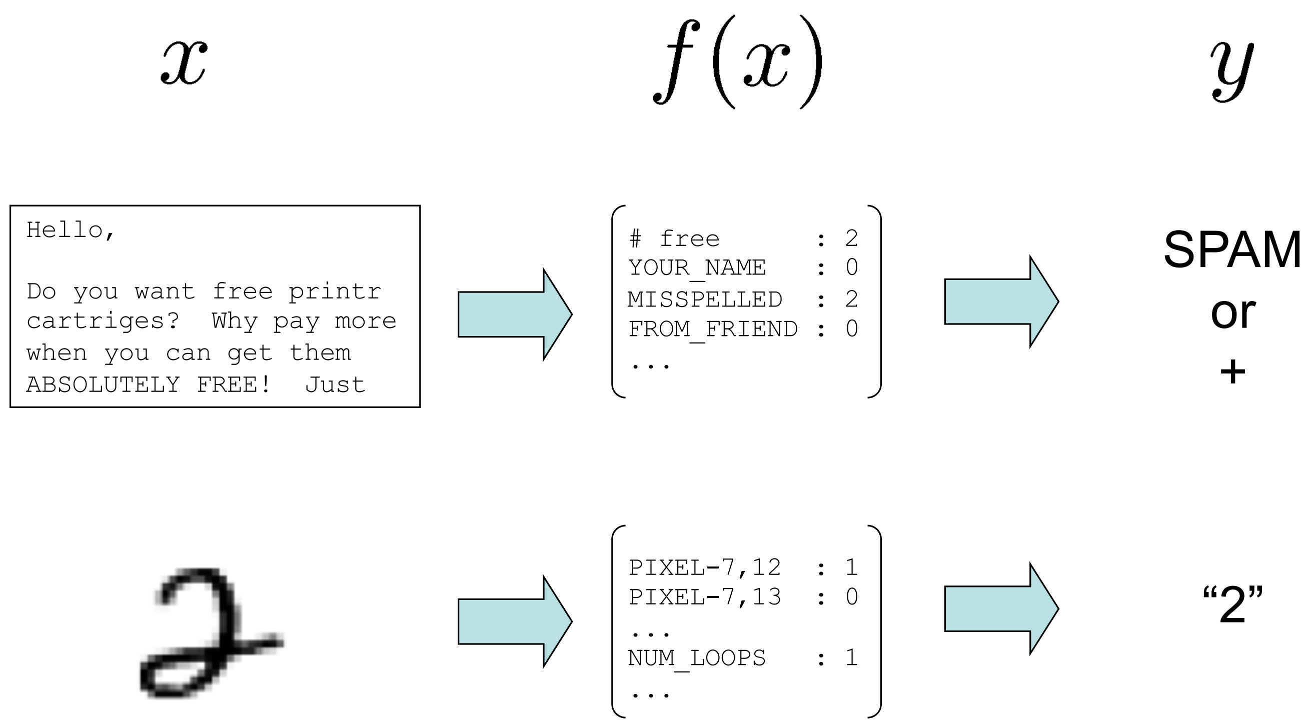
CS 188: Artificial Intelligence

Linear and Logistic Regression

Spring 2024

University of California, Berkeley

Classification with Feature Vectors



Regression with Feature Vectors

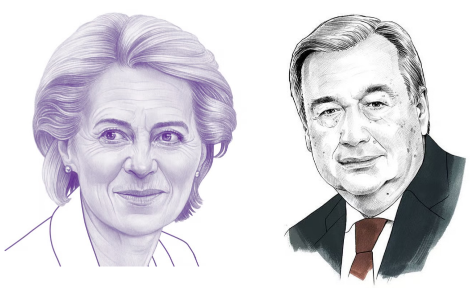
x $f(x)$ y

[Office space at 2024 Shattuck Ave]

```
CONST          : 1
Sq ft.    : 40000
Dist. BART: 0.1
# offices   : 16
# views       : 2
...
```

Market rent:

84000



```
CONST          : 1
PIXEL-7,12   : 1
PIXEL-7,13   : 0
...
OUTSIDE?     : 1
...
```

Compatibility:

8

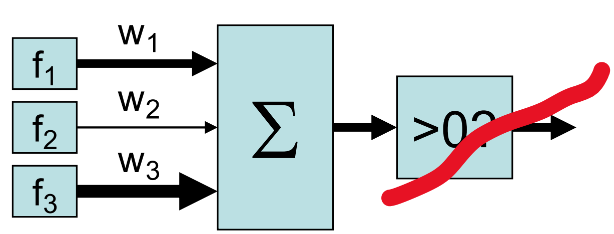
Linear ~~Classifiers~~ Regression

- Inputs are feature values
- Each feature has a weight
- Sum is the ~~activation~~ prediction

$$h_w \; \text{~~activation~~}_w(x) = \sum_i w_i \cdot f_i(x) = w \cdot f(x)$$

- ~~If the activation is:~~
  - ~~Positive, output +1~~
  - ~~Negative, output -1~~

  Output h_w



Weights

Dot product $w \cdot f$ *gives the prediction*

$$w \cdot f(x_1)$$

```
w:
CONST     : 5000
Sq ft.  :   0.8
Dist. BART: 100
# offices : 300
# views   : 1000
...

f(x1):
CONST        : 1
Sq ft.   : 40000
Dist. BART: 0.1
# offices  : 16
# views      : 2
...
```

$$w \cdot f(x_2)$$

```
w:
CONST     : 5000
Sq ft.  :   0.8
Dist. BART: 100
# offices : 300
# views   : 1000
...

f(x2):
CONST        : 1
Sq ft.   : 50000
Dist. BART: 0.2
# offices  :  4
# views      : 0
...
```

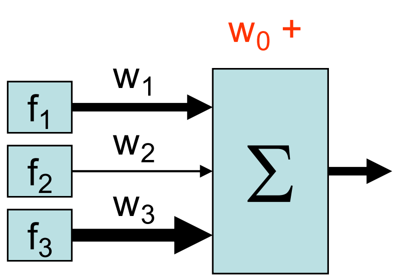
Which weight makes the least sense for predicting office rent?

Linear Regression

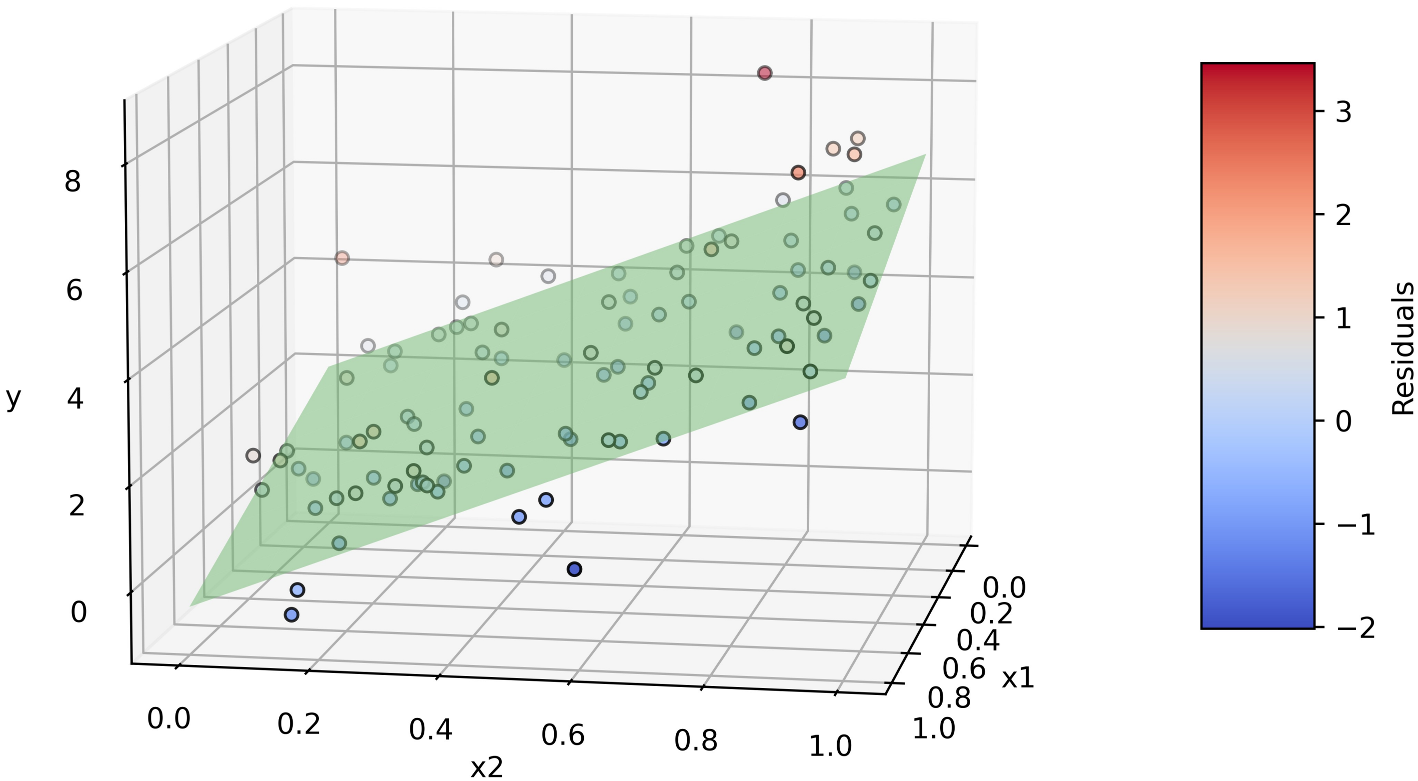
- Inputs are feature values
- Each feature has a weight
- Sum is the prediction

Either make sure one of the features is a constant *or* add this w_0 to the equation (equivalent)

$$h_w(x) = \sum_i w_i \cdot f_i(x) = w \cdot f(x) + w_0$$



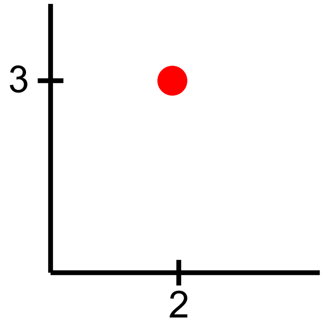
Linear Regression with 2d Feature Vector



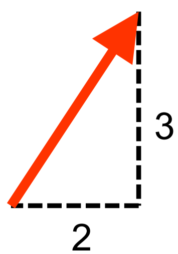
Code credit: Claude3

Review: Vectors

- A tuple like (2,3) can be interpreted two different ways:



A **point** on a coordinate grid

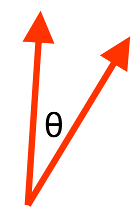


A **vector** in space. Notice we are not on a coordinate grid.

