Simple Variable Selection LASSO: Sparse Regression

Machine Learning – CSE546

Kevin Jamieson

University of Washington

April 24, 2019

Sparsity

$$\widehat{w}_{LS} = \arg\min_{w} \sum_{i=1}^{n} \left(y_i - x_i^T w\right)^2$$

- Vector **w** is sparse, if many entries are zero

  - **Interpretability**: What are the relevant dimension to make a prediction?

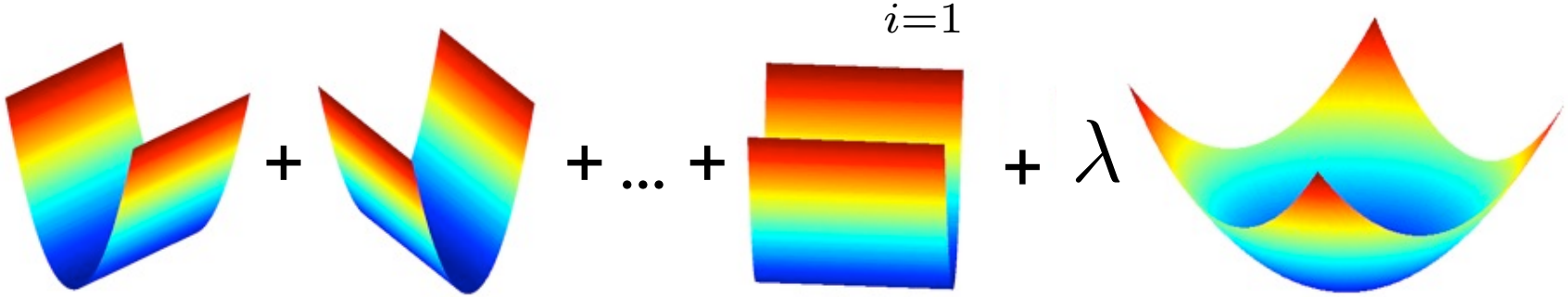
Lot size
Single Family
Year built
Last sold price
Last sale price/sqft
Finished sqft
Unfinished sqft
Finished basement sqft
floors
Flooring types
Parking type
Parking amount
Cooling
Heating
Exterior materials
Roof type
Structure style
Dishwasher
Garbage disposal
Microwave
Range / Oven
Refrigerator
Washer
Dryer
Laundry location
Heating type
Jetted Tub
Deck
Fenced Yard
Lawn
Garden
Sprinkler System

  - How do we find "best" subset among all possible?

Ridge vs. Lasso Regression

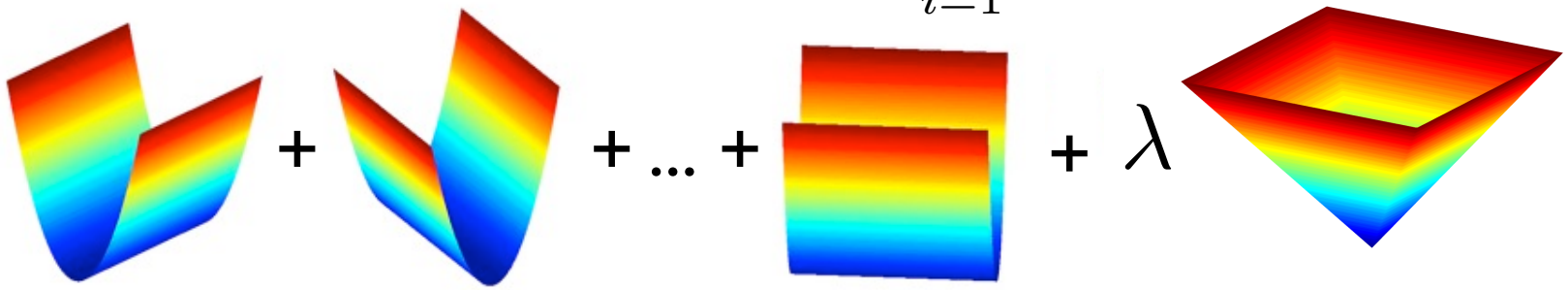
- Ridge Regression objective:

$$\widehat{w}_{ridge} = \arg\min_{w} \sum_{i=1}^{n} \left(y_i - x_i^T w\right)^2 + \lambda ||w||_2^2$$



- Lasso objective:

$$\widehat{w}_{lasso} = \arg\min_{w} \sum_{i=1}^{n} \left(y_i - x_i^T w\right)^2 + \lambda ||w||_1$$



Penalized Least Squares

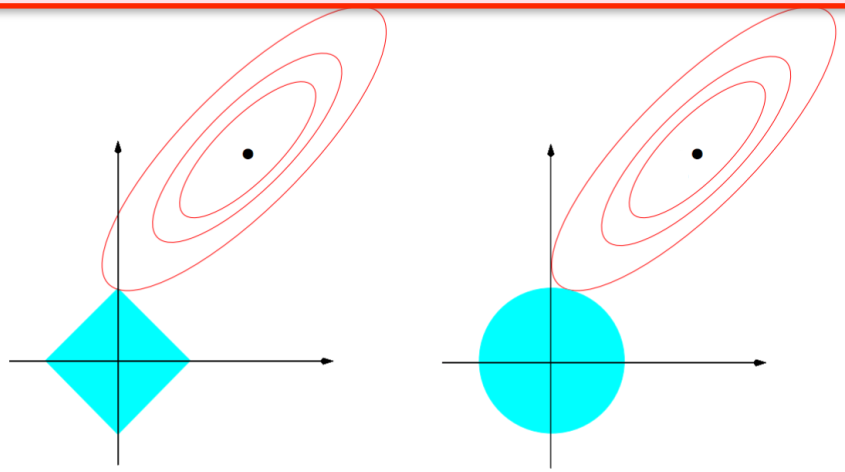
$$\text{Ridge} : r(w) = ||w||_2^2 \qquad \text{Lasso} : r(w) = ||w||_1$$

$$\widehat{w}_r = \arg\min_w \sum_{i=1}^{n} \left(y_i - x_i^T w\right)^2 + \lambda r(w)$$

For any $\lambda \geq 0$ for which $\widehat{w}_r$ achieves the minimum, there exists a $\nu \geq 0$ such that

