syde675 pattern recognition
Review, Logistic regression

Notes inspired from: Shier Nee, SAW, Murphy PRML

Outline

- Decision Theory
- Information Theory
- Optimization

Decision Theory

A set of quantitative methods for reaching optimal decisions.

Quoted from The Editors of Encyclopaedia Britannica

> Is concerned with the reasoning underlying an agent's choices, whether this is a mundane choice between taking the bus or getting a taxi, or a more far-reaching choice about whether to pursue a demanding political career.
>
> Quoted from Stanford Encyclopedia of Philosophy

How can we decide which action is best?
This is where Bayesian decision theory comes in.

Bayes's Theorem

- Describes the probability of an event, based on prior knowledge of conditions that might be related to the event.
- Bayes' rule itself is very simple: it is just a formula for computing the probability distribution over possible values of an unknown (or **hidden**) quantity H given some observed data **Y = y**:

Prior distribution

Likelihood

$$p(H = h|Y = y) = \frac{p(H = h)p(Y = y|H = h)}{p(Y = y)}$$

$$\text{posterior} \propto \text{prior} \times \text{likelihood}$$

Posterior distribution / new belief state

Marginal Likelihood

Decision Theory

We assume the decision maker, or agent, has a set of possible actions, A, to choose from.

Example: hypothetical doctor treating someone who may have COVID
Actions, *a*:

- Do nothing
- Give patient expensive drugs with bad side effect but can save their life.

Every action has cost and benefits → depend on underlying state of nature $h \in \mathcal{H}$.

Encode this information to loss function $\ell(h, a)$, loss when we incur if action $a \in A$ is taken at state of nature, $h \in \mathcal{H}$.

We can compute posterior expected loss or risk for each possible action.

$$R(a|\boldsymbol{x}) \triangleq \mathbb{E}_{p(h|\boldsymbol{x})}\left[\ell(h,a)\right] = \sum_{h \in \mathcal{H}} \ell(h,a) p(h|\boldsymbol{x}) \longrightarrow \pi^*(\boldsymbol{x}) = \operatorname*{argmin}_{a \in \mathcal{A}} \mathbb{E}_{p(h|\boldsymbol{x})}\left[\ell(h,a)\right]$$

Classification- Zero-one loss

Use Bayesian decision theory to decide the optimal class label to predict given an observed input $x \in X$.

Suppose

- the states of nature correspond to class labels, so $H = Y = \{1, \ldots, C\}$.
- the actions correspond to class labels, so $A = Y$.

In this setting, a very commonly used loss function is the zero-one loss, $\ell_{01}(y^*, \hat{y})$, simply counts how many mistakes a hypothesis would make on training data.

	$\hat{y}=0$	$\hat{y}=1$
$y^*=0$	0	1
$y^*=1$	1	0

$$\ell_{01}(y^*, \hat{y}) = \mathbb{I}(y^* \neq \hat{y})$$

In this case, the posterior expected loss is

$$R(\hat{y}|\boldsymbol{x}) = p(\hat{y} \neq y^*|\boldsymbol{x}) = 1 - p(y^* = \hat{y}|\boldsymbol{x})$$

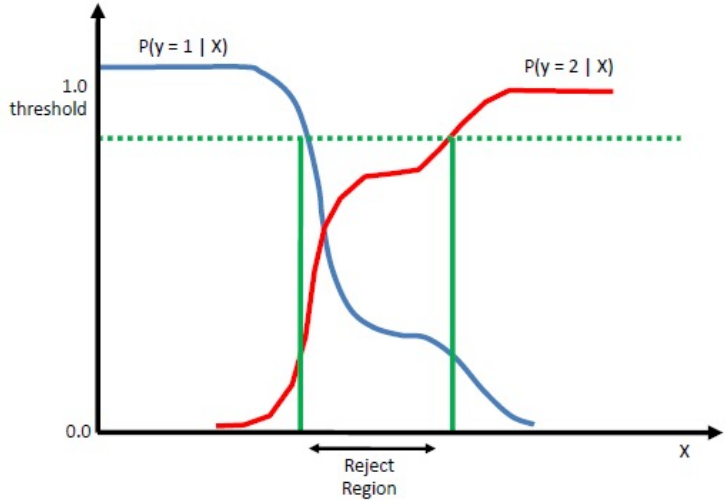
Mode of the posterior distribution or maximum a posteriori or MAP estimate

$$\pi(\boldsymbol{x}) = \underset{y \in \mathcal{Y}}{\operatorname{argmax}}\, p(y|\boldsymbol{x})$$

Classification- Rejection option

In some cases, we may be able to say “I don’t know” instead of returning an answer that we don’t really trust; this is called picking the reject option.

This is particularly important in domains such as medicine and finance where we may be risk averse



https://www.youtube.com/watch?v=P18EdAKuC1U

Classification- Accuracy

		Predicted condition	
	Total population = P + N	Positive (PP)	Negative (PN)
Actual condition	Positive (P)	True positive (TP), hit	False negative (FN), type II error, miss, underestimation
	Negative (N)	False positive (FP), type I error, false alarm, overestimation	True negative (TN), correct rejection

If we have imbalanced testing data (class 0 = 100, class 1 = 10), if the model predict everything as class 0, accuracy = 90%

Reliable?? No

Accuracy = (TP + TN) / (TP + FN + FP + TN)

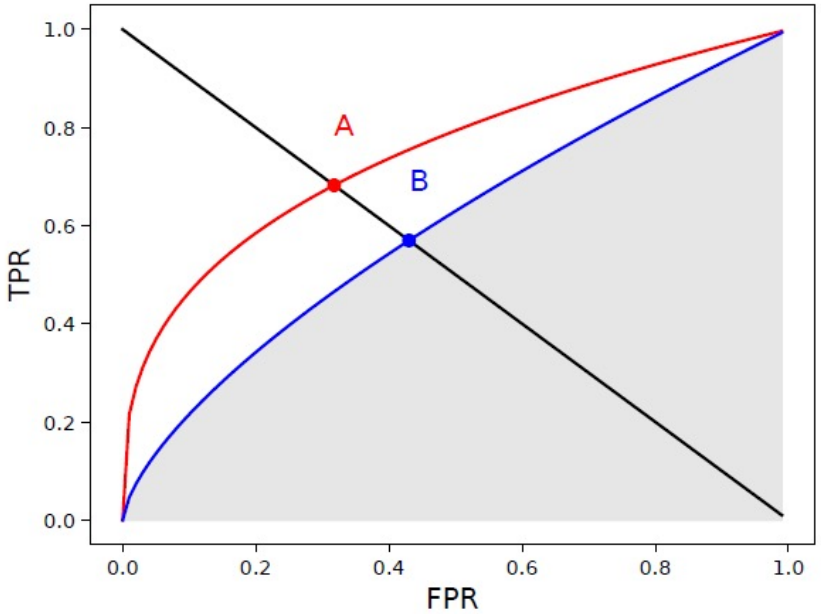
Recall / True Positive Rate (TPR) = predict how many disease correctly in disease cohort

$$TPR = TP / (TP + FN)$$

False Positive Rate (FPR) = predict how many disease in healthy cohort

$$FPR = FP / (FP + TN)$$

Classification- ROC curve



Area under ROC curve is computed.

Area Under ROC

The higher the AUROC, the better the classifier.

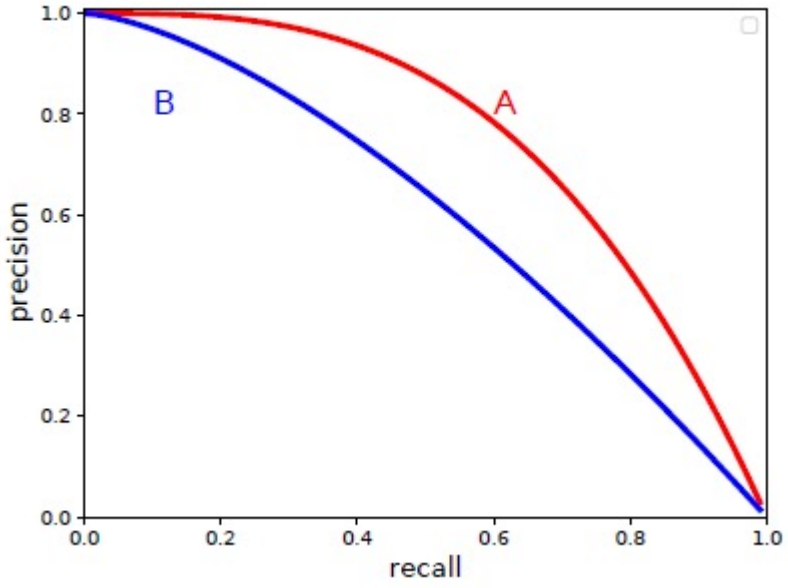
		Predicted condition	
	Total population = P + N	Positive (PP)	Negative (PN)
Actual condition	Positive (P)	True positive (TP), hit	False negative (FN), type II error, miss, underestimation
	Negative (N)	False positive (FP), type I error, false alarm, overestimation	True negative (TN), correct rejection

TPR = TP / (TP + FN) → predict how many disease correctly in disease cohort

FPR = FP / (FP + TN) → predict how many disease in healthy cohort

If we have imbalanced class, is AUROC (TPR vs FPR) reliable?

Classification- Precision-Recall Curve



Area under PR curve is computed.

The higher the AUC of PR curve, the better the classifier.

- F1 – score

Mostly used in information retrieval (query)

Recall = TPR = TP / (TP + FN) → predict how many positive correctly in positive cohort
Precision = Positive Predictive Value (PPV)
= TP / (TP +FP) → predict how many positive correctly if prediction is positive.

If we have imbalanced data, is AUROC (Precision vs Recall) reliable?

Regression – Loss function

L2 loss

$$\ell_2(h,a) = (h-a)^2$$

- Sensitive to outlier

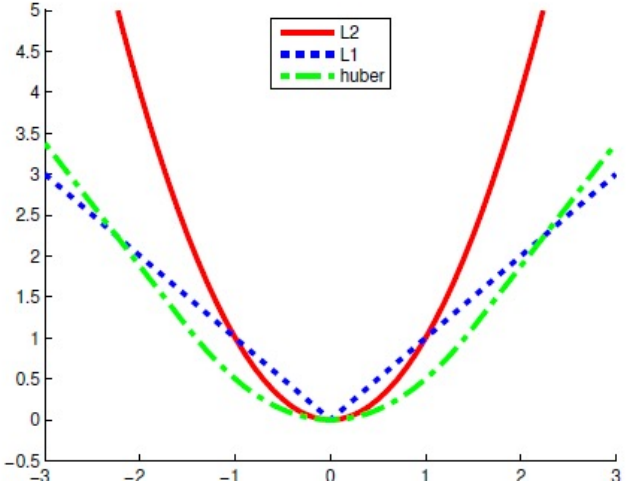
L1 loss

$$\ell_1(h,a) = |h-a|$$

Huber loss

$$\ell_\delta(h,a) = \begin{cases} r^2/2 & \text{if } |r| \le \delta \\ \delta|r| - \delta^2/2 & \text{if } |r| > \delta \end{cases}$$

where $r = h - a$.