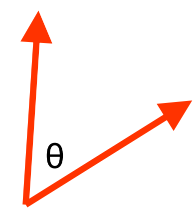
- A tuple with more elements like (2, 7, -3, 6) is a point or vector in higher-dimensional space (hard to visualize)

Review: Vectors

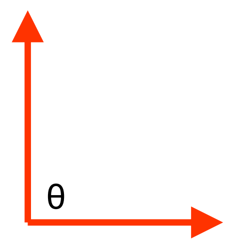
- Definition of dot product:
  - $a \cdot b = \Sigma_i\, a_i\, b_i = |a|\ |b| \cos(\theta)$
  - θ is the angle between the vectors a and b
- Consequences of this definition:
  - Vectors closer together
    = "similar" vectors
    = smaller angle θ between vectors
    = larger (more positive) dot product
  - If θ < 90°, then dot product is positive
  - If θ = 90°, then dot product is zero
  - If θ > 90°, then dot product is negative



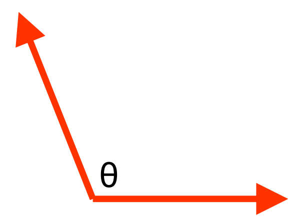
a · b large, positive



a · b small, positive

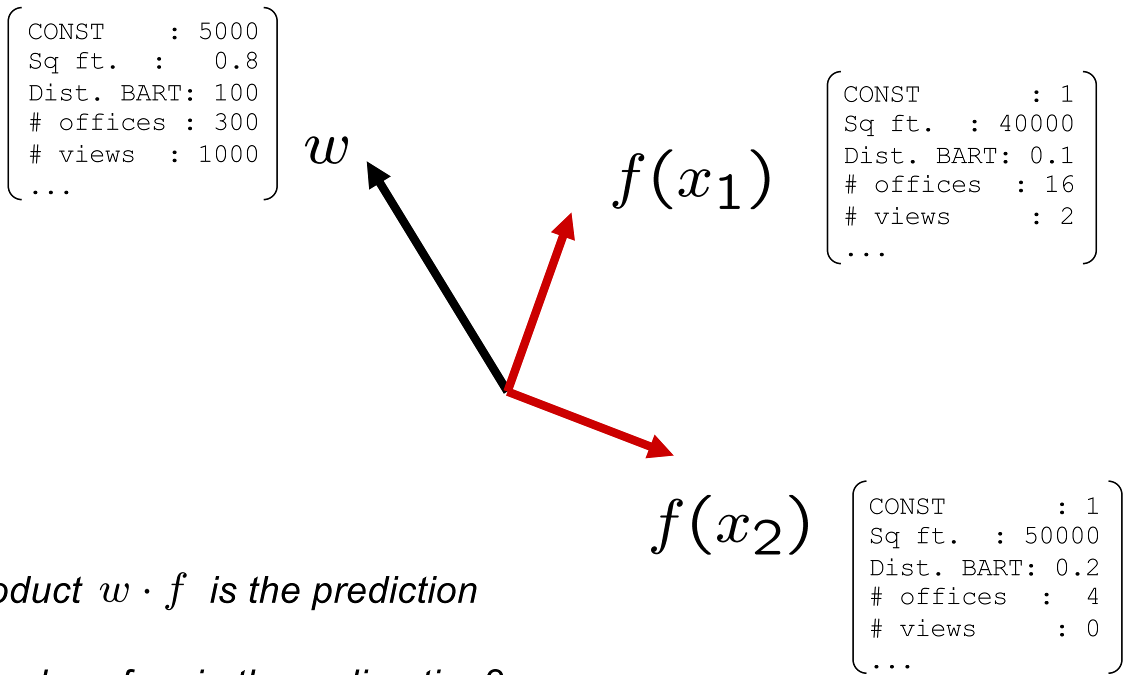


a · b zero



a · b negative

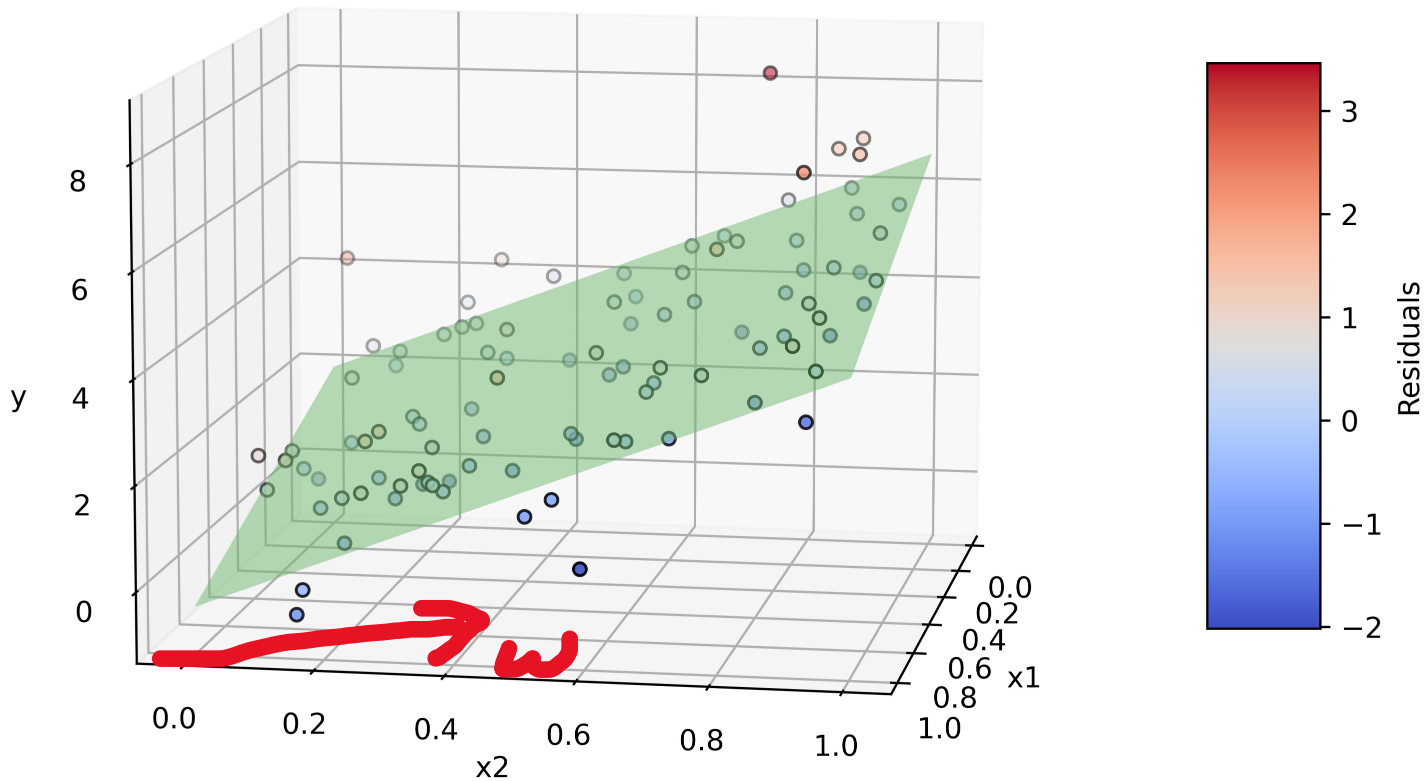
Weights



Dot product $w \cdot f$ *is the prediction*

How far does f go in the w direction?

Linear Regression with 2d Feature Vector



w points in direction where best-fit plane is steepest

Code credit: Claude3

How to find the weights?

known   known   unknown

$$\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}, \quad \mathbf{X} = \begin{bmatrix} 1 & x_1^1 & \cdots & x_n^1 \\ 1 & x_1^2 & \cdots & x_n^2 \\ \vdots & \vdots & \cdots & \vdots \\ 1 & x_1^N & \cdots & x_n^N \end{bmatrix}, \quad \mathbf{w} = \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_n \end{bmatrix}$$

data point →

↑ different values of same feature

How to find the weights?

data point

$$\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}, \quad \mathbf{X} = \begin{bmatrix} 1 & x_1^1 & \cdots & x_n^1 \\ 1 & x_1^2 & \cdots & x_n^2 \\ \vdots & \vdots & \cdots & \vdots \\ 1 & x_1^N & \cdots & x_n^N \end{bmatrix}, \quad \mathbf{w} = \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_n \end{bmatrix}$$

What does matrix product Xw look like? Vector like w and y

What is the entry in the first row and first (and only) column of Xw?