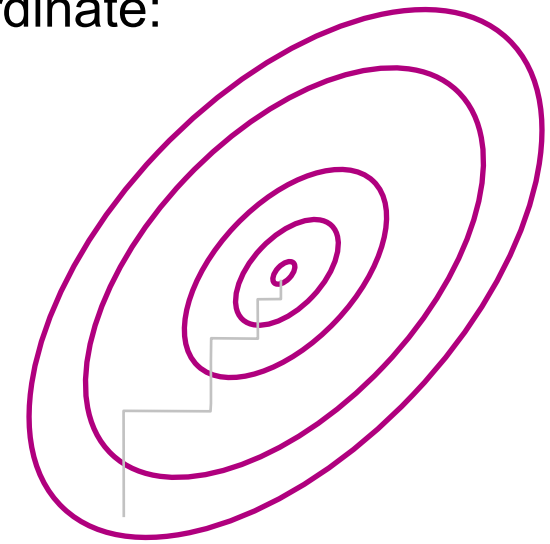
$$\widehat{w}_r = \arg\min_w \sum_{i=1}^{n} \left(y_i - x_i^T w\right)^2 \qquad \text{subject to } r(w) \leq \nu$$

No closed form for solution for L1 norm



Coordinate Descent

- Given a function, we want to find minimum
- Often, it is easy to find minimum along a single coordinate:



- How do we pick next coordinate?
- Super useful approach for *many* problems
  - Converges to optimum in some cases, such as LASSO

Optimizing LASSO Objective One Coordinate at a Time

Fix any $j \in \{1, \ldots, d\}$

$$\sum_{i=1}^{n} \left(y_i - x_i^T w\right)^2 + \lambda ||w||_1 = \sum_{i=1}^{n} \left(y_i - \sum_{k=1}^{d} x_{i,k}\, w_k\right)^2 + \lambda \sum_{k=1}^{d} |w_k|$$

$$= \sum_{i=1}^{n} \left(\left(y_i - \sum_{k \neq j} x_{i,k}\, w_k\right) - x_{i,j}\, w_j\right)^2 + \lambda \sum_{k \neq j} |w_k| + \lambda |w_j|$$

Initialize $\widehat{w}_k = 0$ for all $k \in \{1, \ldots, d\}$

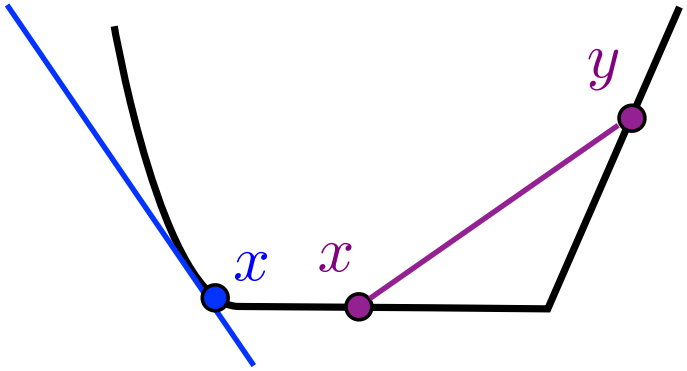
Loop over $j \in \{1, \ldots, d\}$: loss function is decomposable

$$r_i^{(j)} = y_i - \sum_{k \neq j} x_{i,j} \widehat{w}_k$$

$$\widehat{w}_j = \arg\min_{w_j} \sum_{i=1}^{n} \left(r_i^{(j)} - x_{i,j}\, w_j\right)^2 + \lambda |w_j|$$

Convex Functions

- Equivalent definitions of convexity:



f convex:

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

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

- **Gradients** lower bound convex functions and are unique at **x** iff function differentiable at **x**
- **Subgradients** generalize gradients to non-differentiable points:
  - Any supporting hyperplane at **x** that lower bounds entire function

$$g \text{ is a subgradient at } x \text{ if } f(y) \geq f(x) + g^T (y - x)$$

HAS TO GO THROUGH POINT OF TANGENCY

Taking the Subgradient

$$\widehat{w}_j = \arg\min_{w_j} \sum_{i=1}^{n} \left(r_i^{(j)} - x_{i,j}\, w_j \right)^2 + \lambda |w_j|$$

g is a subgradient at x if $f(y) \geq f(x) + g^T(y - x)$

- Convex function is minimized at w if 0 is a sub-gradient at w.

$$\partial_{w_j} |w_j| = \begin{cases} 1 & \text{if } w_j > 0 \\ [-1, 1] & \text{if } w_j = 0 \\ -1 & \text{if } w_j < 0 \end{cases}$$

$$\partial_{w_j} \sum_{i=1}^{n} \left(r_i^{(j)} - x_{i,j}\, w_j \right)^2 = \sum_{i=1}^{n} (-2x_{i,j}) \left(r_i^{(j)} - x_{i,j} w_j \right)$$

$$= \underbrace{-2 \left(\sum_{i=1}^{n} x_{i,j} r_i^{(j)} \right)}_{=:c_j} + \underbrace{2 \left(\sum_{i=1}^{n} x_{i,j}^2 \right)}_{=:a_j} w_j$$

when f is differentiable, the gradient = subgradient

Setting Subgradient to 0

$$\partial_{w_j}\left(\sum_{i=1}^{n}\left(r_i^{(j)} - x_{i,j}\, w_j\right)^2 + \lambda|w_j|\right) = \begin{cases} a_j w_j - c_j - \lambda & \text{if } w_j < 0 \\ [-c_j - \lambda, -c_j + \lambda] & \text{if } w_j = 0 \\ a_j w_j - c_j + \lambda & \text{if } w_j > 0 \end{cases}$$

absolute value

set all these equal to zero

btwn subgrads of left and right at zero

$$a_j = 2(\sum_{i=1}^{n} x_{i,j}^2) \qquad c_j = 2(\sum_{i=1}^{n} r_i^{(j)} x_{i,j})$$

$$\widehat{w}_j = \arg\min_{w_j} \sum_{i=1}^{n}\left(r_i^{(j)} - x_{i,j}\, w_j\right)^2 + \lambda|w_j|$$

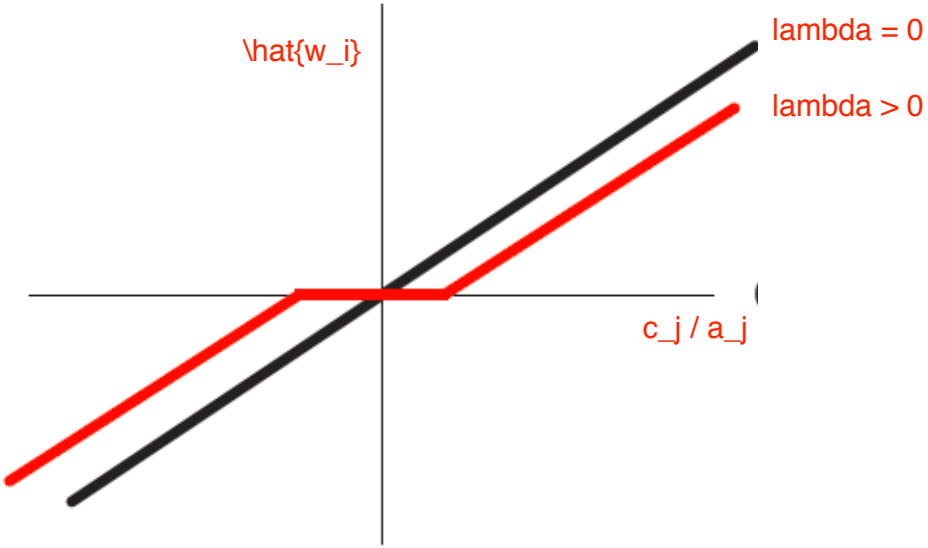
w is a minimum if
0 is a sub-gradient at w

$$\widehat{w}_j = \begin{cases} (c_j + \lambda)/a_j & \text{if } c_j < -\lambda \\ 0 & \text{if } |c_j| \le \lambda \\ (c_j - \lambda)/a_j & \text{if } c_j > \lambda \end{cases}$$