Compare Distribution

Kullback Leibler divergence, or KL divergence is a measure of how one probability distribution is different from a second, reference probability distribution.

$$\mathbb{KL}\left(p\|q\right) \triangleq \sum_{y\in\mathcal{Y}} p(y)\log\frac{p(y)}{q(y)}$$

KL divergence

$$\mathbb{KL}(p\|q) = \sum_{y\in\mathcal{Y}} p(y)\log p(y) - \sum_{y\in\mathcal{Y}} p(y)\log q(y)$$

$$= -\mathbb{H}(p) + \mathbb{H}(p,q)$$

$$\mathbb{H}(p) \triangleq -\sum_{y} p(y)\log p(y)$$

$$\mathbb{H}(p,q) \triangleq -\sum_{y} p(y)\log q(y)$$

Entropy;
- Max if the distribution p is uniform
- Zero if p is a delta function

Cross-Entropy

If KL = 0 correctly predict the probabilities of all possible future events

Hypothesis Testing

Suppose we have two hypotheses or models, commonly called the null hypothesis, M_0, and the alternative hypothesis, M_1, and we want to know which one is more likely to be true.

The optimal decision to pick alternative hypothesis iff

$$P(M_1|D) > P(M_0|D)$$

$$\frac{P(M_1|D)}{P(M_0|D)} > 1$$

Hypothesis Testing

If we use uniform prior $p(M_0) = p(M_1) = 0.5$, the decision rule becomes

$$\frac{P(D \mid M_1)}{P(D \mid M_0)} > 1$$

Bayes Factor = ratio of marginal likelihood of two models

$$B_{1,0} \triangleq \frac{P(D \mid M_1)}{P(D \mid M_0)}$$

Bayes factor $BF(1,0)$	Interpretation
$BF < \frac{1}{100}$	Decisive evidence for M_0
$BF < \frac{1}{10}$	Strong evidence for M_0
$\frac{1}{10} < BF < \frac{1}{3}$	Moderate evidence for M_0
$\frac{1}{3} < BF < 1$	Weak evidence for M_0
$1 < BF < 3$	Weak evidence for M_1
$3 < BF < 10$	Moderate evidence for M_1
$BF > 10$	Strong evidence for M_1
$BF > 100$	Decisive evidence for M_1

Table 5.6: Jeffreys scale of evidence for interpreting Bayes factors.

Bayesian Model Selection

Suppose we have a set *M* of more than 2 models, and we want to pick the most likely.

The optimal action is picking the most probable model

$$\hat{m} = \underset{m \in \mathcal{M}}{\operatorname{argmax}}\, p(m|\mathcal{D})$$

where

$$p(m|\mathcal{D}) = \frac{p(\mathcal{D}|m)p(m)}{\sum_{m \in \mathcal{M}} p(\mathcal{D}|m)p(m)}$$

Bayesian Model Selection

If the prior over models is uniform, p(m) = 1/|M|, then the Maximum A Posterior (MAP) is given by

$$\hat{m} = \operatorname*{argmax}_{m \in \mathcal{M}} p(\mathcal{D}|m)$$

The quantity $p(\mathcal{D}|m)$ is given by

$$p(\mathcal{D}|m) = \int p(\mathcal{D}|\boldsymbol{\theta}, m) p(\boldsymbol{\theta}|m) d\boldsymbol{\theta}$$

prior

This is known as the **marginal likelihood**, or the **evidence** for model m.

Occam's razor

Consider two models, a simple one, m_1, and a more complex one, m2.

Suppose that both can explain the data by suitably optimizing their parameters.

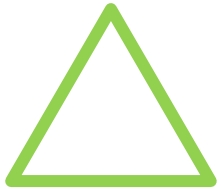
Intuitively we should prefer m_1, since it is simpler and just as good as m_2.

This principle is known as Occam's razor

Bayesian Occam's razor effect

Intuition from Squid Game

Complex Model



Simple Model

$$p(\mathcal{D}|m) = \int p(\mathcal{D}|\boldsymbol{\theta}, m)p(\boldsymbol{\theta}|m)d\boldsymbol{\theta}$$

Many θ averaged over the parameter space

$$p(\mathcal{D}|m) = \int p(\mathcal{D}|\boldsymbol{\theta}, m)p(\boldsymbol{\theta}|m)d\boldsymbol{\theta}$$

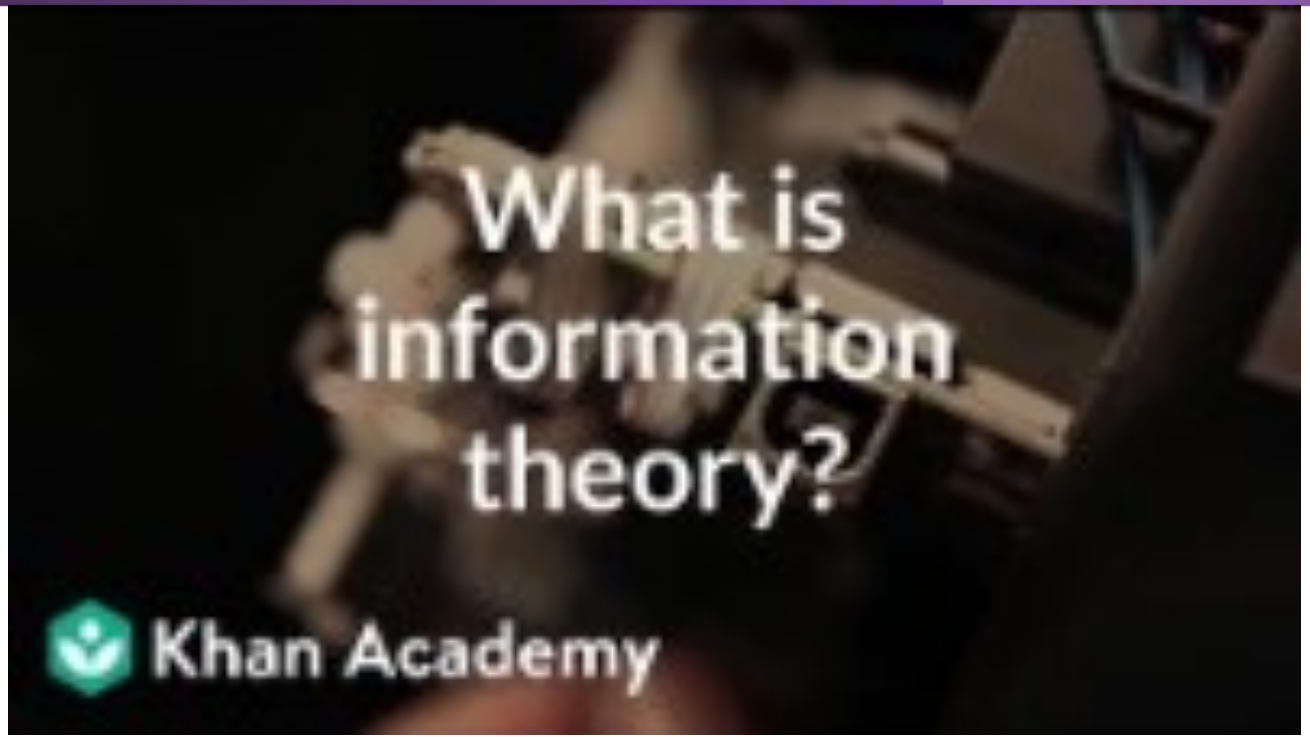
Few θ averaged over the parameter space

Decision Theory

A set of quantitative methods for reaching optimal decisions.

Bayesian Statistics
- AUROC curve
- Precision Recall Curve / F1 score
- Uses Bayes factor
- Choose model with larger marginal likelihood

Information Theory



https://www.khanacademy.org/computing/computer-science/informationtheory/info-theory/v/intro-information-theory

Entropy

Quantifying information is the foundation of the field of information theory.

The intuition behind quantifying information is the idea of measuring how much surprise there is in an event. Those events that are rare (low probability) are more surprising and therefore have more information than those events that are common (high probability).

- **Low Probability Event**: High Information (*surprising*).
- **High Probability Event**: Low Information (*unsurprising*).

Rare events are more uncertain or more surprising and require more information to represent them than common events.

Entropy

We can calculate the amount of information there is in an event using the probability of the event. This is called "*Shannon information*," "*self-information*," or simply the "*information*," and can be calculated for a discrete event *x* as follows:

- information(x) = -log(p(x))

Where *log()* is the base-2 logarithm and *p(x)* is the probability of the event *x*.

The choice of the base-2 logarithm means that the units of the information measure is in bits (binary digits). This can be directly interpreted in the information processing sense as the number of bits required to represent the event.

The calculation of information is often written as *h()*; for example:

- h(x) = -log(p(x))

The negative sign ensures that the result is always positive or zero.

Cross Entropy

Calculates the number of bits required to represent or transmit an average event from one distribution compared to another distribution

The intuition for this definition comes if we consider a target or underlying probability distribution P and an approximation of the target distribution Q. Then the cross-entropy of Q from P is average bits of information needed to identify an event drawn from the estimated probability distribution q, rather than the true distribution p

The cross-entropy between two probability distributions, such as Q from P, can be stated formally as:

- H(P, Q)

$$H(P,Q) = -\sum^{\{x \in X\}} P(x) \times \log(Q(x))$$

Where H() is the cross-entropy function, P may be the target distribution and Q is the approximation of the target distribution.