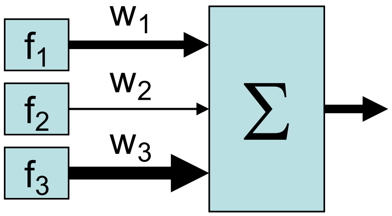
$w_0 + w_1 x_1^1 + \ldots + w_n x_n^1$

We want Xw to look like y

Linear Regression

- Inputs are feature values
- Each feature has a weight
- Sum is the prediction

$$h_w(x) = \sum_i w_i \cdot f_i(x) = w \cdot f(x)$$



Premise of linear regression

data point

$$\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}, \quad \mathbf{X} = \begin{bmatrix} 1 & x_1^1 & \cdots & x_n^1 \\ 1 & x_1^2 & \cdots & x_n^2 \\ \vdots & \vdots & \cdots & \vdots \\ 1 & x_1^N & \cdots & x_n^N \end{bmatrix}, \quad \mathbf{w} = \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_n \end{bmatrix}$$

For a proposed weight vector w, its badness is $|Xw - y|^2 / 2$

$|v|$ is the length of the vector; $|v|^2 = \Sigma_i v_i^2 = v^T v$

Loss

So ~~badness~~(w) = $\Sigma_i (h_w(x^i)w - y_i)^2 / 2$

Solving for w

data point

$$\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}, \quad \mathbf{X} = \begin{bmatrix} 1 & x_1^1 & \cdots & x_n^1 \\ 1 & x_1^2 & \cdots & x_n^2 \\ \vdots & \vdots & \cdots & \vdots \\ 1 & x_1^N & \cdots & x_n^N \end{bmatrix}, \quad \mathbf{w} = \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_n \end{bmatrix}$$

Find $\text{argmin}_w \ |Xw - y|^2 / 2$

$$\nabla_{\mathbf{w}} \frac{1}{2} (\mathbf{y} - \mathbf{X}\mathbf{w})^T (\mathbf{y} - \mathbf{X}\mathbf{w}) = 0$$

$$= \nabla_{\mathbf{w}} \frac{1}{2} \left(\mathbf{y}^T\mathbf{y} - \mathbf{y}^T\mathbf{X}\mathbf{w} - \mathbf{w}^T\mathbf{X}^T\mathbf{y} + \mathbf{w}^T\mathbf{X}^T\mathbf{X}\mathbf{w} \right)$$

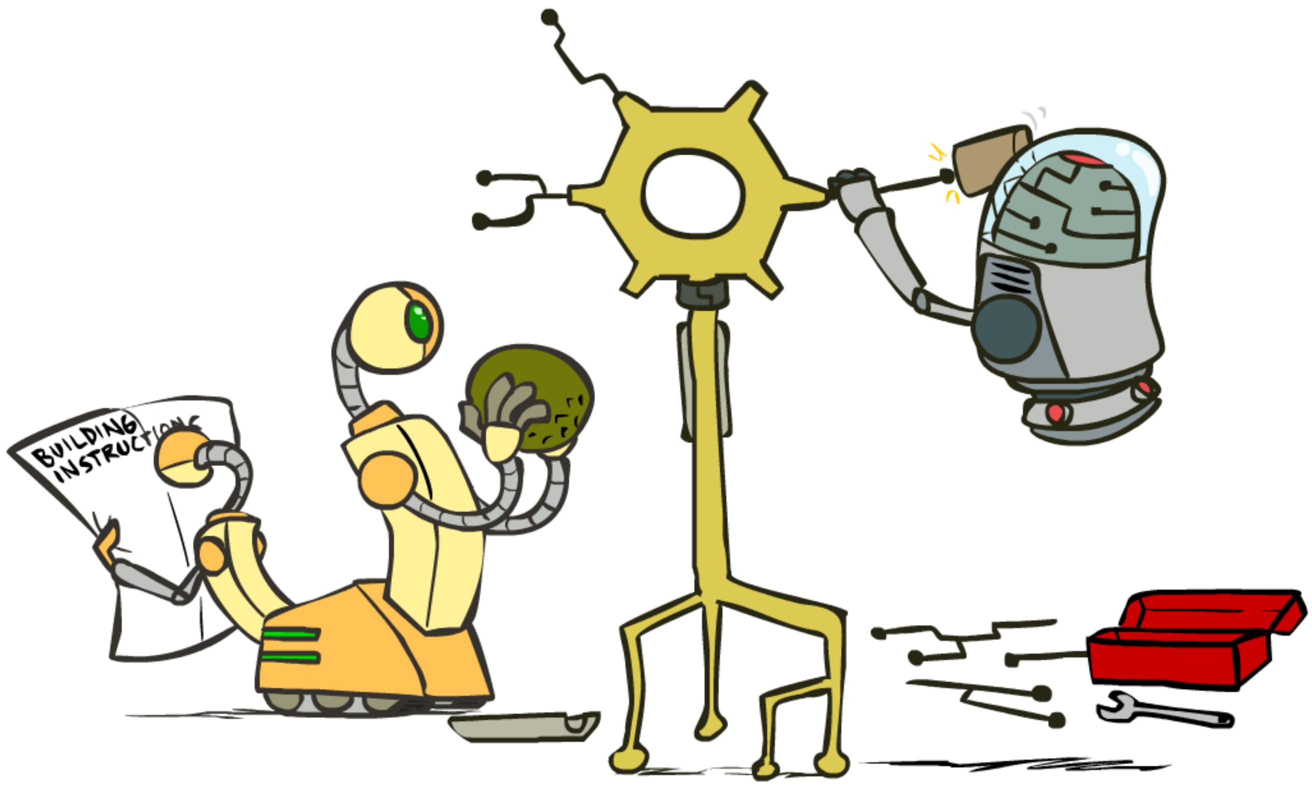
$$= \nabla_{\mathbf{w}} \frac{1}{2} \left(\mathbf{y}^T\mathbf{y} - 2\mathbf{w}^T\mathbf{X}^T\mathbf{y} + \mathbf{w}^T\mathbf{X}^T\mathbf{X}\mathbf{w} \right) = -\mathbf{X}^T\mathbf{y} + \mathbf{X}^T\mathbf{X}\mathbf{w}$$

$$\hat{\mathbf{w}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}.$$

If you ever actually need to do this sort of stuff:

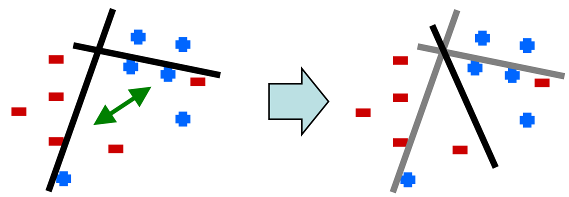
https://cs.nyu.edu/~roweis/notes/matrixid.pdf

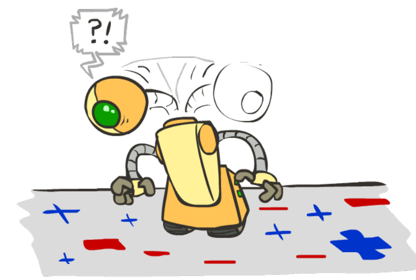
Back to Classification: Improving the Perceptron



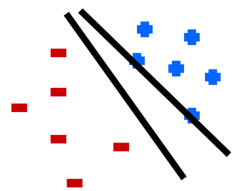
Problems with the Perceptron

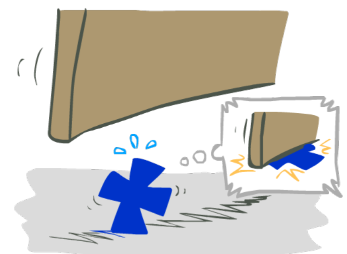
- Noise: if the data isn't separable, weights might thrash
  - Averaging weight vectors over time can help (averaged perceptron)



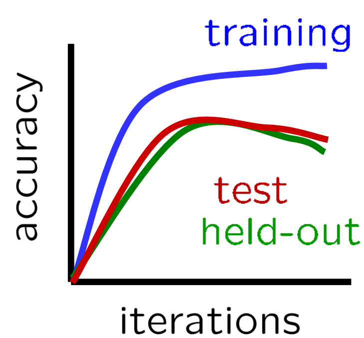


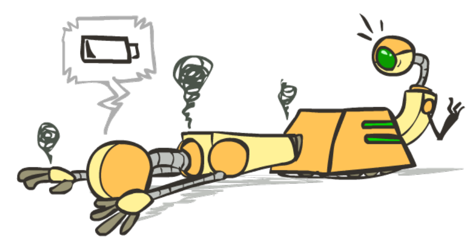
- Mediocre generalization: finds a "barely" separating solution