Soft Thresholding

$$\widehat{w}_j = \begin{cases} (c_j + \lambda)/a_j & \text{if } c_j < -\lambda \\ 0 & \text{if } |c_j| \leq \lambda \\ (c_j - \lambda)/a_j & \text{if } c_j > \lambda \end{cases}$$

$$a_j = 2\sum_{i=1}^{n} x_{i,j}^2 \qquad c_j = 2\sum_{i=1}^{n}\left(y_i - \sum_{k \neq j} x_{i,k}\, w_k\right) x_{i,j}$$



Coordinate Descent for LASSO (aka Shooting Algorithm)

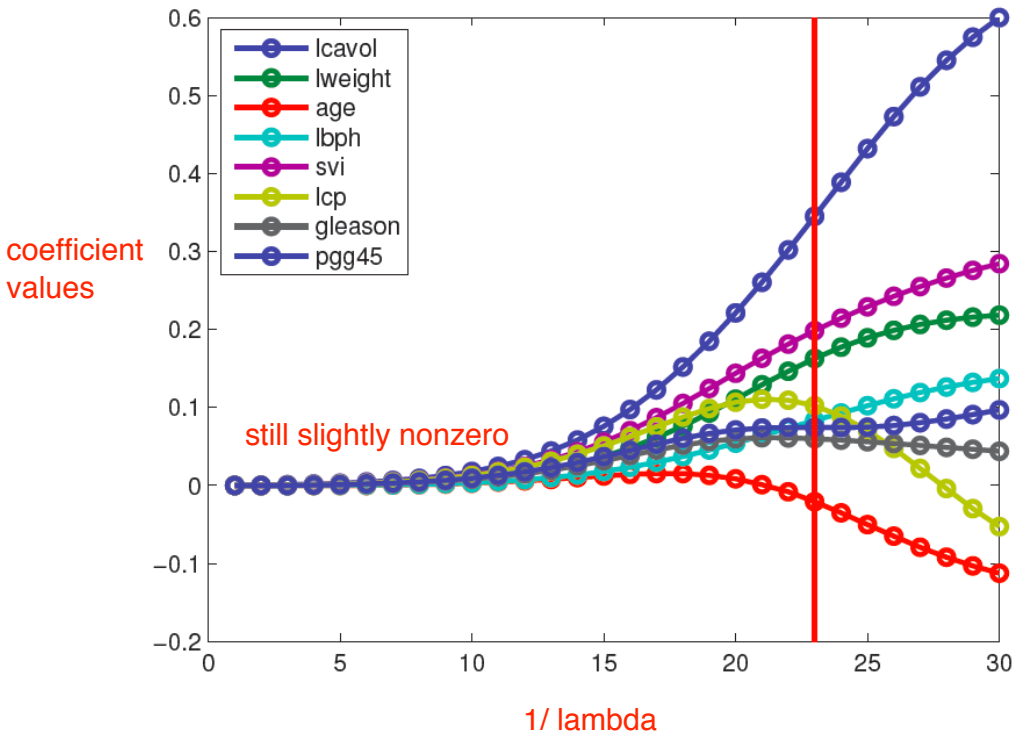
- Repeat until convergence do not repeat coordinates?
  - Pick a coordinate *l* at (random or sequentially)
    - Set:
    - Where:

$$\widehat{w}_j = \begin{cases} (c_j + \lambda)/a_j & \text{if } c_j < -\lambda \\ 0 & \text{if } |c_j| \leq \lambda \\ (c_j - \lambda)/a_j & \text{if } c_j > \lambda \end{cases}$$

$$a_j = 2\sum_{i=1}^{n} x_{i,j}^2 \qquad c_j = 2\sum_{i=1}^{n}\left(y_i - \sum_{k \neq j} x_{i,k}\, w_k\right) x_{i,j}$$

  - For convergence rates, see Shalev-Shwartz and Tewari 2009
- Other common technique = LARS
  - Least angle regression and shrinkage, Efron et al. 2004