KL divergence / Relative Entropy

$$\begin{aligned}\mathbb{KL}\left(p\|q\right) &= \sum_{y\in\mathcal{Y}} p(y)\log p(y) - \sum_{y\in\mathcal{Y}} p(y)\log q(y) \\ &= -\mathbb{H}(p) + \mathbb{H}(p,q)\end{aligned}$$

$$\mathbb{H}(p) + \mathbb{KL}\left(p\|q\right) = \mathbb{H}(p,q)$$

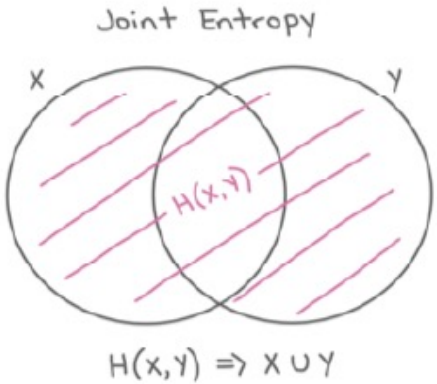
average number of extra bits required when compressing data coming from p if your code is designed based on q distribution

average bits of information needed to identify an event drawn from the estimated probability distribution q, rather than the true distribution p

Joint Entropy

The joint entropy of two random variables X and Y is defined as

$$\mathbb{H}(X,Y) = -\sum_{x,y} p(x,y) \log_2 p(x,y)$$

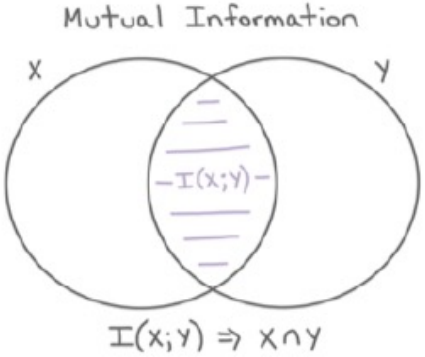


Mutual Information

How similar two distributions were

The mutual information between rv's X and Y is defined as follows:

$$\mathbb{I}(X;Y) \triangleq \mathbb{KL}\left(p(x,y) \| p(x)p(y)\right) = \sum_{y \in Y} \sum_{x \in X} p(x,y) \log \frac{p(x,y)}{p(x)\,p(y)}$$



Used in Clustering – to measure cluster quality

$$NMI(X,Y) = \frac{\mathbb{I}(X;Y)}{\min\left(\mathbb{H}(X), \mathbb{H}(Y)\right)} \leq 1$$

NMI = Normalized Mutual Information

Information Theory

Quantifying information using entropy

- Cross entropy
- KL divergence / Relative entropy
- Joint entropy
- Mutual Information

Optimization

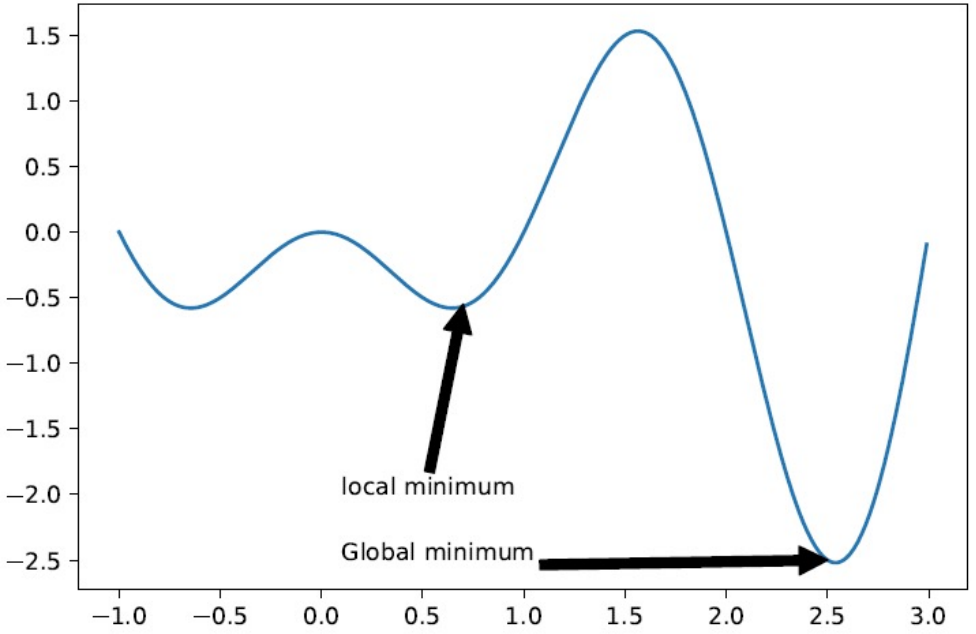
The core problem in machine learning is parameter estimation (aka model fitting).

This requires solving an optimization problem, where we try to find the values for a set of variables, $\theta \in \Theta$, that minimize a scalar-valued loss function or cost function, $\mathcal{L}(\theta)$.

$$\boldsymbol{\theta}^* \in \underset{\boldsymbol{\theta} \in \Theta}{\operatorname{argmin}} \mathcal{L}(\boldsymbol{\theta})$$

Global / Local Optimization

$$\boldsymbol{\theta}^* \in \underset{\boldsymbol{\theta} \in \Theta}{\operatorname{argmin}} \mathcal{L}(\boldsymbol{\theta})$$

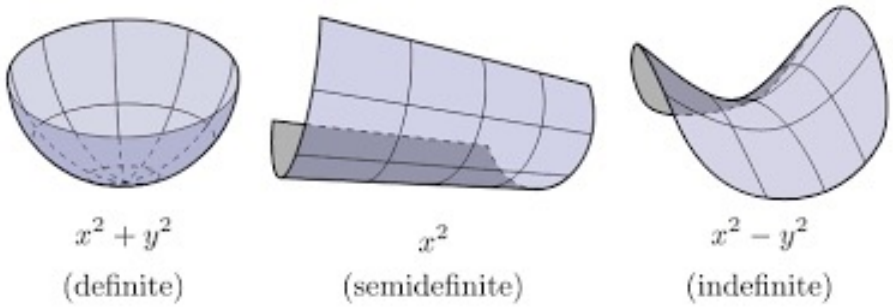


Global / Local minima

Let $g(\theta) = \nabla \mathcal{L}(\theta)$ be the gradient vector,

$\mathbf{H}(\theta) = \nabla^2 \mathcal{L}(\theta)$ be the Hessian matrix.

- Necessary condition: If θ^* is a local minimum, then we must have $g^* = \mathbf{0}$ (i.e., θ^* must be a **stationary point**), and $\mathbf{H}^*$ must be positive semi-definite.
- Sufficient condition: If $g^* = \mathbf{0}$ and $\mathbf{H}^*$ is positive definite, then θ^* is a local optimum.



Constrained / Unconstrained Optimization

Inequality constraints

Equality constraints

We define the **feasible set** as the subset of the parameter space that satisfies the constraints:

$$\mathcal{C} = \{\boldsymbol{\theta} : g_j(\boldsymbol{\theta}) \leq 0 : j \in \mathcal{I}, h_k(\boldsymbol{\theta}) = 0 : k \in \mathcal{E}\} \subseteq \mathbb{R}^D \tag{8.4}$$

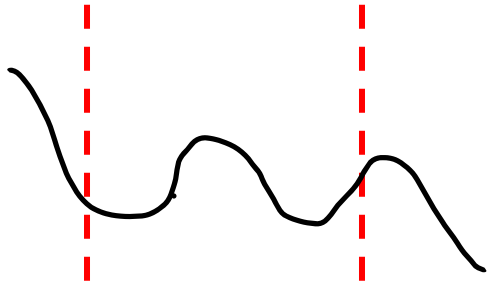
Our **constrained optimization** problem now becomes

$$\boldsymbol{\theta}^* \in \operatorname*{argmin}_{\boldsymbol{\theta} \in \mathcal{C}} \mathcal{L}(\boldsymbol{\theta}) \tag{8.5}$$

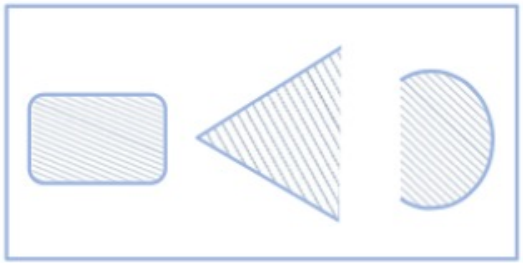
If $\mathcal{C} = \mathbb{R}^D$, it is called **unconstrained optimization**.

If too many constrained → empty feasible sets

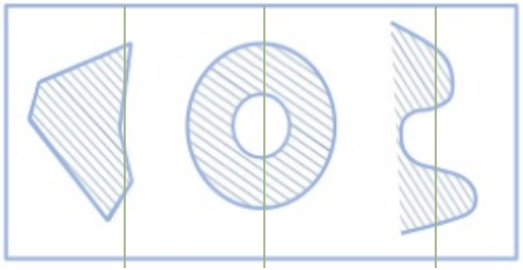
A common strategy to solve constrained problem is add penalty to the loss function such as using Lagrangian multiplier.



Convex / Concave Optimization



Convex



Not Convex

We say $\mathcal{S}$ is a **convex set** if, for any $x, x' \in \mathcal{S}$, we have

$$\lambda x + (1-\lambda)x' \in \mathcal{S}, \quad \forall\, \lambda \in [0,1]$$

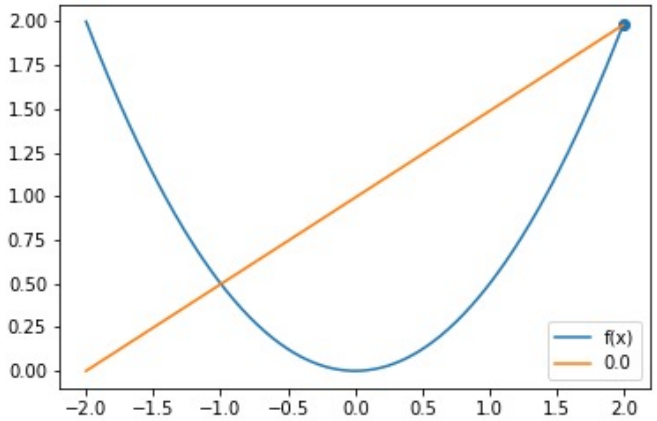
Convex Function

We say f is a **convex function** if its **epigraph** (the set of points above the function, illustrated in Figure 8.4a) defines a convex set. Equivalently, a function $f(\boldsymbol{x})$ is called convex if it is defined on a convex set and if, for any $\boldsymbol{x}, \boldsymbol{y} \in \mathcal{S}$, and for any $0 \leq \lambda \leq 1$, we have

$$f(\lambda \boldsymbol{x} + (1-\lambda)\boldsymbol{y}) \leq \lambda f(\boldsymbol{x}) + (1-\lambda) f(\boldsymbol{y}) \tag{8.7}$$

Y-axis value that fell between the domain from x and y

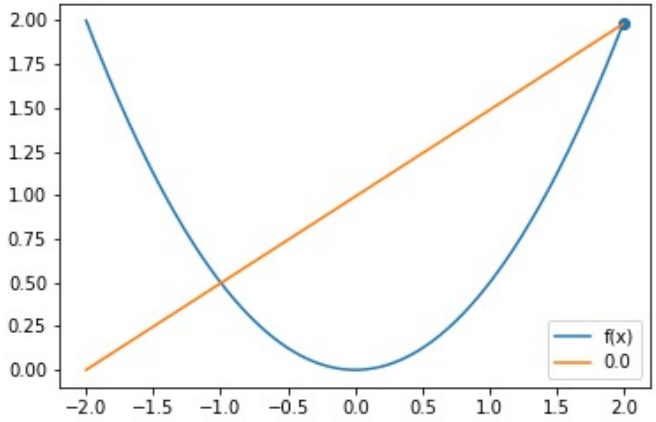
Straight line between (x, f(x)) and (y, f(y)) as a function of λ