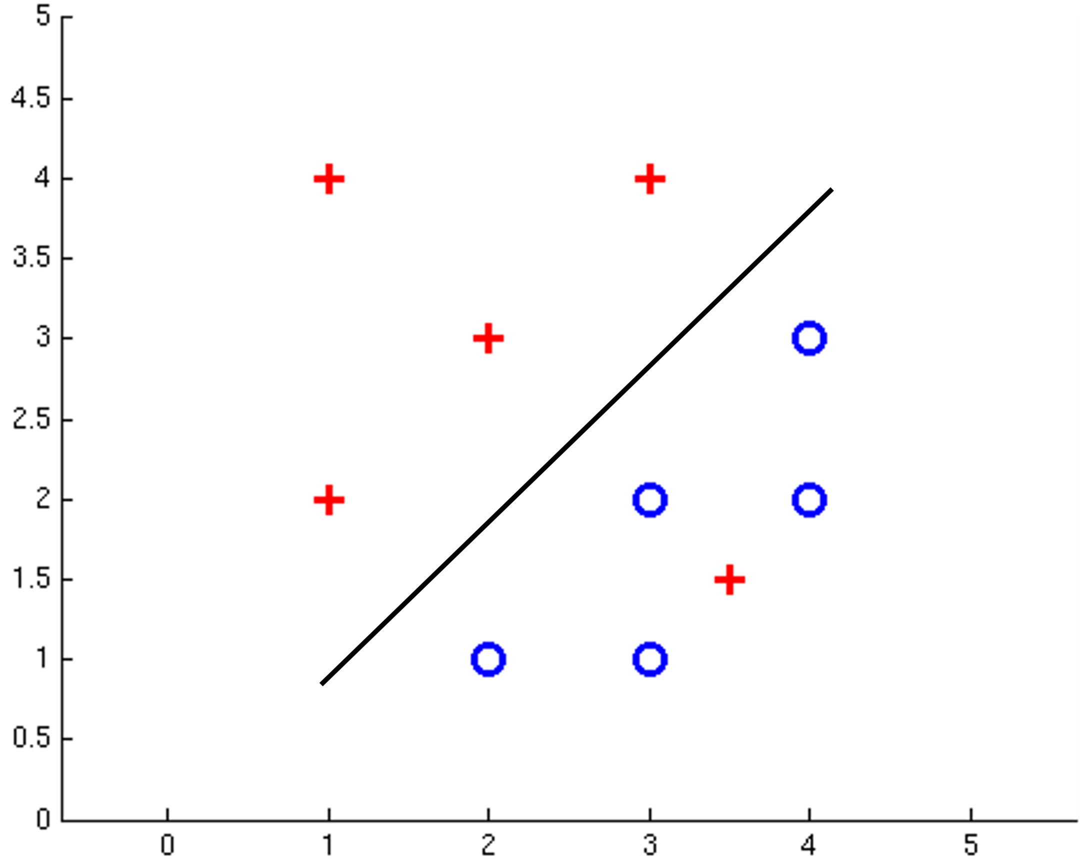
- Overtraining: test / held-out accuracy usually rises, then falls
  - Overtraining is a kind of overfitting



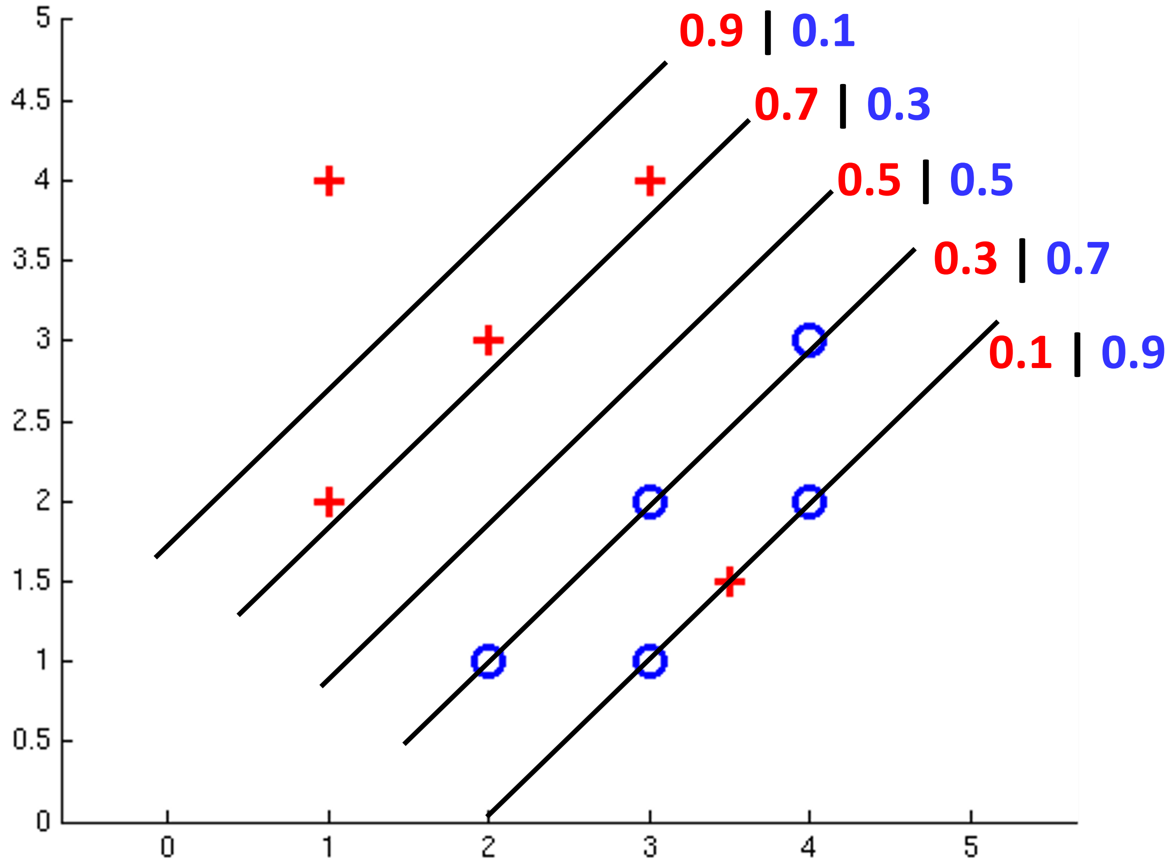


Non-Separable Case: Deterministic Decision

Even the best linear boundary makes at least one mistake



Non-Separable Case: Probabilistic Decision

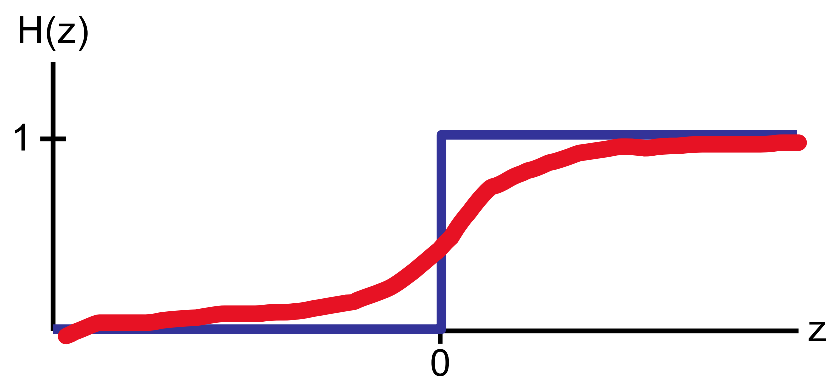


Perceptrons give deterministic decisions

- Perceptron scoring: $z = w \cdot f(x)$
- If $z = w \cdot f(x)$ positive → classifier says: 1.0 probability this is class +1
- If $z = w \cdot f(x)$ negative → classifier says: 0.0 probability this is class +1

- Step function

$$H(z) = \begin{cases} 1 & z > 0 \\ 0 & z \leq 0 \end{cases}$$



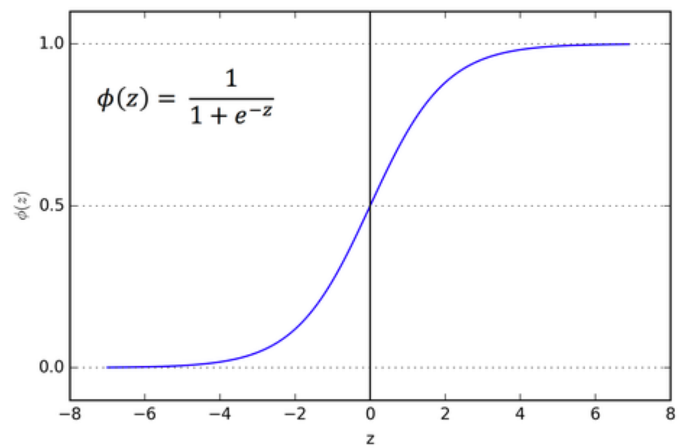
- z = output of perceptron
  H(z) = probability the class is +1, according to the classifier

How to get probabilistic decisions?

- Perceptron scoring: $z = w \cdot f(x)$
- If $z = w \cdot f(x)$ very positive → probability of class +1 should approach 1.0
- If $z = w \cdot f(x)$ very negative → probability of class +1 should approach 0.0

- Sigmoid function

$$\phi(z) = \frac{1}{1 + e^{-z}}$$



- z = output of perceptron
- $\phi(z)$ = probability the class is +1, according to the classifier