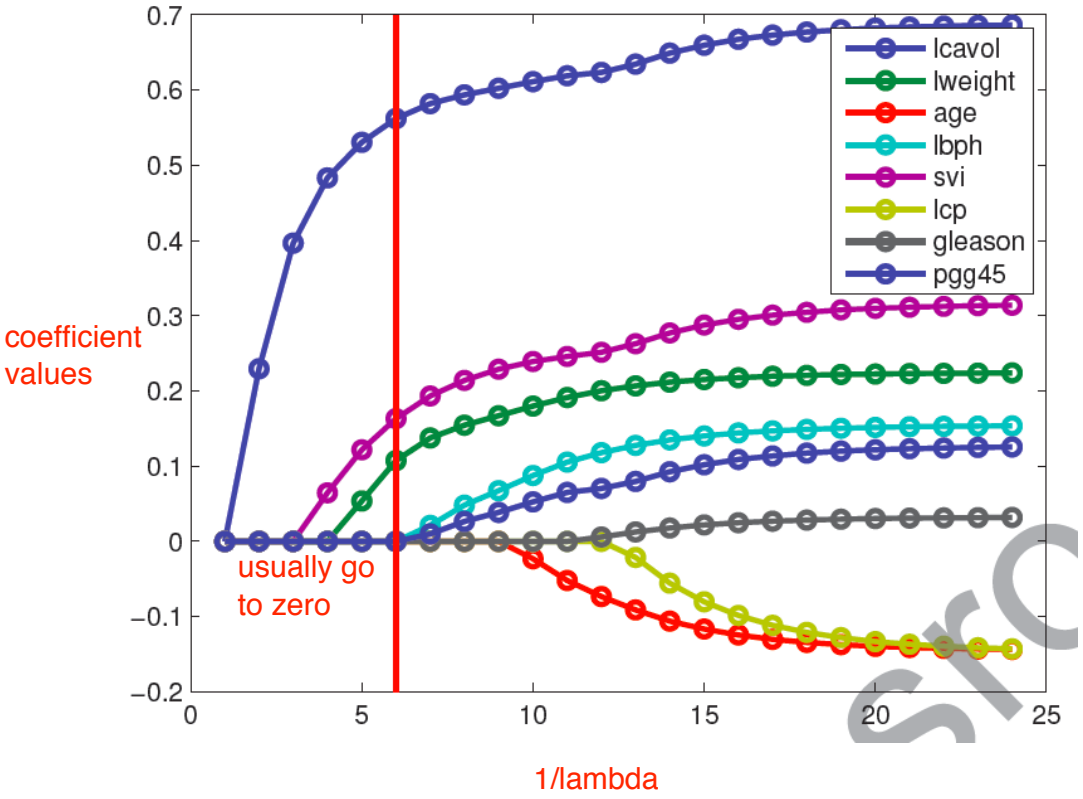
Recall: *Ridge* Coefficient Path



From Kevin Murphy textbook

- Typical approach: select λ using cross validation

Now: ***LASSO*** *Coefficient Path*



From
Kevin Murphy
textbook

What you need to know

- Variable Selection: find a sparse solution to learning problem
- Lasso (or L1 regularization) is just one way to find sparse solutions
- **Be careful about interpreting selected features**
  - sensitive to **correlations** between features
  - sensitive to **scale** of features (normalize your data)
  - result depends on **algorithm** used (with same loss function)
  - under certain conditions theoretical guarantees exist for lasso

units on data matter…

i.e. how we choose coordina in coordinate descent

i.e. correlated variables are ones that are expected to vary together (ie. number bathrooms and number toilets)… One of these will be excluded, but that does not mean toilets are unimportant…

zero weight on a feature does NOT necessarily mean that feature is meaningless: it could be correlated to another feature. In this case, all variability in that data is explained by just one of these two, even though both may be important.

CAREFUL ON INTERPRETING OUTPUT

Classification Logistic Regression

Machine Learning – CSE546

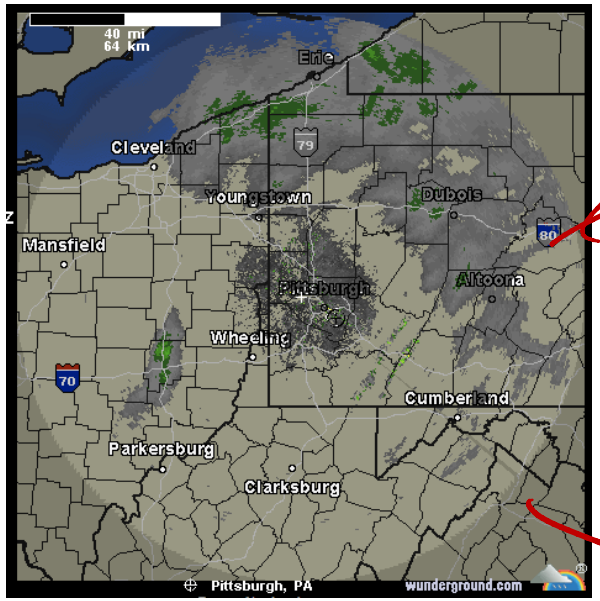
Kevin Jamieson

University of Washington

April 24, 2019

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

Weather prediction revisted



Temperature

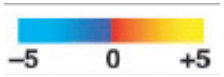
63°F

Reading Your Brain, Simple Example

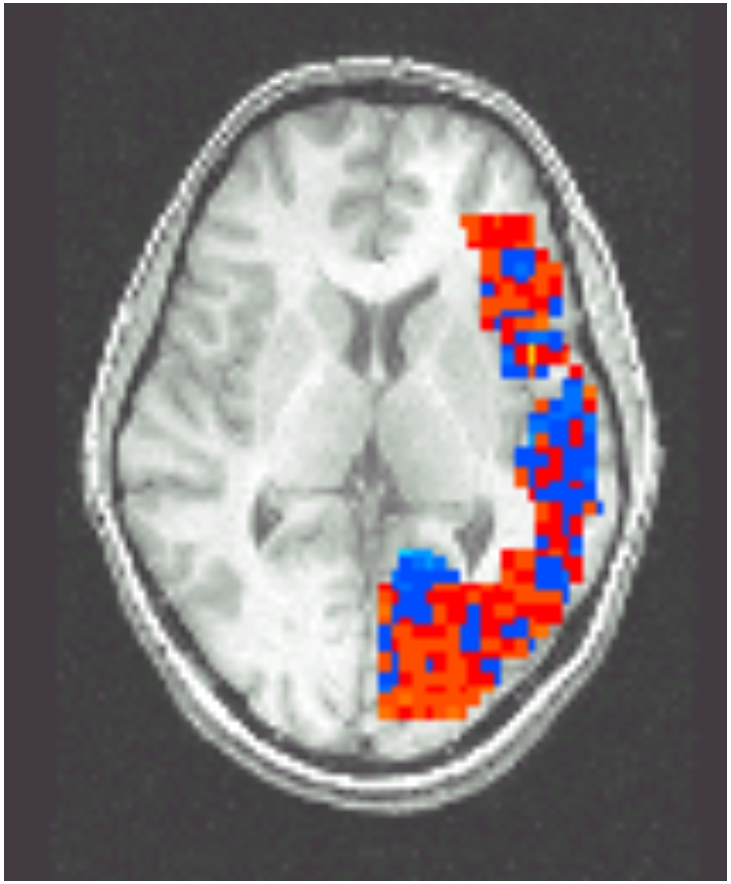
[Mitchell et al.]

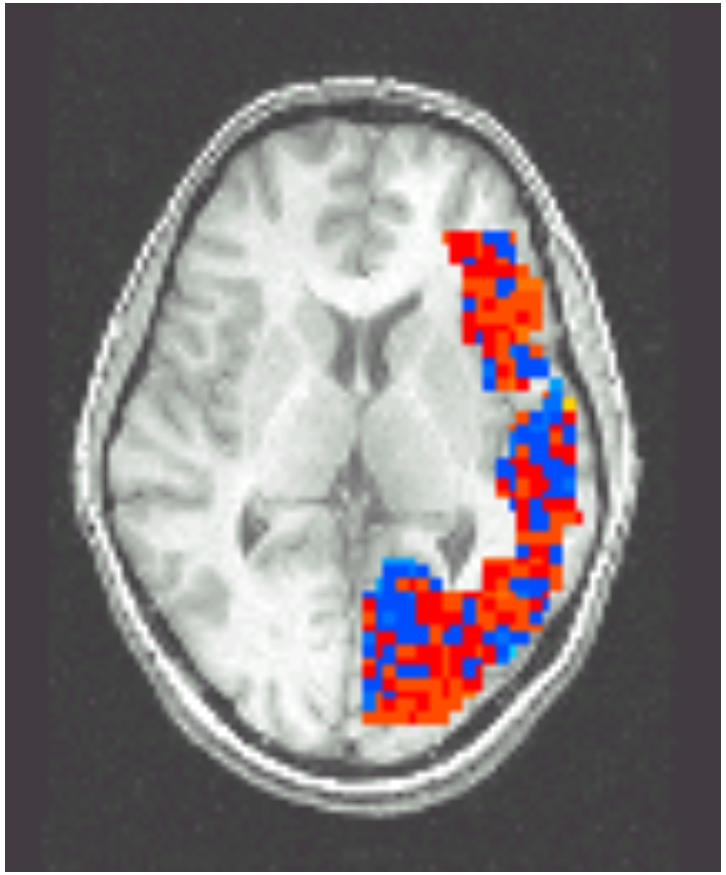
Pairwise classification accuracy: 85%

Person



Animal





Classification

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

i.e. image (pixels) —> subject in image (dog, cat, fish)

$$Y \in \{1, 2, \ldots, k\}$$

- **Loss function:** # of incorrect predictions —> 1{f(x) != y}
- **Expected loss of f:**