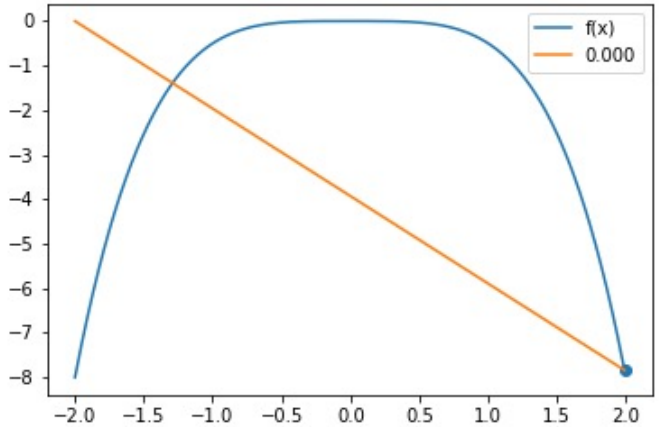
x = min x; y = max x

Convex Function

$$f(\lambda \boldsymbol{x} + (1-\lambda)\boldsymbol{y}) \leq \lambda f(\boldsymbol{x}) + (1-\lambda) f(\boldsymbol{y})$$



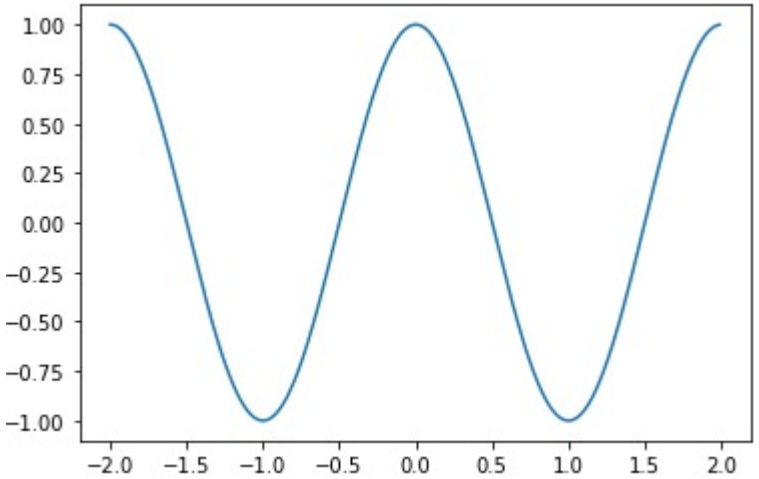
Convex



Concave

Convex Function

$$f(\lambda \boldsymbol{x} + (1-\lambda)\boldsymbol{y}) \leq \lambda f(\boldsymbol{x}) + (1-\lambda) f(\boldsymbol{y})$$



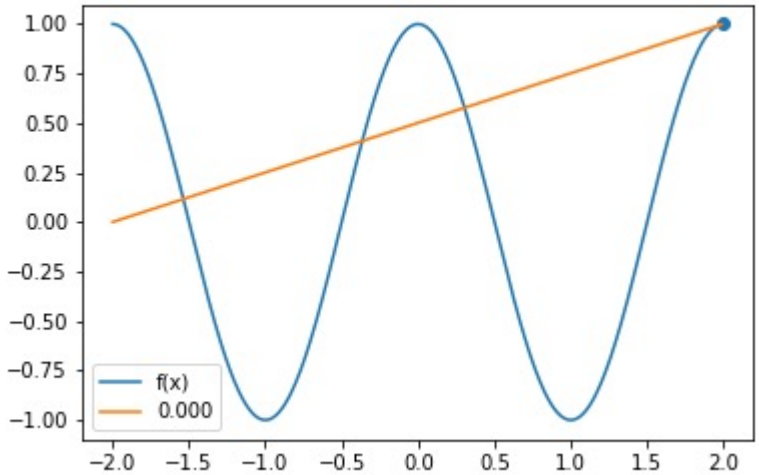
Is this graph convex or concave??
Choose your answer in the next slide.

Is the graph convex or not?

Convex Function

We say f is a **convex function** if its **epigraph** (the set of points above the function, illustrated in Figure 8.4a) defines a convex set. Equivalently, a function $f(\boldsymbol{x})$ is called convex if it is defined on a convex set and if, for any $\boldsymbol{x}, \boldsymbol{y} \in \mathcal{S}$, and for any $0 \leq \lambda \leq 1$, we have

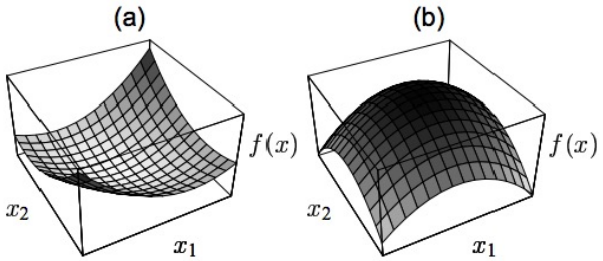
$$f(\lambda \boldsymbol{x} + (1-\lambda)\boldsymbol{y}) \leq \lambda f(\boldsymbol{x}) + (1-\lambda) f(\boldsymbol{y}) \tag{8.7}$$



Not Convex

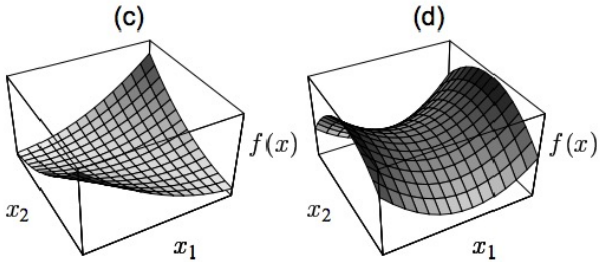
Type of Convex Function

Strictly convex



Strictly concave

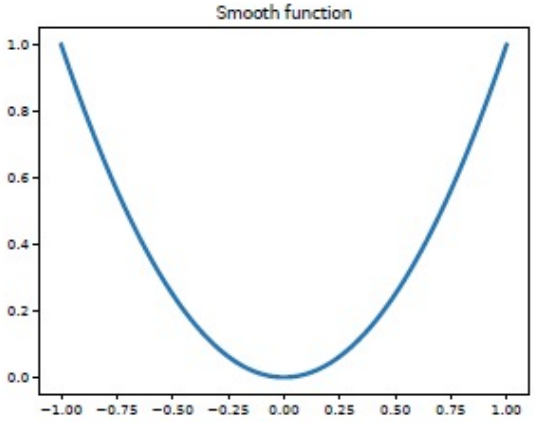
Convex but not strictly



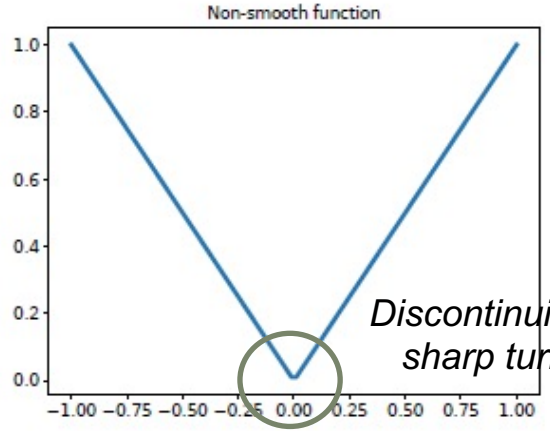
Neither convex nor concave → saddle point

Figure 8.6: The quadratic form $f(\boldsymbol{x}) = \boldsymbol{x}^{\mathsf{T}}\mathbf{A}\boldsymbol{x}$ in 2d. (a) $\mathbf{A}$ is positive definite, so f is convex. (b) $\mathbf{A}$ is negative definite, so f is concave. (c) $\mathbf{A}$ is positive semidefinite but singular, so f is convex, but not strictly. Notice the valley of constant height in the middle. (d) $\mathbf{A}$ is indefinite, so f is neither convex nor concave. The stationary point in the middle of the surface is a saddle point. From Figure 5 of [She94].

Smooth and Non-Smooth Optimization



(a)



Discontinuity / sharp turn

(b)

$$|f(x_1) - f(x_2)| \leq L|x_1 - x_2|$$

Lipschitz constant – quantify the degree of smoothness

First Order Method

- Gradient descent
- Step size/ learning rate
- Convergence rate
- Momentum Method

First Order Method

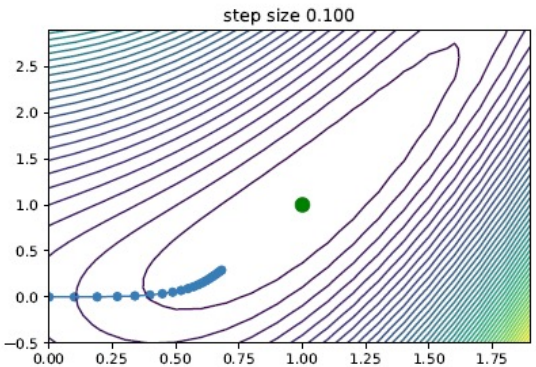
- Gradient descent
- Step size/ learning rate
- Convergence rate
- Momentum Method

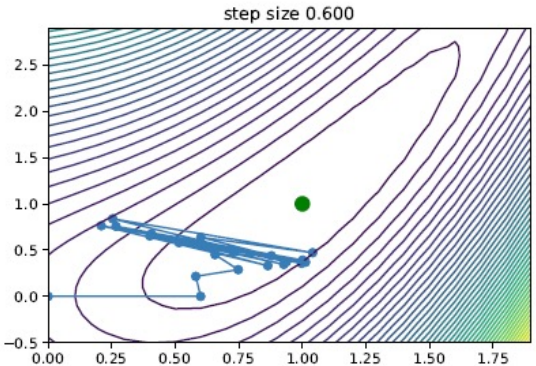
$$\theta_{t+1} = \theta_t + \rho_t d_t$$

Updated parameter

Step size

Descent direction





steepest descent will have global convergence iff the step size satisfies

$$\rho < \frac{2}{\lambda_{\max}(\mathbf{A})}$$

$$\mathcal{L}(\theta) = \tfrac{1}{2}\theta^{\mathsf{T}}\mathbf{A}\theta + b^{\mathsf{T}}\theta + c \text{ with } \mathbf{A} \succeq \mathbf{0}.$$