Probabilistic Decisions: Example

$$\frac{1}{1+e^{-wx}}$$

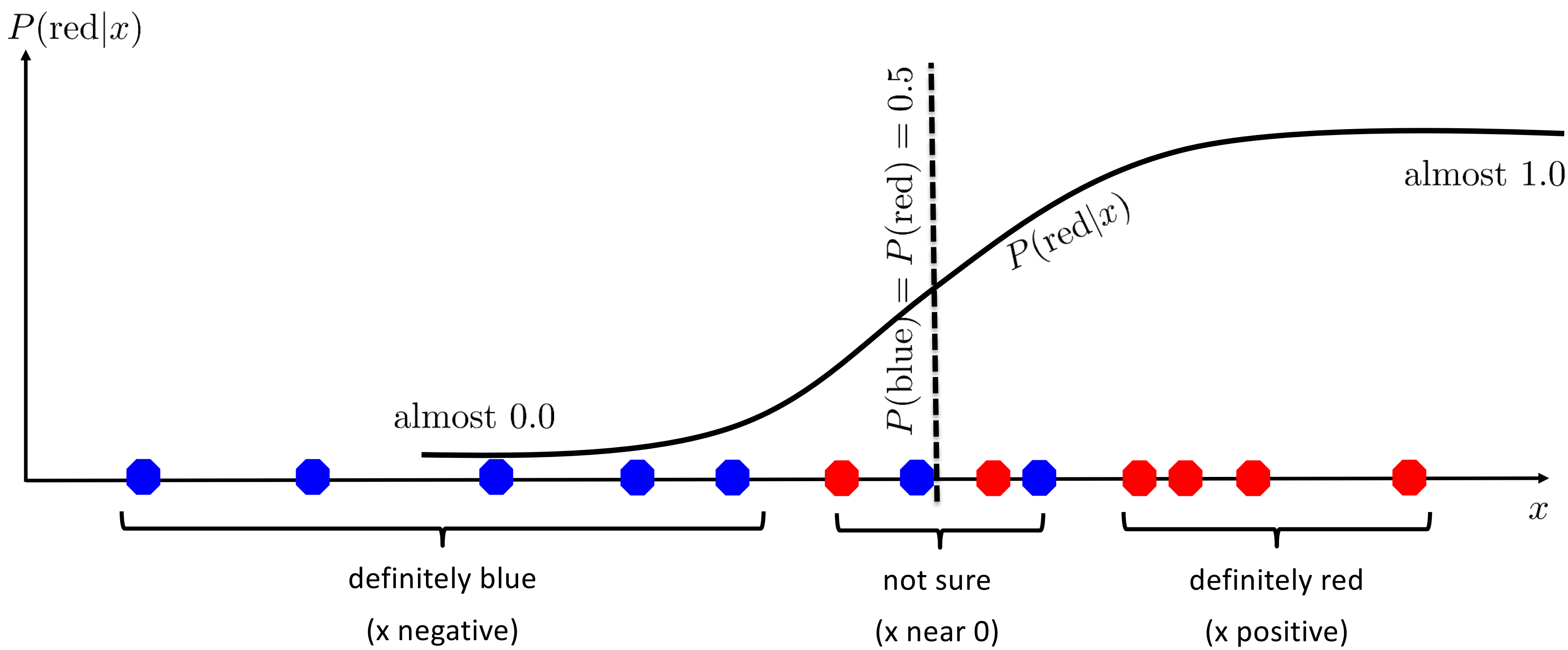
where w is some weight constant (vector) we have to learn, and wx is the dot product of w and x

- Suppose w = [–3, 4, 2] and x = [1, 2, 0]
- What label will be selected if we classify deterministically?
  - wx = –3+8+0 = 5
  - 5 is positive, so the classifier guesses the positive label
- What are the probabilities of each label if we classify probabilistically?
  - 1 / (1 + e^{-5}) = 0.9933 probability of positive label
  - 1 – 0.9933 = 0.0067 probability of negative label

A 1D Example

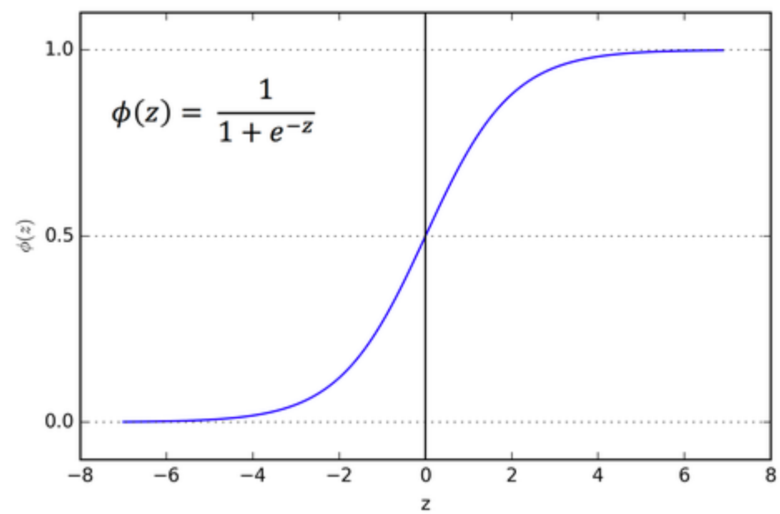
$$P(\text{red}|x) = \frac{1}{1 + e^{-wx}}$$

where w is some weight constant (1D vector) we have to learn



Where does the sigmoid function come from?

- Suppose we have two hypotheses:
  - A: P(heads) = 2/3
  - B: P(heads) = 1/3
- Each heads we see is a "bit" or factor of 2 of evidence for Hypothesis A
- Each tails we see is a "bit" of evidence for B
- If we have n more heads than tails:
  - A is 2^n times more likely than B
  - $P(A) = 2^n / (1 + 2^n)$
  - $= 1 / (1 + 2^{-n})$
  - … but we like e better than 2



Best w?

- Recall maximum likelihood estimation: Choose the w value that maximizes the probability of the observed (training) data

$$\begin{aligned}
\text{Likelihood} &= P(\text{training data}|w) \\
&= \prod_i P(\text{training datapoint } i \mid w) \\
&= \prod_i P(\text{point } x^{(i)} \text{ has label } y^{(i)}|w) \\
&= \prod_i P(y^{(i)}|x^{(i)}; w) \\
\text{Log Likelihood} &= \sum_i \log P(y^{(i)}|x^{(i)}; w)
\end{aligned}$$

Best w?

- Recall maximum likelihood estimation: Choose the w value that maximizes the probability of the observed (training) data

$$
\begin{aligned}
& P(\text{point } x^{(i)} \text{ has label } y^{(i)} = +1 \mid w) \\
= \; & P(y^{(i)} = +1 \mid x^{(i)}; w) \\
= \; & \frac{1}{1 + e^{-w \cdot x^{(i)}}}
\end{aligned}
$$

$$
\begin{aligned}
& P(\text{point } x^{(i)} \text{ has label } y^{(i)} = -1 \mid w) \\
= \; & P(y^{(i)} = -1 \mid x^{(i)}; w) \\
= \; & 1 - \frac{1}{1 + e^{-w \cdot x^{(i)}}}
\end{aligned}
$$

Best w?

- Maximum likelihood estimation:

$$\max_w \quad ll(w) = \max_w \quad \sum_i \log P(y^{(i)}|x^{(i)}; w)$$

with:

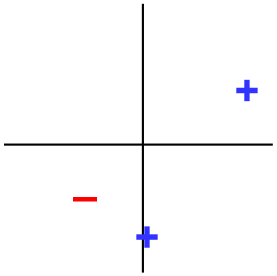
$$P(y^{(i)} = +1|x^{(i)}; w) = \frac{1}{1 + e^{-w \cdot f(x^{(i)})}}$$

$$P(y^{(i)} = -1|x^{(i)}; w) = 1 - \frac{1}{1 + e^{-w \cdot f(x^{(i)})}}$$

That's Logistic Regression Loss(w) = -log likelihood(w)

Logistic Regression Example

- What function are we trying to maximize for this training data?
  - Data point [2, 1] is class +1
  - Data point [0, –2] is class +1
  - Data point [–1, –1] is class –1



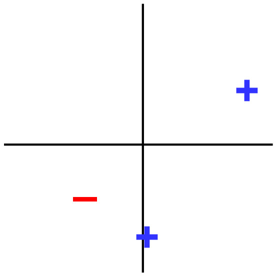
$$\max_w \; ll(w) = \max_w \; \sum_i \log P(y^{(i)}|x^{(i)};w)$$

$$P(y^{(i)} = +1|x^{(i)};w) = \frac{1}{1+e^{-w\cdot f(x^{(i)})}}$$

$$P(y^{(i)} = -1|x^{(i)};w) = 1 - \frac{1}{1+e^{-w\cdot f(x^{(i)})}}$$

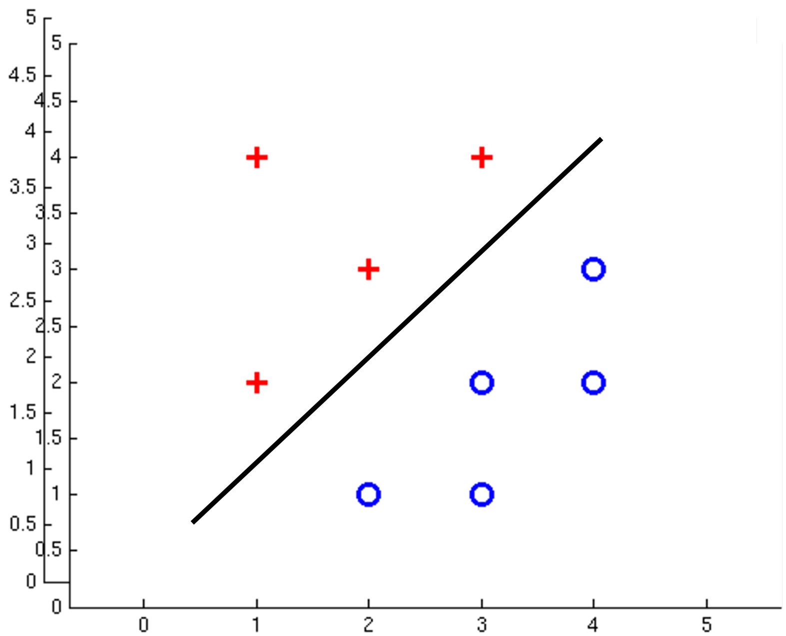
Logistic Regression Example

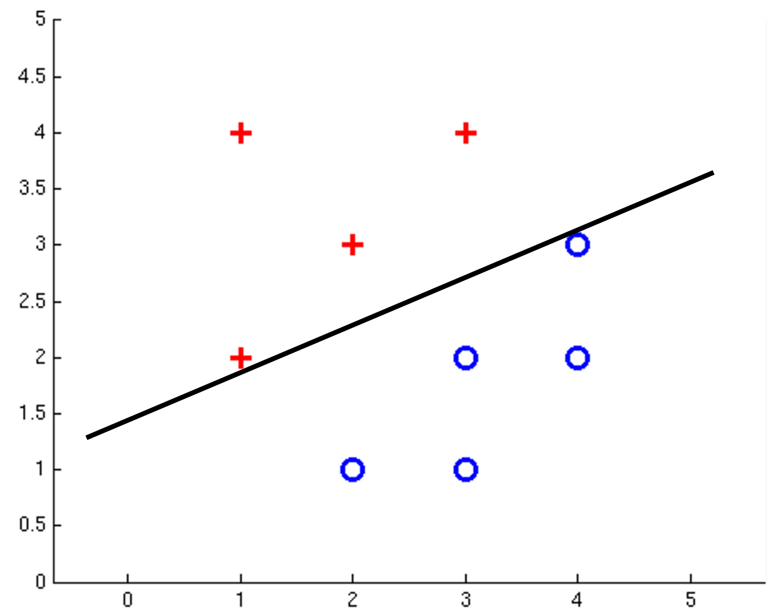
- What function are we trying to maximize for this training data?
  - Data point [2, 1] is class +1
  - Data point [0, –2] is class +1
  - Data point [–1, –1] is class –1