E_{XY} [1{f(x) != y}] = E_X [E_{Y I X}[1{f(x) != Y I X = x }]

P(Y != f(x) I X = x)

= sum_i

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

choose the Y that maximizes P(Y I X = x)

Classification

- **Learn**: f:**X** —>Y
  - **X** – features
  - Y – target classes $Y \in \{1, 2, \ldots, k\}$
- **Loss function:** $\ell(f(x), y) = \mathbf{1}\{f(x) \neq y\}$
- **Expected loss of f:**

$$\mathbb{E}_{XY}[\mathbf{1}\{f(X) \neq Y\}] = \mathbb{E}_X[\mathbb{E}_{Y|X}[\mathbf{1}\{f(x) \neq Y\}|X = x]]$$

$$\mathbb{E}_{Y|X}[\mathbf{1}\{f(x) \neq Y\}|X = x] = \sum_i P(Y = i|X = x)\mathbf{1}\{f(x) \neq i\} = \sum_{i \neq f(x)} P(Y = i|X = x)$$

$$= 1 - P(Y = f(x)|X = x)$$

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

$$f(x) = \arg\max_y \mathbb{P}(Y = y|X = x)$$

Binary Classification

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

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

- **Loss function:** $\ell(f(x), y) = \mathbf{1}\{f(x) \neq y\}$

- **Expected loss of f:**

$$\mathbb{E}_{XY}[\mathbf{1}\{f(X) \neq Y\}] = \mathbb{E}_X[\mathbb{E}_{Y|X}[\mathbf{1}\{f(x) \neq Y\}|X = x]]$$

$$\mathbb{E}_{Y|X}[\mathbf{1}\{f(x) \neq Y\}|X = x] = \sum_i P(Y = i|X = x)\mathbf{1}\{f(x) \neq i\} = \sum_{i \neq f(x)} P(Y = i|X = x)$$

$$= 1 - P(Y = f(x)|X = x)$$

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

$$f(x) = \arg\max_y \mathbb{P}(Y = y|X = x)$$

Binary Classification

- Bayes optimal classifier:

$$f(x) = \arg\max_y \mathbb{P}(Y = y | X = x) \qquad Y \in \{0, 1\}$$

- Suppose we have iid example pairs $\{(x_i, y_i)\}_{i=1}^n$ and m of the x_i's are equal to $\widetilde{x}$.

What is a natural estimator for $\theta_* := \mathbb{P}(Y = 1 | X = \widetilde{x})$?

As we saw earlier, if we know the true conditional distribution P(Y I X) we know which Y to choose. But here we have to approximate this distribution from the data.

Binary Classification

- Bayes optimal classifier:

$$f(x) = \arg\max_y \mathbb{P}(Y = y | X = x) \qquad Y \in \{0, 1\}$$

- Suppose we have iid example pairs $\{(x_i, y_i)\}_{i=1}^n$ and m of the x_i's are equal to $\widetilde{x}$.

What is a natural estimator for $\theta_* := \mathbb{P}(Y = 1 | X = \widetilde{x})$?

Most natural estimator is approximate P(Y | X) from data and use that: P(Y = y AND X = x) / P(X = x) from data

If k of the m labels are equal to $Y = 1$ then

$$\widehat{\theta}_{MLE} = k/m = \arg\max_\theta \theta^k (1 - \theta)^{m-k}$$

Binary Classification

- Bayes optimal classifier:

$$f(x) = \arg\max_y \mathbb{P}(Y = y | X = x) \qquad Y \in \{0, 1\}$$

- Suppose we have iid example pairs $\{(x_i, y_i)\}_{i=1}^n$

vector of d zeros or ones.

In a discrete feature space, such as $x_i \in \{0,1\}^d$, this motivates the estimator

$$\widehat{f}(x) = \arg\max_{y \in \{0,1\}} \frac{\sum_{i=1}^n \mathbf{1}\{x_i = x, y_i = y\}}{\sum_{i=1}^n \mathbf{1}\{x_i = x\}}$$

Potential issues?

2^d possible values for x_i… but if n < 2^d we cannot approximate P(x_i) if we have never seen that particular x_i before

Binary Classification

- Bayes optimal classifier:

$$f(x) = \arg\max_y \mathbb{P}(Y = y | X = x) \qquad Y \in \{0, 1\}$$

- Suppose we have iid example pairs $\{(x_i, y_i)\}_{i=1}^n$

In a discrete feature space, such as $x_i \in \{0,1\}^d$, this motivates the estimator

$$\widehat{f}(x) = \arg\max_{y \in \{0,1\}} \frac{\sum_{i=1}^n \mathbf{1}\{x_i = x, y_i = y\}}{\sum_{i=1}^n \mathbf{1}\{x_i = x\}}$$

Potential issues?

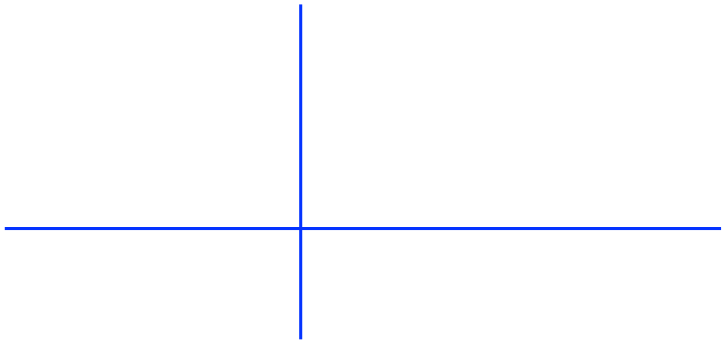
2^d possible inputs could be larger than number of examples n

To generalize to new unseen x's we need a model for $P(Y = y | X = x)$ for any x

Link Functions

- Estimating P(Y=1|**X**=x): Why not use standard linear regression model?