λmax = max eigenvalue

First Order Method

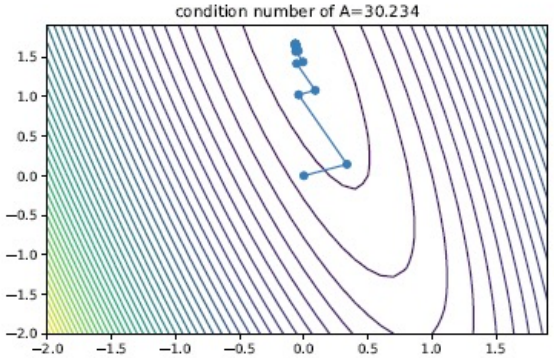
- Gradient descent
- Step size/ learning rate
- Convergence rate, μ
- Momentum Method

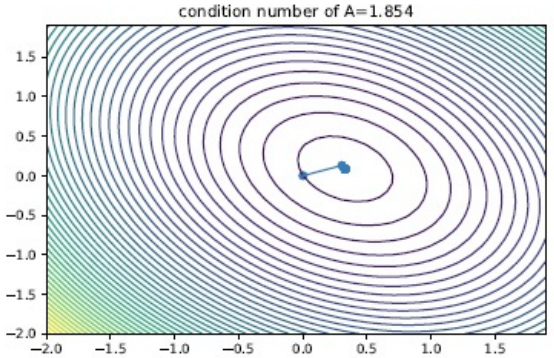
$$\mathcal{L}(\theta) = \frac{1}{2}\theta^{\mathsf{T}}\mathbf{A}\theta + b^{\mathsf{T}}\theta + c \text{ with } \mathbf{A} \succeq \mathbf{0}.$$

$$\mu = \left(\frac{\lambda_{\max} - \lambda_{\min}}{\lambda_{\max} + \lambda_{\min}}\right)^2$$

$$\mu = \left(\frac{\kappa-1}{\kappa+1}\right)^2, \text{ where } \kappa = \frac{\lambda_{\max}}{\lambda_{\min}} \text{ is the condition number of } \mathbf{A}.$$

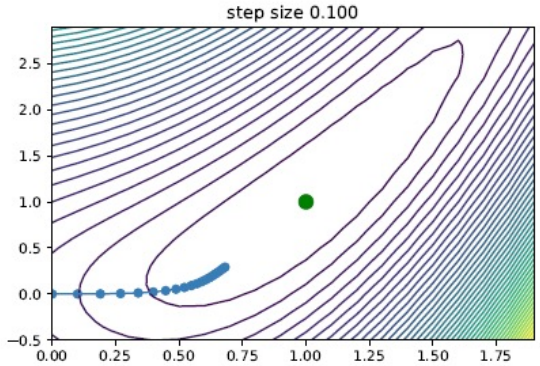
The condition number measures how “skewed” the space is, in the sense of being far from a symmetrical “bowl”





First Order Method

- Gradient descent
- Step size/ learning rate
- Convergence rate
- Momentum Method



Thinking like a ball rolling downward. At flat surface, it roll down slowly. At sharp region, roll down faster.

momentum

$$\boldsymbol{m}_t = \beta \boldsymbol{m}_{t-1} + \boldsymbol{g}_{t-1}$$

$$\boldsymbol{\theta}_t = \boldsymbol{\theta}_{t-1} - \rho_t \boldsymbol{m}_t$$

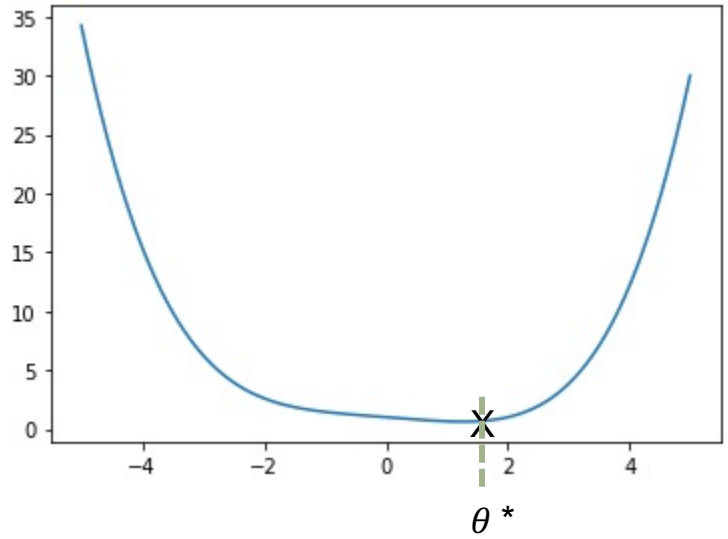
Normally β=0.9, if β = 0 → gradient descent

First Order Method – Line Search

Consider we want to minimize this loss function, $f(\theta) = \frac{1}{20}\theta^4 - \frac{2}{5}\theta + 1$

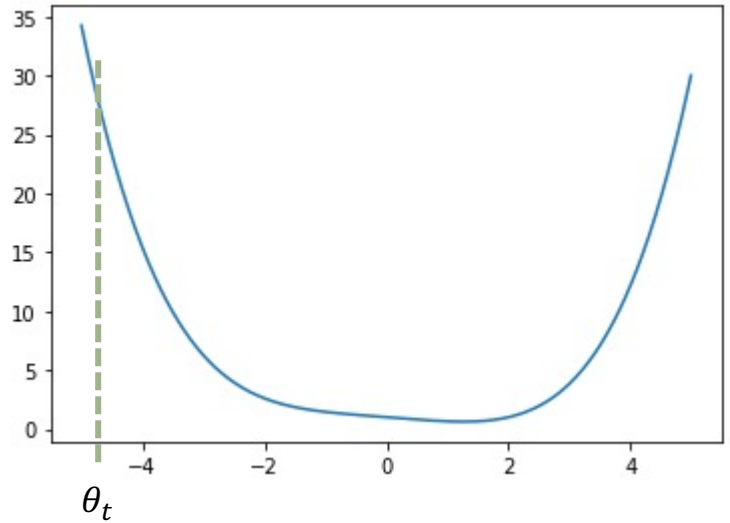
We know the min point is at θ * = 1.5
(compute f'(θ) = 0)

What if we use line search (iterative method to find θ *)



First Order Method – Line Search

Start with random number, θ_t

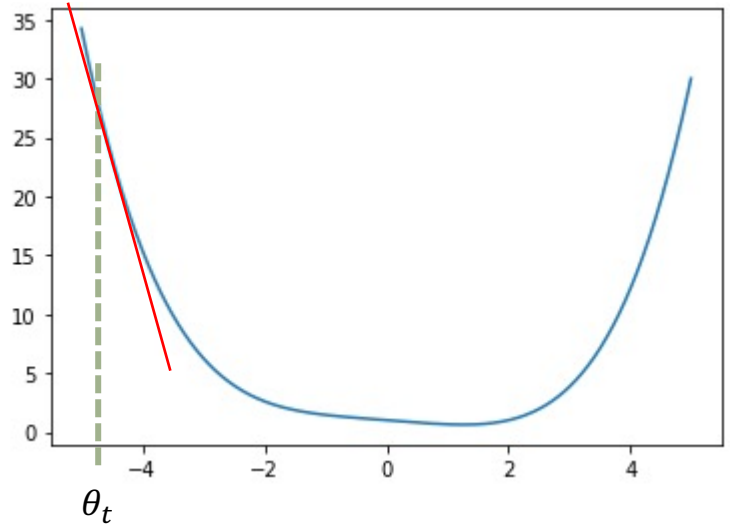


First Order Method – Line Search

Start with random number, θ_t

Compute the gradient at x_k

$$f'(\theta_t) = \frac{f(\theta) - f(\theta_t)}{\theta - \theta_t}$$



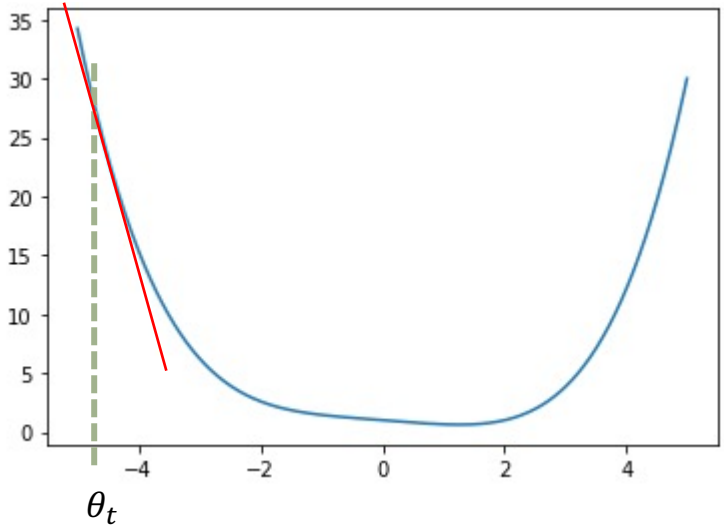
First Order Method – Line Search

Start with random number, θ_k

Compute the gradient at θ_k

$$f'(\theta_t) = \frac{f(\theta) - f(\theta_t)}{\theta - \theta_t}$$

If $f'(\theta_t)$ is negative, move θ_t to the right



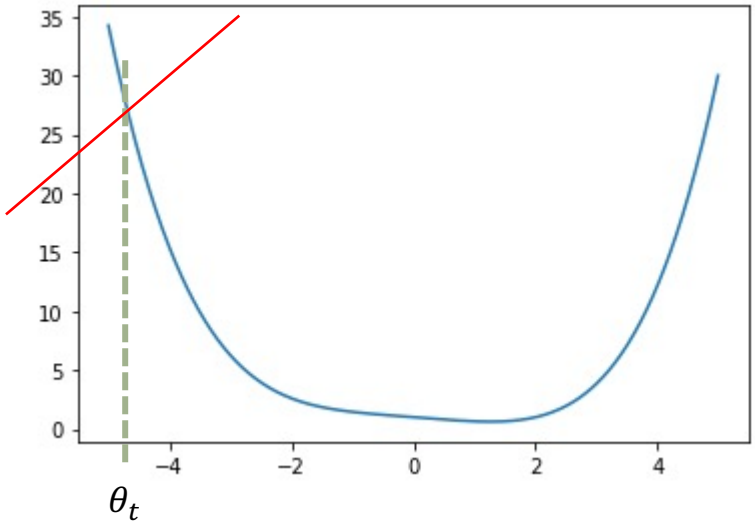
First Order Method – Line Search

Start with random number, θ_k

Compute the gradient at x_k

$$f'(\theta_t) = \frac{f(\theta) - f(\theta_t)}{\theta - \theta_t}$$

If $f'(\theta_t)$ is negative, move θ_t to the right
If $f'(\theta_t)$ is positive, move θ_t to the left