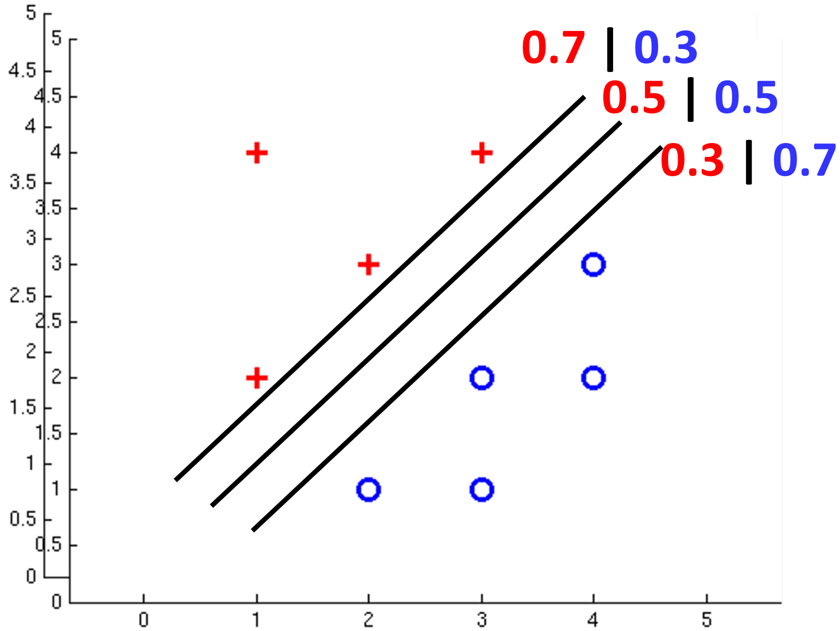
$$\operatorname*{argmax}_{w} \left[\log\left(\frac{1}{1+e^{-(2w_1+w_2)}}\right) + \log\left(\frac{1}{1+e^{-(-2w_2)}}\right) + \log\left(1 - \frac{1}{1+e^{-(-w_1-w_2)}}\right) \right]$$

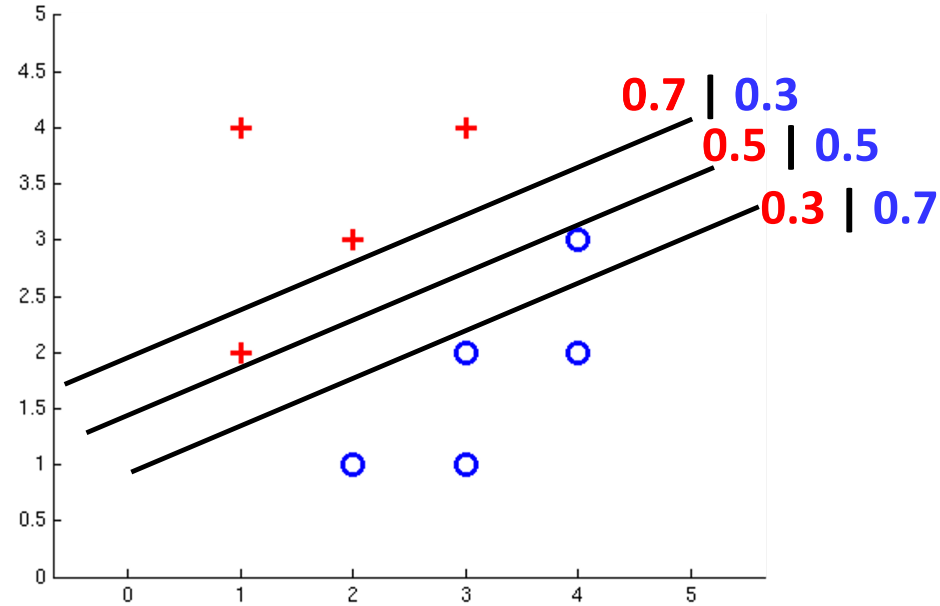
Separable Case: Deterministic Decision – Many Options





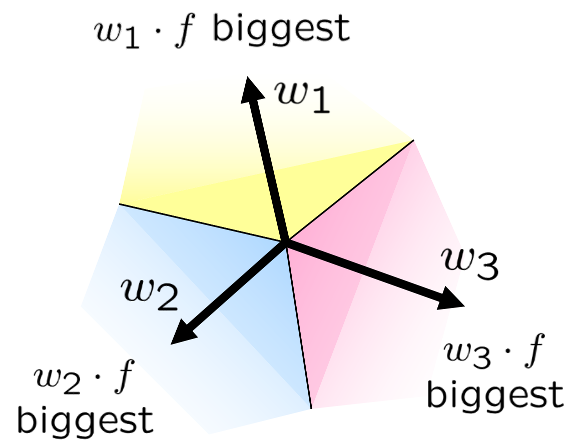
Separable Case: Probabilistic Decision – Clear Preference





Multiclass Logistic Regression

- Recall Perceptron:
  - A weight vector for each class: w_y
  - Score (activation) of a class y: $w_y \cdot f(x)$
  - Prediction highest score wins $y = \arg\max_y \; w_y \cdot f(x)$



- How to make the scores into probabilities?

$$\underbrace{z_1, z_2, z_3}_{\text{original activations}} \rightarrow \underbrace{\frac{e^{z_1}}{e^{z_1}+e^{z_2}+e^{z_3}}, \frac{e^{z_2}}{e^{z_1}+e^{z_2}+e^{z_3}}, \frac{e^{z_3}}{e^{z_1}+e^{z_2}+e^{z_3}}}_{\text{softmax activations}}$$

Multi-Class Probabilistic Decisions: Example

$$z_1, z_2, z_3 \rightarrow \frac{e^{z_1}}{e^{z_1} + e^{z_2} + e^{z_3}}, \frac{e^{z_2}}{e^{z_1} + e^{z_2} + e^{z_3}}, \frac{e^{z_3}}{e^{z_1} + e^{z_2} + e^{z_3}}$$

- Suppose $w_1 = [-3, 4, 2]$, $w_2 = [2, 2, 7]$, $w_3 = [0, -1, 0]$, and $x = [1, 2, 0]$
- What label will be selected if we classify deterministically?
  - $w_1 \cdot x = 5$, and $w_2 \cdot x = 6$, and $w_3 \cdot x = -2$
  - $w_2 \cdot x$ has the highest score, so the classifier guesses class 2
- What are the probabilities of each label if we classify probabilistically?
  - Probability of class 1: $e^5 / (e^5 + e^6 + e^{-2}) = 0.2689$
  - Probability of class 2: $e^6 / (e^5 + e^6 + e^{-2}) = 0.7310$
  - Probability of class 3: $e^{-2} / (e^5 + e^6 + e^{-2}) = 0.0002$

Best w?

- Recall maximum likelihood estimation: Choose the w value that maximizes the probability of the observed (training) data

$$
\begin{aligned}
\text{Likelihood} &= P(\text{training data}|w) \\
&= \prod_i P(\text{training datapoint } i \mid w) \\
&= \prod_i P(\text{point } x^{(i)} \text{ has label } y^{(i)}|w) \\
&= \prod_i P(y^{(i)}|x^{(i)}; w) \\
\text{Log Likelihood} &= \sum_i \log P(y^{(i)}|x^{(i)}; w)
\end{aligned}
$$

Best w?

- Maximum likelihood estimation:

$$\max_{w} \quad ll(w) = \max_{w} \quad \sum_{i} \log P(y^{(i)}|x^{(i)};w)$$

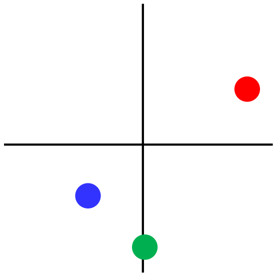
with:

$$P(y^{(i)}|x^{(i)};w) = \frac{e^{w_{y^{(i)}} \cdot f(x^{(i)})}}{\sum_{y} e^{w_y \cdot f(x^{(i)})}}$$

= Multi-Class Logistic Regression

Multi-Class Logistic Regression Example

- What function are we trying to maximize for this training data?
  - Data point [2, 1] is class Red
  - Data point [0, –2] is class Green
  - Data point [–1, –1] is class Blue

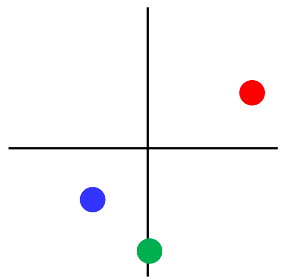


$$\max_w \quad ll(w) = \max_w \quad \sum_i \log P(y^{(i)}|x^{(i)};w)$$

$$P(y^{(i)}|x^{(i)};w) = \frac{e^{w_{y^{(i)}} \cdot f(x^{(i)})}}{\sum_y e^{w_y \cdot f(x^{(i)})}}$$

Multi-Class Logistic Regression Example

- What function are we trying to maximize for this training data?
  - Data point [2, 1] is class Red
  - Data point [0, –2] is class Green
  - Data point [–1, –1] is class Blue