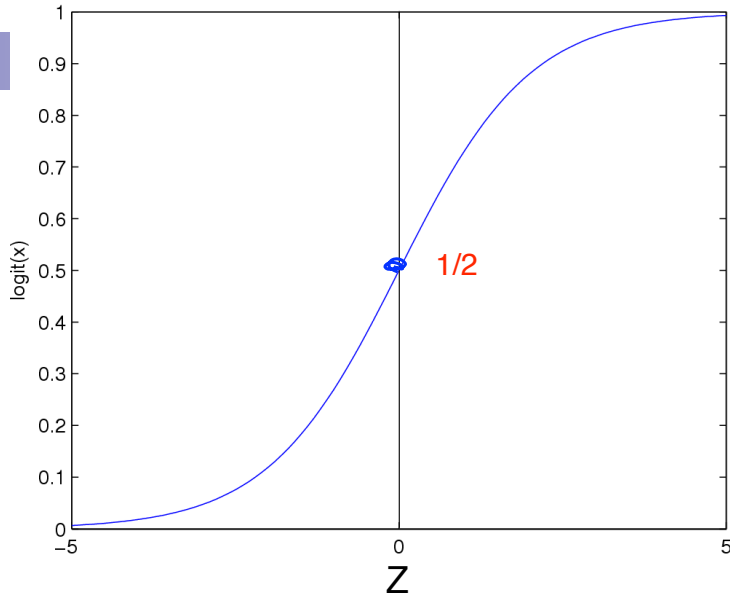
probabilities can only take values in zero to one but a linear function can take the value of any real number



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

Logistic Regression

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



- Learn P(Y=1|**X**=x) directly
  - Assume a particular functional form for link function
  - Sigmoid applied to a linear function of the input features:

our weights matrix for each element in x_i = {0,1}^d

$$\mathbb{P}(Y=1|X=x,w)=\sigma(w^Tx)=\frac{1}{1+\exp(-w^Tx)}$$

$$\mathbb{P}(Y=0|X=x,w)=1-\sigma(w^Tx)=\frac{\exp(-w^Tx)}{1+\exp(-w^Tx)}$$

$$=\frac{1}{1+\exp(w^Tx)}$$

Features can be discrete or continuous!

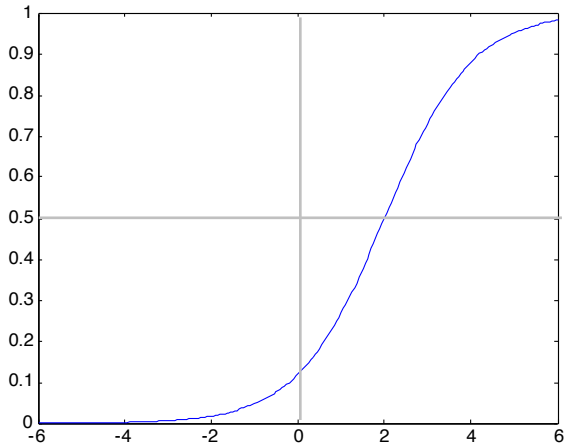
Understanding the sigmoid

bias w_Tx

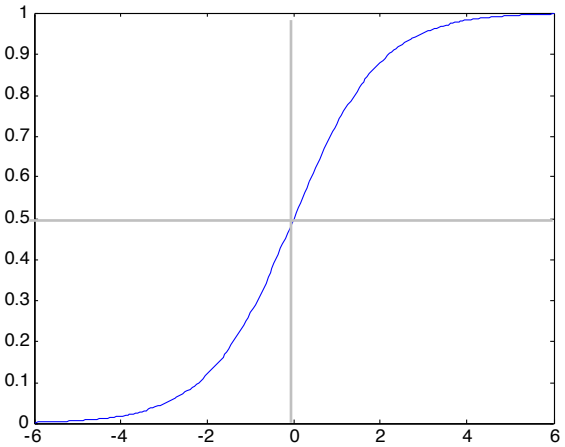
$$g(w_0 + \sum_i w_i x_i) = \frac{1}{1 + e^{w_0 + \sum_i w_i x_i}}$$

In the 2D case when w = w_1, a single element

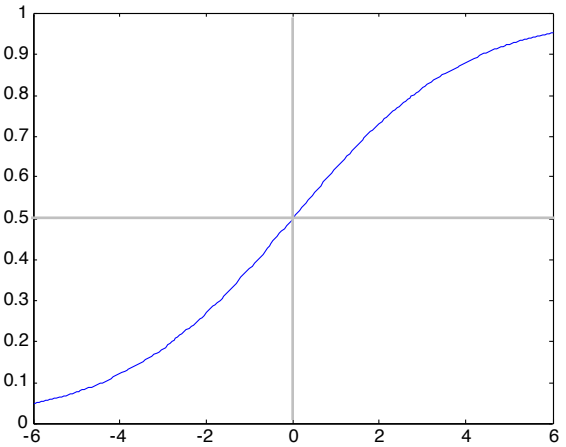
w_0=-2, w_1=-1



w_0=0, w_1=-1



w_0=0, w_1=-0.5



Sigmoid for binary classes

$$\mathbb{P}(Y=1|X=x,w)=\sigma(w^Tx)=\frac{1}{1+\exp(-w^Tx)}$$

$$\mathbb{P}(Y=0|X=x,w)=1-\sigma(w^Tx)=\frac{\exp(-w^Tx)}{1+\exp(-w^Tx)}$$
$$=\frac{1}{1+\exp(w^Tx)}$$

$$\frac{\mathbb{P}(Y=1|x,w)}{\mathbb{P}(Y=0|x,w)}=\exp(w^Tx)$$

If this ratio is greater than one, I will choose Y = 1.
If it is less than one, I will choose Y = 0

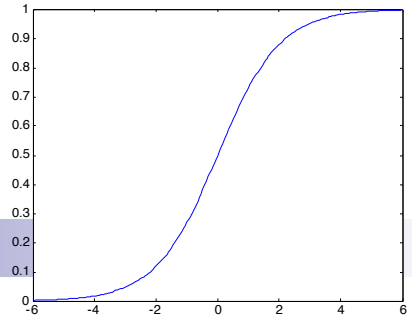
$$\log\left(\frac{\mathbb{P}(Y=1|x,w)}{\mathbb{P}(Y=0|x,w)}\right)=w^Tx$$