First Order Method – Line Search

Start with random number, θ

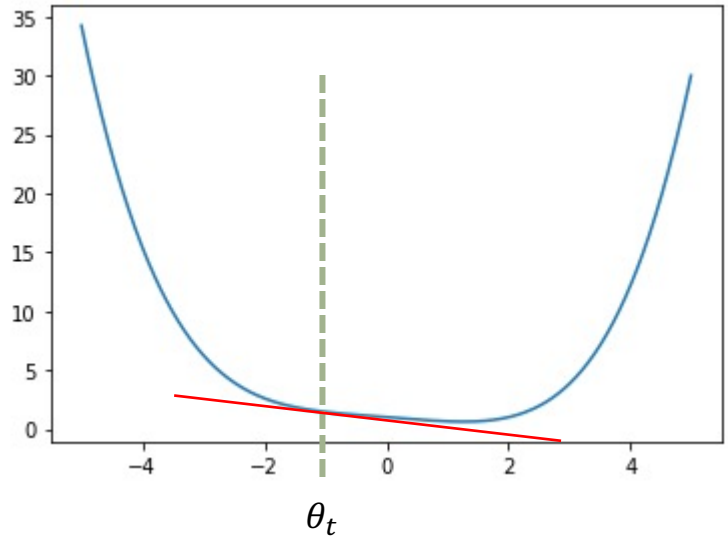
Compute the gradient at x_k

$$f'(\theta_t) = \frac{f(\theta) - f(\theta_t)}{\theta - \theta_t}$$

If $f'(\theta_t)$ is negative, move x_k to the right
If $f'(\theta_t)$ is positive, move x_k to the left

$$\theta_{t+1} = \theta_t - \alpha f'(\theta_t)$$

Large step size, α will overshoot
Small step size, α will be very slow



First Order Method – Line Search

Slow to converge

Second Order Method – Newton Method

Approximate with non linear graph

Compute the second derivative at θ_k

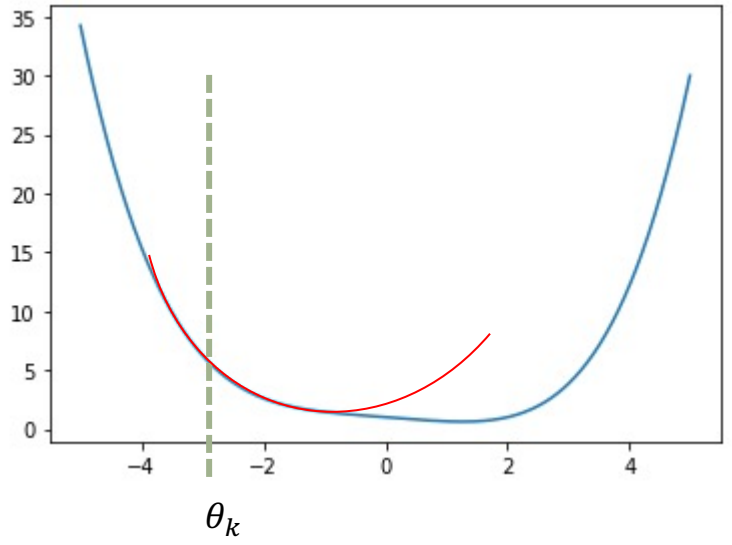
$$f''(\theta_k) = \frac{f(\theta) - f(\theta_k)}{\theta - \theta_k}$$

$$\theta_{t+1} = \theta_k - \alpha f'(\theta_k)$$

Using some algebra, the update formula is

$$\theta_{t+1} = \theta_k - \alpha \frac{1}{f''(\theta_k)} f'(\theta_k)$$

- Faster convergence



Second Order Method – Newton Method

Higher dimension

$$\theta_{t+1} = \theta_t - \rho_t \mathbf{H}_t^{-1} g_t$$

where

$$\mathbf{H}_t \triangleq \nabla^2 \mathcal{L}(\theta)|_{\theta_t} = \nabla^2 \mathcal{L}(\theta_t) = \mathbf{H}(\theta_t)$$

H = Hessian matrix
ρ = step size
g_t = gradient

$$\mathbf{H}_f = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \, \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \, \partial x_n} \\ \frac{\partial^2 f}{\partial x_2 \, \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} & \cdots & \frac{\partial^2 f}{\partial x_2 \, \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n \, \partial x_1} & \frac{\partial^2 f}{\partial x_n \, \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{bmatrix},$$

Stochastic Gradient Descent

Batch Gradient Descent

$$f(\theta) = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2$$

$$\theta_{t+1} = \theta_t - \alpha\, f'(\theta)$$

Need to compute for all data,
if you have sample size, m = 1million
Very slow to update θ

Stochastic Gradient Descent

$$cost\,(\theta, (x^{(i)}, y^{(i)})) = \frac{1}{2}(h_\theta(x^{(i)}) - y^{(i)})^2$$

1. Random Shuffle Data
2. Repeat {
   for i = 1, ... , m {

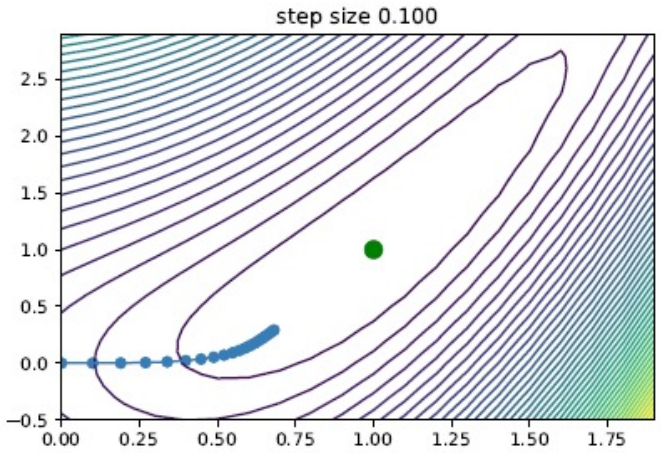
$$\theta_{t+1} = \theta_t - \alpha\, \frac{\partial}{\partial \theta_t} cost(\theta, (x^{(i)}, y^{(i)}))$$

   }
}

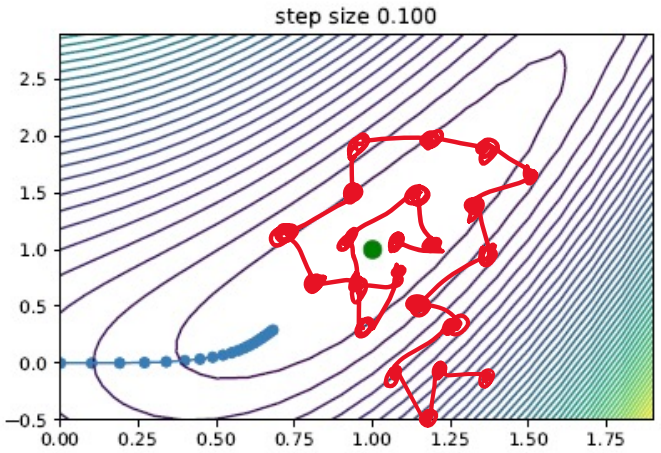
Update θ using only one data point.
Faster

Stochastic Gradient Descent

Batch Gradient Descent



Stochastic Gradient Descent



Decision Theory | Information Theory | Optimization

Linear Model

Notes inspired from: Shier Nee, SAW, Murphy PRML
Based on: Probabilistic Machine Learning by Kevin Murphy

Recap

- Probability: Univariate Model – Gaussian
- Probability: Multivariate Model – Gaussian
- Statistic – Maximum Likelihood Estimation, Regularizer
- Decision Theory - Bayesian
- Information Theory - Entropy
- Optimization - Stochastic Gradient Descent

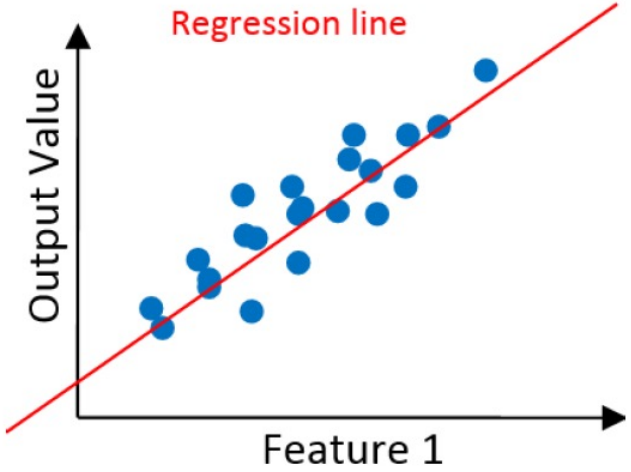
Outline

- Logistic Regression
- Linear Regression
- Generalized Linear Model

Logistic Regression

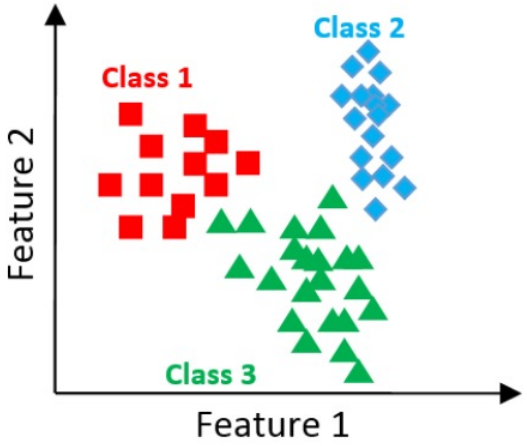
Regression:

predict continues output value



Classification:

predict discrete output value (class label)



Logistic Regression

- Logistic Regression is a method for classification

- For example:
  - Email: Spam / Not Spam?
  - Tumor: Malignant (Yes) / Benign (No)

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

0: “Negative Class”

1: “Positive Class”

Logistic Regression

Binary logistic regression often follows the following model

$$p(y|\boldsymbol{x};\boldsymbol{\theta}) = \mathrm{Ber}(y|\boldsymbol{\sigma}(\boldsymbol{w}^{\mathsf{T}}\boldsymbol{x} + b))$$

Bernoulli

sigmoid

weight

bias

$$f(k;p) = \begin{cases} p & \text{if } k = 1, \\ q = 1 - p & \text{if } k = 0. \end{cases}$$

$$p(y = 1|x;\theta) = \sigma(a) = \frac{1}{1 + e^{-a}}, \text{where } a = log\frac{p}{1-p}$$

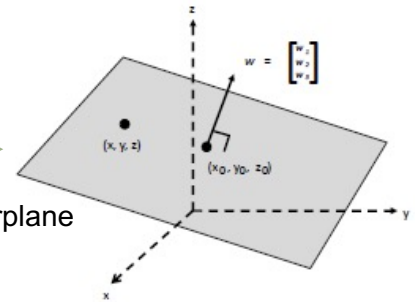
Bernoulli distribution: https://www.youtube.com/watch?v=8-c1P5fwhaE