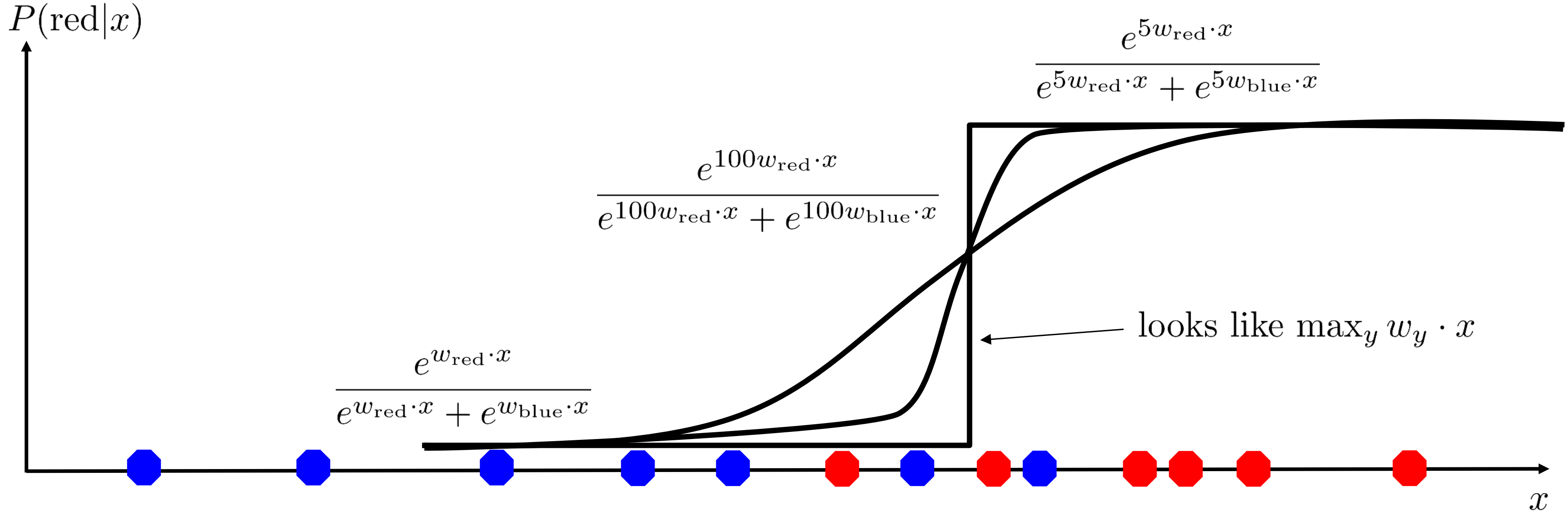
$$\underset{w}{\operatorname{argmax}} \left[\begin{array}{c} \log\left(\frac{e^{2w_1+w_2}}{e^{2w_1+w_2}+e^{2w_1+w_2}+e^{2w_1+w_2}}\right) \\ +\log\left(\frac{e^{-2w_2}}{e^{-2w_2}+e^{-2w_2}+e^{-2w_2}}\right) \\ +\log\left(\frac{e^{-w_1-w_2}}{e^{-w_1-w_2}+e^{-w_1-w_2}+e^{-w_1-w_2}}\right) \end{array} \right]$$

Log probability of [2, 1] being red

Log probability of [0,–2] being green

Log probability of [–1, –1] being blue

Softmax with Different Bases



$$P(\text{red}|x) = \frac{e^{w_{\text{red}}\cdot x}}{e^{w_{\text{red}}\cdot x} + e^{w_{\text{blue}}\cdot x}}$$

Softmax and Sigmoid

- Binary perceptron is a special case of multi-class perceptron
  - Multi-class: Compute $w_y \cdot f(x)$ for each class y, pick class with the highest activation
  - Binary case:
    Let the weight vector of +1 be w (which we learn).
    Let the weight vector of -1 always be 0 (constant).
  - Binary classification as a multi-class problem:
    Activation of negative class is always 0.
    If w · f is positive, then activation of +1 (w · f) is higher than -1 (0).
    If w · f is negative, then activation of -1 (0) is higher than +1 (w · f).

Softmax

$$P(\text{red}|x) = \frac{e^{w_{\text{red}} \cdot x}}{e^{w_{\text{red}} \cdot x} + e^{w_{\text{blue}} \cdot x}}$$

with $w_{red} = 0$ becomes:

Sigmoid

$$P(\text{red}|x) = \frac{1}{1 + e^{-wx}}$$

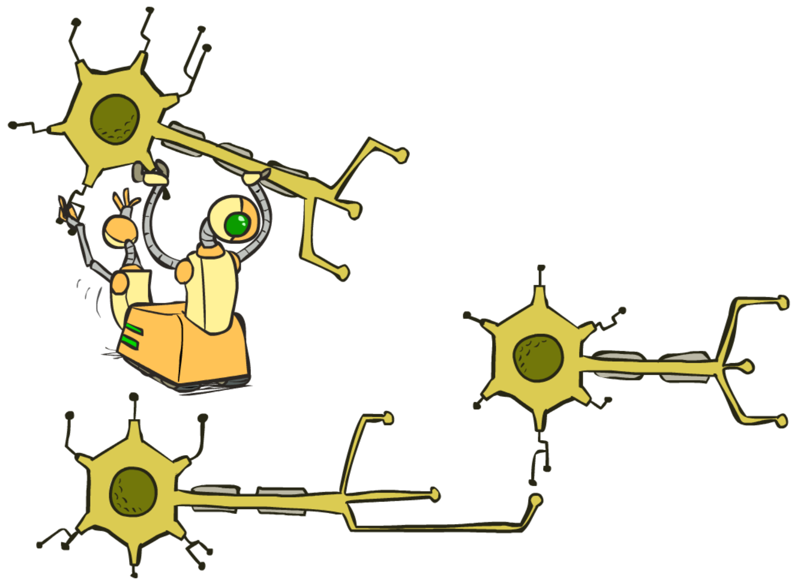
Next Up

- Optimization
  - i.e., how do we solve:

$$\max_w \quad ll(w) = \max_w \quad \sum_i \log P(y^{(i)} | x^{(i)}; w)$$

CS 188: Artificial Intelligence

Optimization



Spring 2024 --- University of California, Berkeley

[These slides were created by Dan Klein and Pieter Abbeel for CS188 Intro to AI at UC Berkeley. All CS188 materials are available at http://ai.berkeley.edu.]

Review: Derivatives and Gradients

- What is the derivative of the function $g(x) = x^2 + 3$?

$$\frac{dg}{dx} = 2x$$

- What is the derivative of g(x) at x=5?

$$\frac{dg}{dx}|_{x=5} = 10$$

Review: Derivatives and Gradients

- What is the gradient of the function $g(x, y) = x^2 y$?
  - Recall: Gradient is a vector of partial derivatives with respect to each variable

$$\nabla g = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \\ \frac{\partial f}{\partial y} \end{bmatrix} = \begin{bmatrix} 2xy \\ \\ x^2 \end{bmatrix}$$

- What is the derivative of g(x, y) at x=0.5, y=0.5?

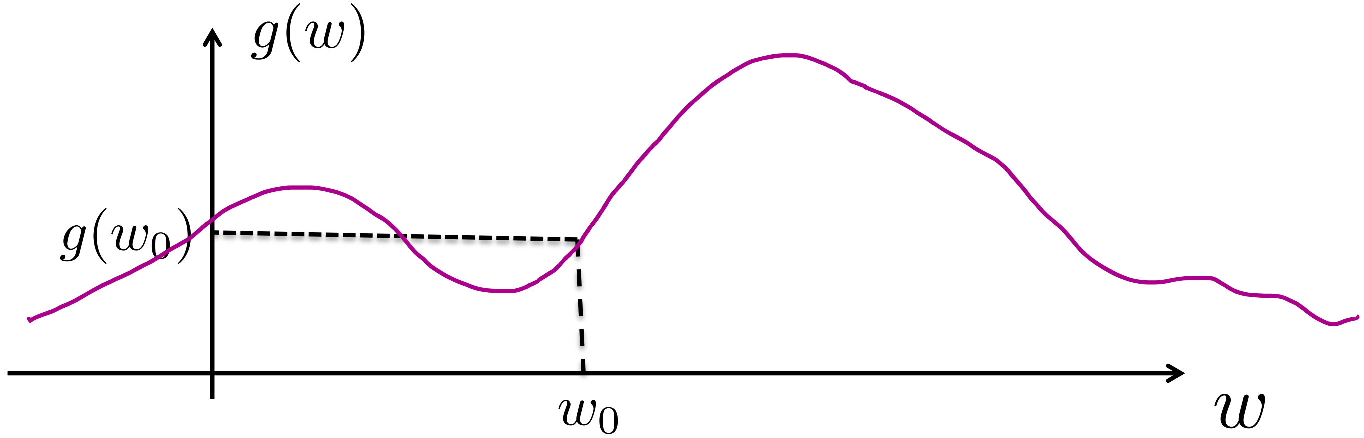
$$\nabla g|_{x=0.5, y=0.5} = \begin{bmatrix} 2(0.5)(0.5) \\ (0.5^2) \end{bmatrix} = \begin{bmatrix} 0.5 \\ 0.25 \end{bmatrix}$$

Hill Climbing

- Recall from local search: simple, general idea
  - Start wherever
  - Repeat: move to the best neighboring state
  - If no neighbors better than current, quit

- What's particularly tricky when hill-climbing for multiclass logistic regression?
  - Optimization over a continuous space
    - Infinitely many neighbors!
    - How to do this efficiently?