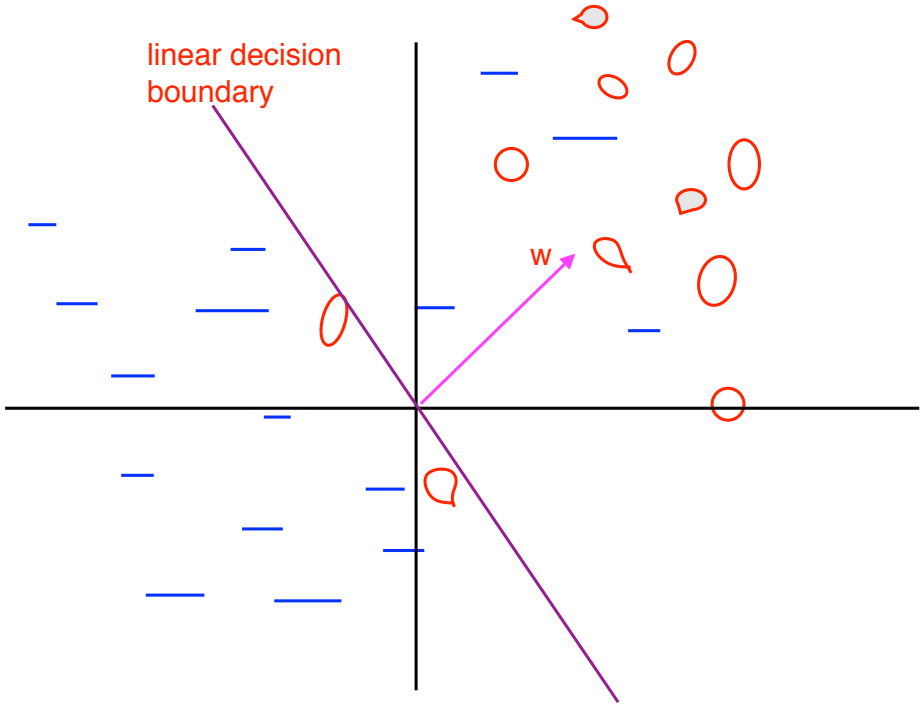
Linear Decision Rule!

w^TX > 0, choose Y = 1
else choose Y = 0

Logistic Regression – a Linear classifier

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





w points in the direction of the one group!

and the line perpendicular to w divides the two groups.

sigmoid converts this to a probability… Gives us a reasonable way to find an approximation to w.

$$\log\left(\frac{\mathbb{P}(Y=1|x,w)}{\mathbb{P}(Y=0|x,w)}\right) = w^T x$$

Loss function: Conditional Likelihood

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

map features to 1 and -1
(still binary classification)

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

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

= 1 / (1 + exp(-wTx))

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

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

take log likelihood and multiply by negative one to convert to minimization problem

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

Loss function: Conditional Likelihood

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

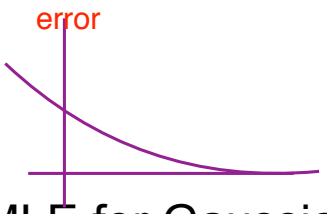
$$\widehat{w}_{MLE} = \arg\max_w \prod_{i=1}^n P(y_i|x_i, w) \qquad P(Y = y|x, w) = \frac{1}{1 + \exp(-y\, w^T x)}$$

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

when the sign of y_i and xTw match (incorrectly classified) we get large error was have the exponential of a positive number. Else the error is small
Heuristically: w matches up y as good as possible

What we are doing now

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



What we did before

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

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?

Sum of convex functions is convex, so is log(1+exp(yxTw)) convex? Yes it is!

Loss function: Conditional Likelihood

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

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

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

all local optima are global optima

Good news: $J(\mathbf{w})$ is convex function of $\mathbf{w}$, no local optima problems

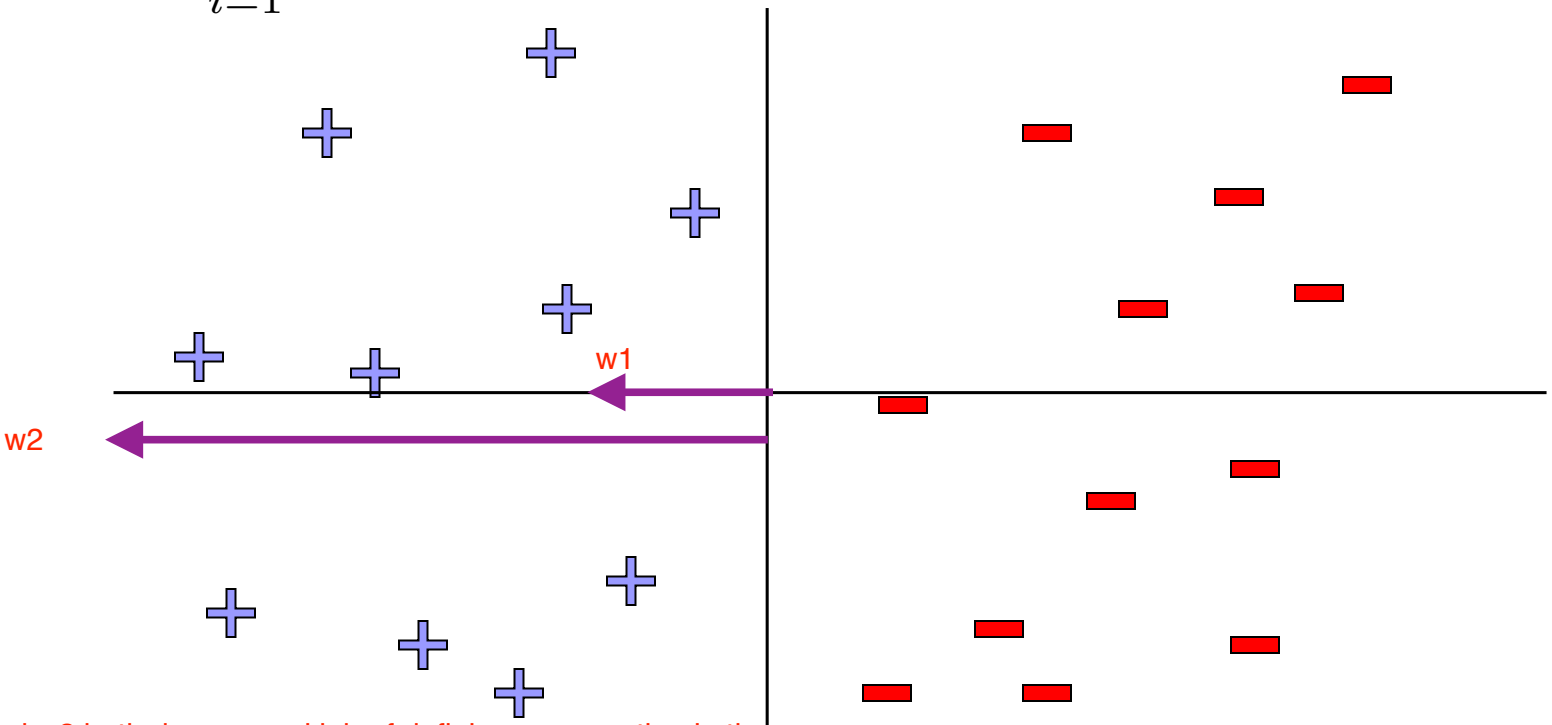
Bad news: no closed-form solution to maximize $J(\mathbf{w})$

Good news: convex functions easy to optimize

Linear Separability

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

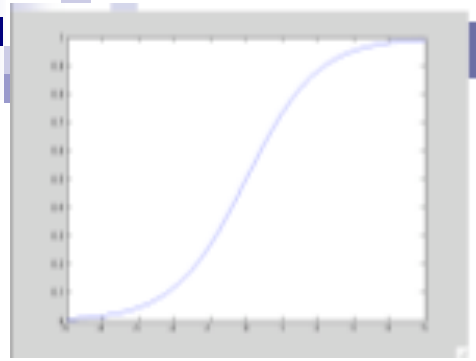
When is this loss small?