Linear Classifier

The prediction can be written as

$$f(\boldsymbol{x}) = \mathbb{I}\left(p(y=1|\boldsymbol{x}) > p(y=0|\boldsymbol{x})\right) = \mathbb{I}\left(\log \frac{p(y=1|\boldsymbol{x})}{p(y=0|\boldsymbol{x})} > 0\right) = \mathbb{I}\left(a > 0\right)$$

$$f(x;\theta) = b + \boldsymbol{w}^{\boldsymbol{T}}\boldsymbol{x}$$



hyperplane

normal vector $w \in \mathbb{R}^D$ and an offset $b \in \mathbb{R}$ from the origin

This linear hyperplane separate 3d space into half → decision boundary

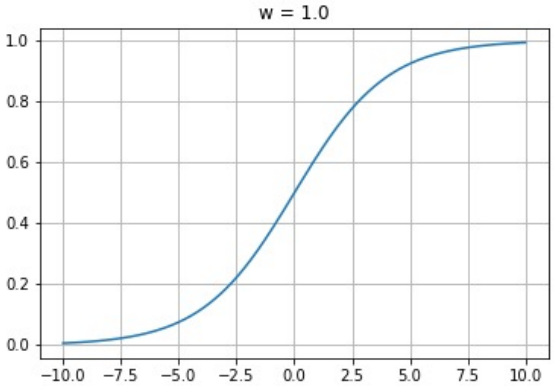
In general, there will be uncertainty about the correct class label, so we need to predict a probability distribution over labels, feed it to sigmoid function

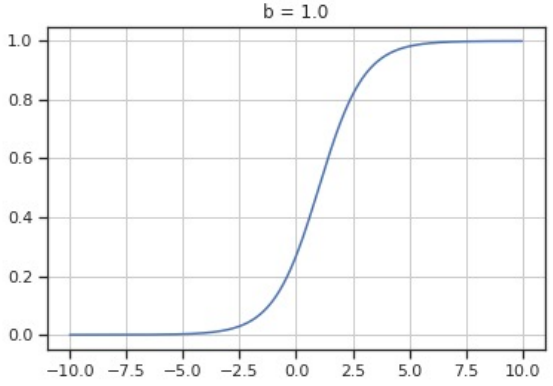
Sigmoid Function

Sigmoid function

$$\sigma(a) = \frac{1}{1 + e^{-a}}, \; where \; a = log\frac{p}{1-p} = b + \boldsymbol{w^T x}$$

Log loss and $\boldsymbol{w}$ determine the steepness of the sigmoid function





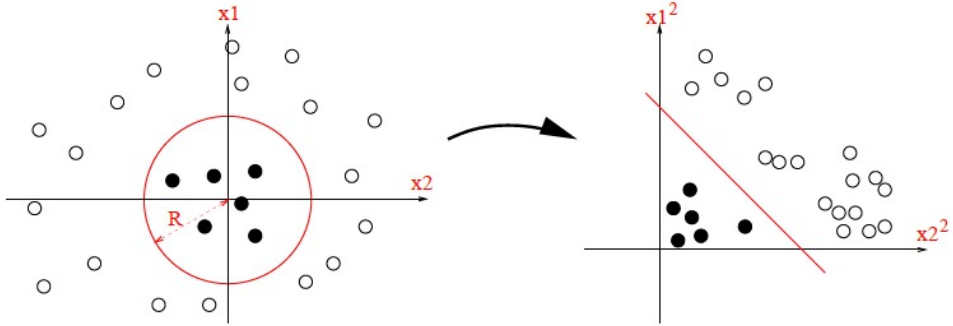
Non Linear Classifier

Transform input features in suitable way

$$\phi(x_1, x_2) = [1, x_1^2, x_2^2]$$

$\boldsymbol{w} = [-R^2, 1, 1]$. Then $\boldsymbol{w}^\mathsf{T}\boldsymbol{\phi}(\boldsymbol{x}) = x_1^2 + x_2^2 - R^2$

Decision boundary (where f(x) = 0) defines a circle with radius R



Logistic Regression – Cost Function

Maximize Maximum Likelihood Estimation / Minimize Negative Log Likelihood

$$\begin{aligned}
\mathrm{NLL}(\boldsymbol{w}) &= -\frac{1}{N}\log p(\mathcal{D}|\boldsymbol{w}) = -\frac{1}{N}\log\prod_{n=1}^{N}\mathrm{Ber}(y_n|\mu_n) \\
&= -\frac{1}{N}\sum_{n=1}^{N}\log[\mu_n^{y_n}\times(1-\mu_n)^{1-y_n}] \\
&= -\frac{1}{N}\sum_{n=1}^{N}[y_n\log\mu_n + (1-y_n)\log(1-\mu_n)] \\
&= \frac{1}{N}\sum_{n=1}^{N}\mathbb{H}(y_n,\mu_n)
\end{aligned}$$

No of Sample

probability

$$\mathbb{H}(p,q) = -[p\log q + (1-p)\log(1-q)]$$

Binary cross-entropy

Logistic Regression – Cost Function

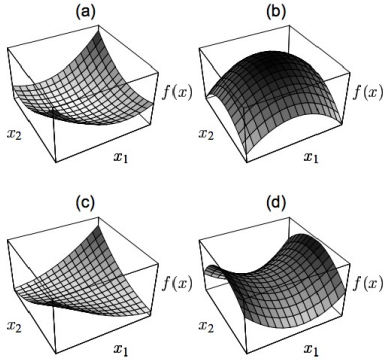
To find the MLE, we must solve

$$\nabla_{\boldsymbol{w}} \mathrm{NLL}(\boldsymbol{w}) = \boldsymbol{g}(\boldsymbol{w}) = \boldsymbol{0}$$

$$\begin{aligned}
\nabla_{\boldsymbol{w}} \mathrm{NLL}(\boldsymbol{w}) &= -\frac{1}{N}\sum_{n=1}^{N}[y_n(1-\mu_n)\boldsymbol{x}_n - (1-y_n)\mu_n\boldsymbol{x}_n] \\
&= -\frac{1}{N}\sum_{n=1}^{N}[y_n\boldsymbol{x}_n - y_n\boldsymbol{x}_n\mu_n - \boldsymbol{x}_n\mu_n + y_n\boldsymbol{x}_n\mu_n)] \\
&= \frac{1}{N}\sum_{n=1}^{N}(\mu_n - y_n)\boldsymbol{x}_n
\end{aligned}$$

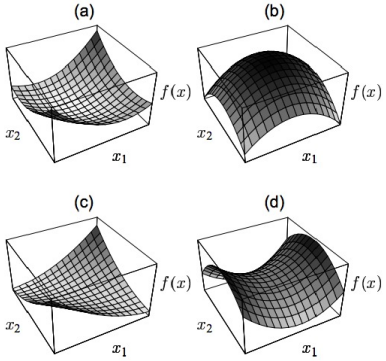
Error

Here, we can see that the gradient is weighed by the error for each input



Check convexity

Logistic Regression – Cost Function



Check convexity

Ensure NLL has bowl shape (global minimum)
→ check Hessian matrix

More formally, we must prove that the Hessian is positive semi-definite, which we now do. (See Chapter 7 for relevant background information on linear algebra.) One can show that the Hessian is given by

$$\mathbf{H}(\boldsymbol{w}) = \nabla_{\boldsymbol{w}} \nabla_{\boldsymbol{w}}^{\mathsf{T}} \mathrm{NLL}(\boldsymbol{w}) = \frac{1}{N} \sum_{n=1}^{N} (\mu_n(1-\mu_n)\boldsymbol{x}_n)\boldsymbol{x}_n^{\mathsf{T}} = \frac{1}{N}\mathbf{X}^{\mathsf{T}}\mathbf{S}\mathbf{X} \tag{10.23}$$

where

$$\mathbf{S} \triangleq \mathrm{diag}(\mu_1(1-\mu_1), \ldots, \mu_N(1-\mu_N)) \tag{10.24}$$

We see that $\mathbf{H}$ is positive definite, since for any nonzero vector $\boldsymbol{v}$, we have

$$\boldsymbol{v}^{\mathsf{T}}\mathbf{X}^{\mathsf{T}}\mathbf{S}\mathbf{X}\boldsymbol{v} = (\boldsymbol{v}^{\mathsf{T}}\mathbf{X}^{\mathsf{T}}\mathbf{S}^{\frac{1}{2}})(\mathbf{S}^{\frac{1}{2}}\mathbf{X}\boldsymbol{v}) = ||\boldsymbol{v}^{\mathsf{T}}\mathbf{X}^{\mathsf{T}}\mathbf{S}^{\frac{1}{2}}||_2^2 > 0 \tag{10.25}$$

This follows since $\mu_n > 0$ for all n, because of the use of the sigmoid function. Consequently the NLL is strictly convex. However, in practice, values of μ_n which are close to 0 or 1 might cause the Hessian to be close to singular. We can avoid this by using ℓ_2 regularization, as we discuss in Section 10.2.7.

Logistic Regression – Cost Function

Since we know the objective is convex (see Section 10.2.3.4), then one can show that this procedure will converge to the global optimum, provided we decay the learning rate at the appropriate rate (see Section 8.4.3). We can improve the convergence speed using variance reduction techniques such as SAGA (Section 8.4.5.2).

Our goal is to solve the following optimization problem

$$\hat{\boldsymbol{w}} \triangleq \underset{\boldsymbol{w}}{\operatorname{argmin}}\, \mathcal{L}(\boldsymbol{w})$$

where $\mathcal{L}(\boldsymbol{w})$ is the loss function, in this case the negative log likelihood:

$$\mathrm{NLL}(\boldsymbol{w}) = -\frac{1}{N}\sum_{n=1}^{N}\left[y_n \log \mu_n + (1-y_n)\log(1-\mu_n)\right]$$

where $\mu_n = \boldsymbol{\sigma}(a_n)$ is the probability of class 1, and $a_n = \boldsymbol{w}^{\mathsf{T}}\boldsymbol{x}_n$ is the log odds.