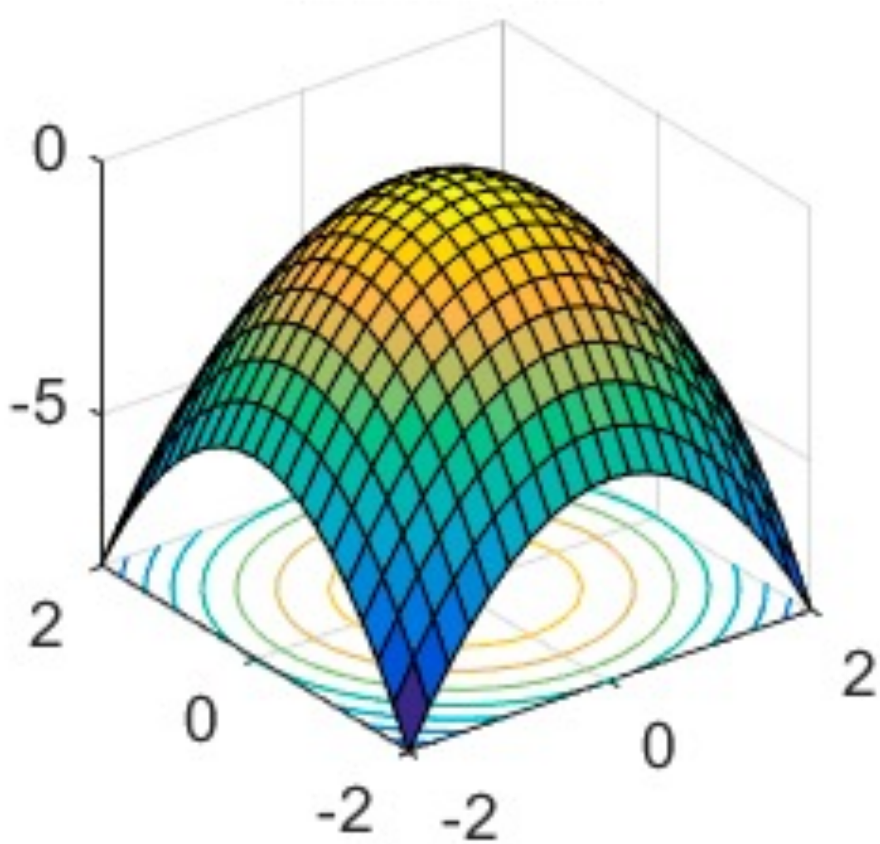
1-D Optimization



- Could evaluate $g(w_0 + h)$ and $g(w_0 - h)$
  - Then step in best direction

- Or, evaluate derivative: $\frac{\partial g(w_0)}{\partial w} = \lim_{h \to 0} \frac{g(w_0 + h) - g(w_0 - h)}{2h}$
  - Tells which direction to step into

2-D Optimization



Source: offconvex.org

Gradient Ascent

- Perform update in uphill direction for each coordinate
- The steeper the slope (i.e. the higher the derivative) the bigger the step for that coordinate
- E.g., consider: $g(w_1, w_2)$

- Updates:

$$w_1 \leftarrow w_1 + \alpha * \frac{\partial g}{\partial w_1}(w_1, w_2)$$

$$w_2 \leftarrow w_2 + \alpha * \frac{\partial g}{\partial w_2}(w_1, w_2)$$

- Updates in vector notation:

$$w \leftarrow w + \alpha * \nabla_w g(w)$$

with: $\nabla_w g(w) = \begin{bmatrix} \frac{\partial g}{\partial w_1}(w) \\ \frac{\partial g}{\partial w_2}(w) \end{bmatrix}$ **= gradient**

Gradient Ascent

- Idea:
  - Start somewhere
  - Repeat: Take a step in the gradient direction

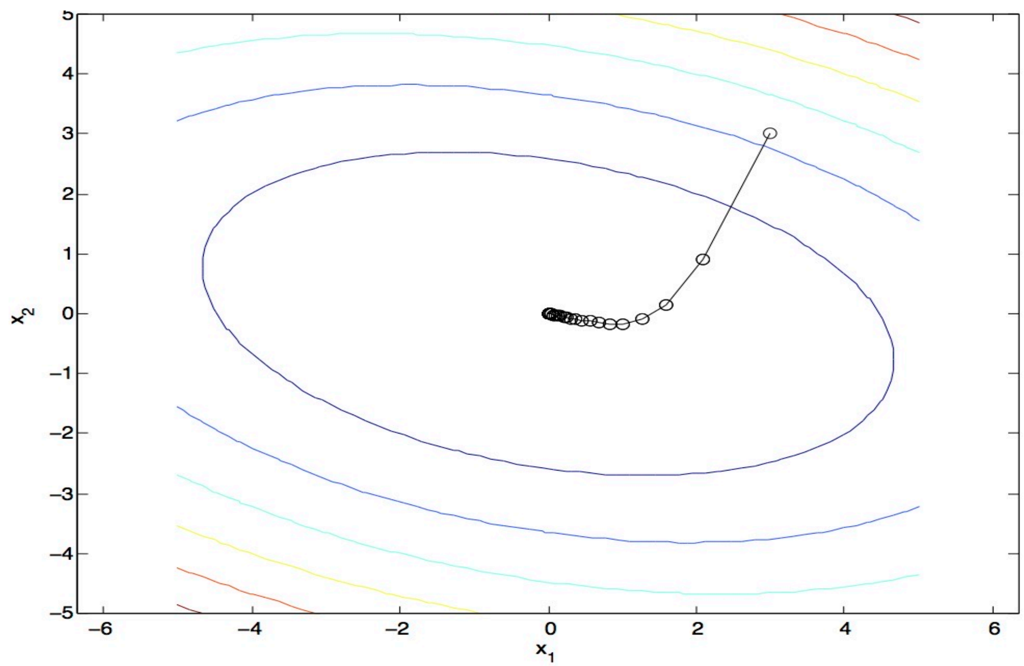
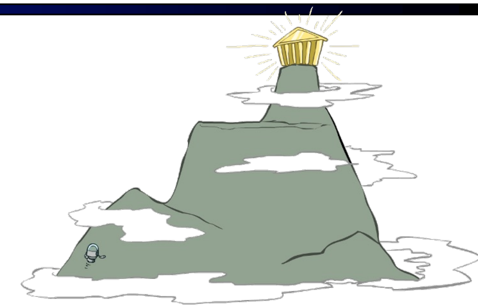


Figure source: Mathworks

What is the Steepest Direction?*



$$\max_{\Delta:\Delta_1^2+\Delta_2^2\le\varepsilon} g(w+\Delta)$$

- First-Order Taylor Expansion: $g(w+\Delta) \approx g(w) + \frac{\partial g}{\partial w_1}\Delta_1 + \frac{\partial g}{\partial w_2}\Delta_2$
- Steepest Descent Direction: $\max_{\Delta:\Delta_1^2+\Delta_2^2\le\varepsilon} g(w) + \frac{\partial g}{\partial w_1}\Delta_1 + \frac{\partial g}{\partial w_2}\Delta_2$
- Note: $\max_{\Delta:\|\Delta\|\le\varepsilon} \Delta^\top a$ → $\Delta = \varepsilon\frac{a}{\|a\|}$
- Hence, solution: $\Delta = \varepsilon\frac{\nabla g}{\|\nabla g\|}$ **Gradient direction = steepest direction** $\nabla g = \begin{bmatrix} \frac{\partial g}{\partial w_1} \\ \frac{\partial g}{\partial w_2} \end{bmatrix}$

Gradient in n dimensions

$$\nabla g = \begin{bmatrix} \frac{\partial g}{\partial w_1} \\ \frac{\partial g}{\partial w_2} \\ \cdots \\ \frac{\partial g}{\partial w_n} \end{bmatrix}$$

Optimization Procedure: Gradient Ascent

- `init` w
- `for iter = 1, 2, …`

$$w \leftarrow w + \alpha * \nabla g(w)$$

- α : learning rate --- tweaking parameter that needs to be chosen carefully
- How? Try multiple choices
  - Crude rule of thumb: update changes w about 0.1 – 1 %

What was the point again?

- We want to set w to maximize the log likelihood that logistic regression assigns to the data

$$\max_w \quad ll(w) = \max_w \quad \sum_i \log P(y^{(i)}|x^{(i)}; w)$$

with:

$$P(y^{(i)} = +1|x^{(i)}; w) = \frac{1}{1 + e^{-w \cdot f(x^{(i)})}}$$

$$P(y^{(i)} = -1|x^{(i)}; w) = 1 - \frac{1}{1 + e^{-w \cdot f(x^{(i)})}}$$

So we (repeatedly) calculate $\nabla_w ll(w)$ and then use that do gradient ascent

Batch Gradient Ascent on the Log Likelihood Objective

$$\max_w \; ll(w) = \max_w \; \underbrace{\sum_i \log P(y^{(i)}|x^{(i)};w)}_{g(w)}$$

- `init` w
- `for iter = 1, 2, …`

$$w \leftarrow w + \alpha * \sum_i \nabla \log P(y^{(i)}|x^{(i)};w)$$

Stochastic Gradient Ascent on the Log Likelihood Objective

$$\max_w \quad ll(w) = \max_w \quad \sum_i \log P(y^{(i)}|x^{(i)}; w)$$

Observation: once gradient on one training example has been computed, might as well incorporate before computing next one

- `init` w
- `for iter = 1, 2, …`
  - `pick random j`

    $$w \leftarrow w + \alpha * \nabla \log P(y^{(j)}|x^{(j)}; w)$$

Mini-Batch Gradient Ascent on the Log Likelihood Objective

$$\max_w \quad ll(w) = \max_w \quad \sum_i \log P(y^{(i)}|x^{(i)}; w)$$

Observation: gradient over small set of training examples (=mini-batch) can be computed in parallel, might as well do that instead of a single one

- init w
- for iter = 1, 2, …
  - pick random subset of training examples J

$$w \leftarrow w + \alpha * \sum_{j \in J} \nabla \log P(y^{(j)}|x^{(j)}; w)$$