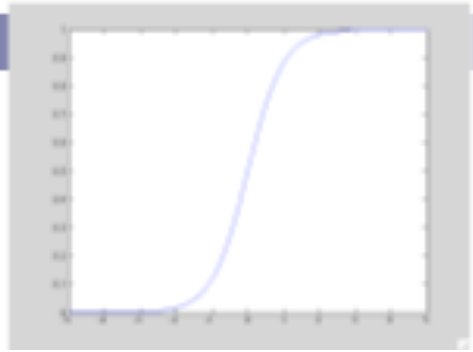
w1 and w2 both do an equal job of defining a separation in the data. We can see larger w will minimize the loss function further so w2 will always be chosen. In fact, ||w|| —> infinity. What can we do? Regularize!

Large parameters → Overfitting

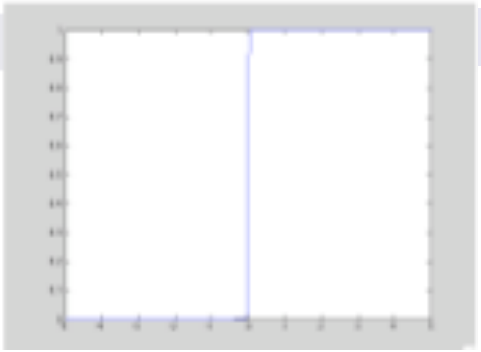
much sharper cut-off boundary



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



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



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

- If data is linearly separable, weights go to infinity

makes the distribution less interpretable as a probability. In the far right case, we are certain of the choice. We lose this idea some features being more probable to be a certain result than others.

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

Regularized Conditional Log Likelihood

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

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

Be sure to not regularize the offset b!

Gradient Descent

Machine Learning – CSE546

Kevin Jamieson

University of Washington

April 24, 2019

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.

Machine Learning Problems

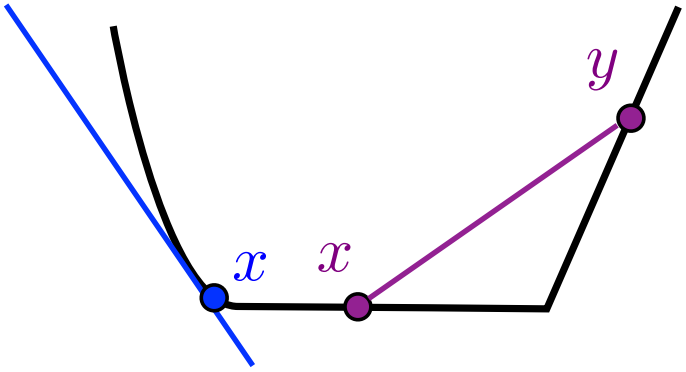
- Have a bunch of iid data of the form:

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

- Learning a model's parameters:

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

Each $\ell_i(w)$ is convex.



g is a subgradient at x if
$f(y) \geq f(x) + g^T(y - x)$

f convex:

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

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

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

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:

$$f(x+\delta) = f(x) + f'(x)\delta + \tfrac{1}{2}f''(x)\delta^2 + \ldots$$

- Gradient descent:

Taylor Series Approximation

- Taylor series in **d** dimensions:

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

- Gradient descent:

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

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

$$\nabla f(w) =$$

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

$$w_* = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$$

$$w_{t+1} - w_* =$$

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

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

$$\nabla f(w) = \mathbf{X}^T(\mathbf{X}w - \mathbf{y}) = \mathbf{X}^T\mathbf{X}w - \mathbf{X}^T\mathbf{y}$$

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

$$= (I - \eta \mathbf{X}^T\mathbf{X})w_t + \eta \mathbf{X}^T\mathbf{y}$$

$$w_* = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$$

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

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

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

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

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

Taylor Series Approximation

- Taylor series in one dimension:

$$f(x+\delta) = f(x) + f'(x)\delta + \tfrac{1}{2}f''(x)\delta^2 + \ldots$$

- Newton's method:

Taylor Series Approximation

- Taylor series in **d** dimensions:

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

- Newton's method:

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

$\nabla f(w) =$

$\nabla^2 f(w) =$

v_t 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 $f(w) = \frac{1}{2}||\mathrm{X}w - \mathrm{y}||_2^2$

$$\nabla f(w) = \mathrm{X}^T(\mathrm{X}w - \mathrm{y})$$

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

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

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

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

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

General case

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

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 $f(w_t) - f(w_*) \leq \epsilon$

Newton's method:

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

Gradient descent:

- f is *smooth* and *strongly convex:* $aI \preceq \nabla^2 f(w) \preceq bI$

Clean converge nice proofs: Bubeck

- 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

April 24, 2019

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

Machine Learning – CSE546

Kevin Jamieson

University of Washington

April 24, 2019

Stochastic Gradient Descent

- 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:

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

Each $\ell_i(w)$ is convex.

Stochastic Gradient Descent

- 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:

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

Each $\ell_i(w)$ is convex.

Gradient Descent:

$$w_{t+1} = w_t - \eta \nabla_w \left(\frac{1}{n}\sum_{i=1}^{n} \ell_i(w) \right) \Big|_{w=w_t}$$

Stochastic Gradient Descent

- 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.

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

Gradient Descent:

$$w_{t+1} = w_t - \eta \nabla_w \left(\frac{1}{n}\sum_{i=1}^n \ell_i(w) \right) \Big|_{w=w_t}$$

Stochastic Gradient Descent:

$$w_{t+1} = w_t - \eta \nabla_w \ell_{I_t}(w) \Big|_{w=w_t}$$

I_t drawn uniform at random from $\{1, \ldots, n\}$

$$\mathbb{E}[\nabla \ell_{I_t}(w)] =$$

Stochastic Gradient Descent: A Learning perspective

Machine Learning – CSE546

Kevin Jamieson

University of Washington

April 24, 2019

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 descent 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 descent rule:

- How do we estimate expected gradient?

SGD: Stochastic Gradient 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!
  - Also called stochastic gradient descent
    - Among many other names
  - VERY useful in practice!!!