Logistic Regression – Optimizer

1. First order method
   - Stochastic Gradient Descent

Slow convergence, when gradient is small

2. Second order method
   - Newton Method (Iteratively reweighted least squares)

$$\theta_{t+1} = \theta_k - \alpha \frac{1}{f''(\theta_k)} f'(\theta_k)$$

Stochastic Gradient Descent

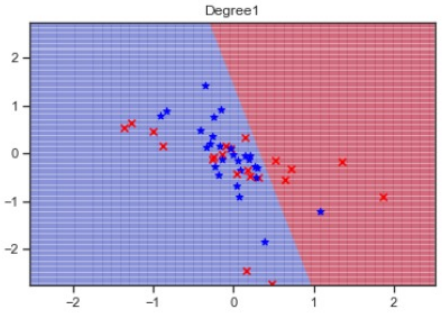
$$cost\left(\theta, (x^{(i)}, y^{(i)})\right) = \frac{1}{2}(h_\theta(x^{(i)}) - y^{(i)})^2$$

1. Random Shuffle Data
2. Repeat {
   for i = 1, ... , m {

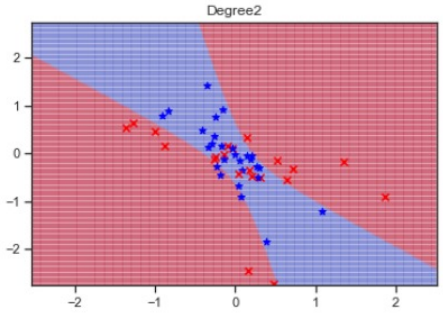
$$\theta_{t+1} = \theta_t - \alpha \frac{\partial}{\partial \theta_t} cost\left(\theta, (x^{(i)}, y^{(i)})\right)$$

   }
}

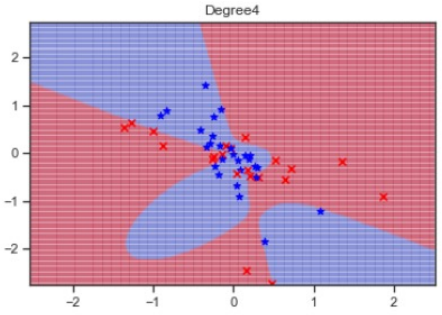
Logistic Regression – Overfitting



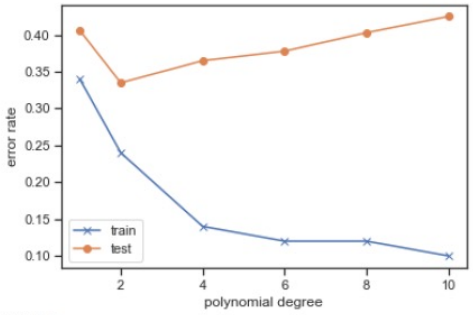
$\hat{w} = [0.51291712, 0.11866937]$



$\hat{w} = [2.27510513, 0.05970325, 11.84198867, 15.40355969, 2.51242311]$



$\hat{w} = [-3.07813766, \cdots, -59.03196044, 51.77152431, 10.25054164]$



(d)

See any trend?

As degree increases,
w increase / decrease?

Logistic Regression – Overfitting

Reduce Overfitting

→ Do not let weight to grow

→ Add regularizer to as penalty

$$\mathcal{L}(\boldsymbol{w}) = \mathrm{NLL}(\boldsymbol{w}) + \lambda ||\boldsymbol{w}||_2^2 \tag{10.47}$$

where $||\boldsymbol{w}||_2^2 = \sum_{d=1}^{D} w_d^2$ and $\lambda = 1/C$. This is called ℓ_2 **regularization** or **weight decay**. The larger the value of λ, the more the parameters are penalized for being "large" (deviating from the zero-mean prior), and thus the less flexible the model. See Figure 10.6 for an illustration.

Logistic Regression – Overfitting

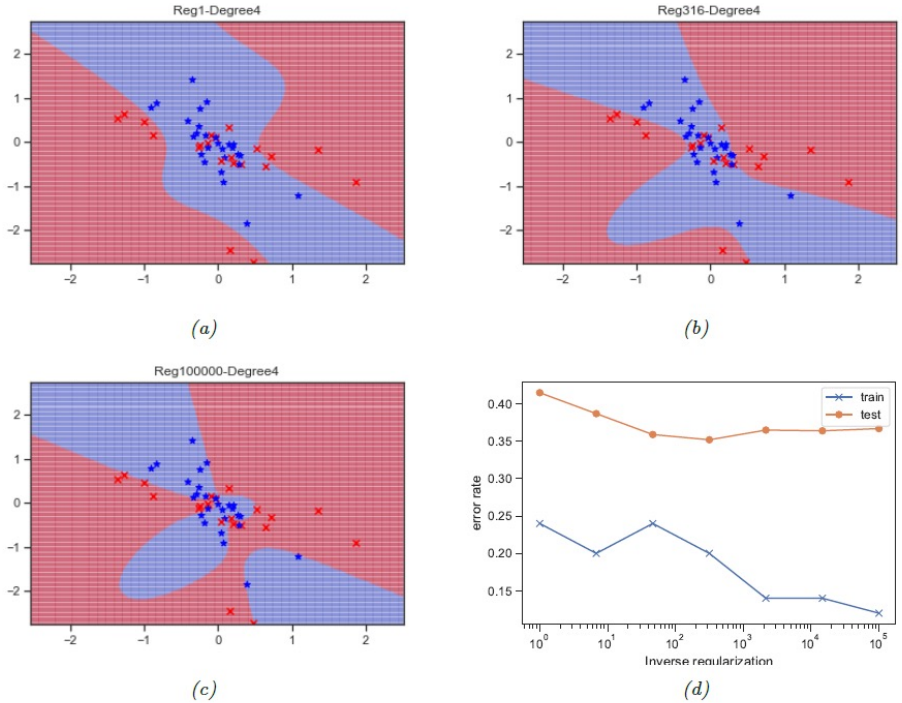


Figure 10.6: Weight decay with variance C applied to two-class, two-dimensional logistic regression problem with a degree 4 polynomial. (a) $C = 1$. (b) $C = 316$. (c) $C = 100,000$. (d) Train and test error vs C. Generated by code at figures.probml.ai/book1/10.6.

Big λ / Small C → less flexible

Logistic Regression – Overfitting

	Binary Logistic Regression	Multiple Logistic Regression
Probability	$p(y\|x;\theta) = \mathrm{Ber}(y\|\sigma(w^\mathsf{T}x + b))$	$p(y\|x;\theta) = \prod_{c=1}^{C} \mathrm{Ber}(y_c\|\sigma(w_c^\mathsf{T}x))$
Activation function	σ = sigmoid activation	σ = softmax activation
Cost function	$\mathbb{H}(p,q) = -\left[p\log q + (1-p)\log(1-q)\right]$	$\mathbb{H}(p,q) = -\sum_{c=1}^{C} p_c \log q_c$
Gradient	$\nabla_w \mathrm{NLL}(w) = \frac{1}{N}(1_N^\mathsf{T}(\mathrm{diag}(\mu - y)\mathbf{X}))^\mathsf{T}$	$g(w) = \frac{1}{N}\sum_{n=1}^{N} x_n(\mu_n - y_n)^\mathsf{T}$
Hessian	$\mathbf{H}(w) = \nabla_w \nabla_w^\mathsf{T} \mathrm{NLL}(w) = \frac{1}{N}\sum_{n=1}^{N}(\mu_n(1-\mu_n)x_n)x_n^\mathsf{T} = \frac{1}{N}\mathbf{X}^\mathsf{T}\mathbf{S}\mathbf{X}$	$\mathbf{H}(w) = \frac{1}{N}\sum_{n=1}^{N}(\mathrm{diag}(\mu_n) - \mu_n\mu_n^\mathsf{T}) \otimes (x_n x_n^\mathsf{T})$

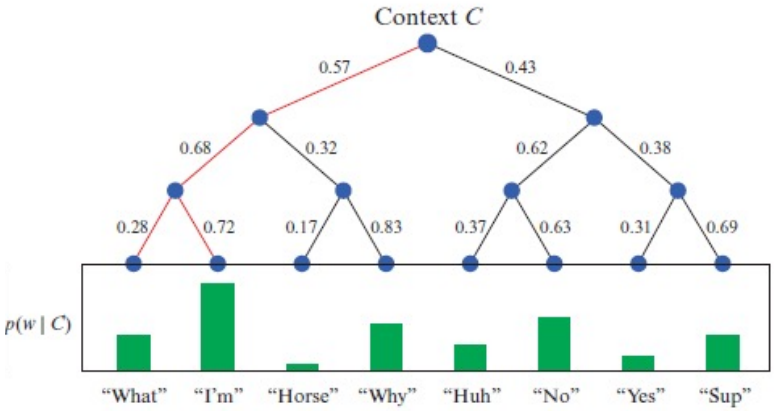
Handling large number of classes

$$\mathbb{H}(\boldsymbol{p}, \boldsymbol{q}) = -\sum_{c=1}^{C} p_c \log q_c$$

Using regular softmax function, when the number of classes, *C* increases, computational cost to compute H increases

To facilitate this, we can use hierarchy softmax

The idea behind decomposing the output layer to a binary tree was to reduce the complexity to obtain probability distribution



Handling imbalance class

- More attention on more ‘common’ dataset
- Less attention on ‘rare’ dataset

Approach
- Resample the data – Oversample / Undersample

Handling outlier

Use Mixture model for the likelihood

$$p(y|\boldsymbol{x}) = \pi \text{Ber}(y|0.5) + (1-\pi)\text{Ber}(y|\sigma(\boldsymbol{w}^\mathsf{T}\boldsymbol{x}))$$

Otherwise, is generated using conditional model

y is generated uniformly at random with probability π

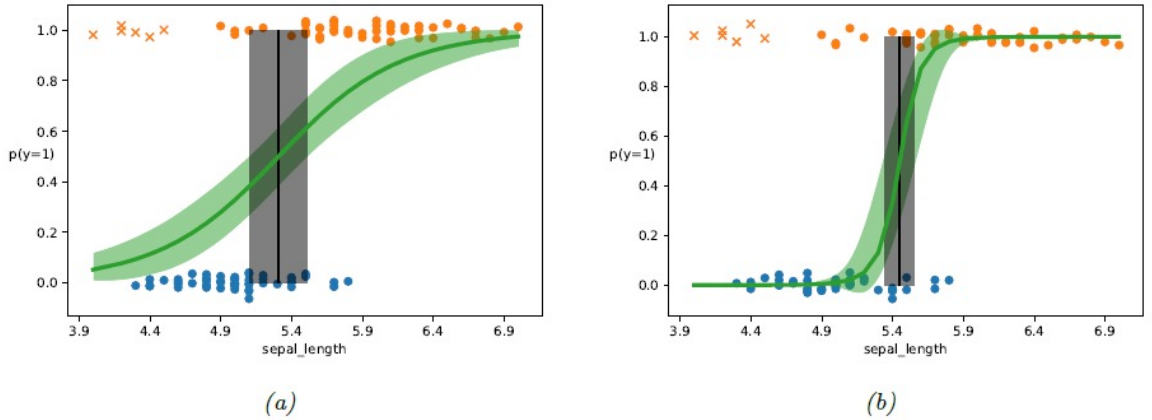


Figure 10.10: (a) Logistic regression on some data with outliers (denoted by x). Training points have been (vertically) jittered to avoid overlapping too much. Vertical line is the decision boundary, and its posterior credible interval. (b) Same as (a) but using robust model, with a mixture likelihood. Adapted from Figure 4.13 of [Mar18]. Generated by code at figures.probml.ai/book1/10.10.

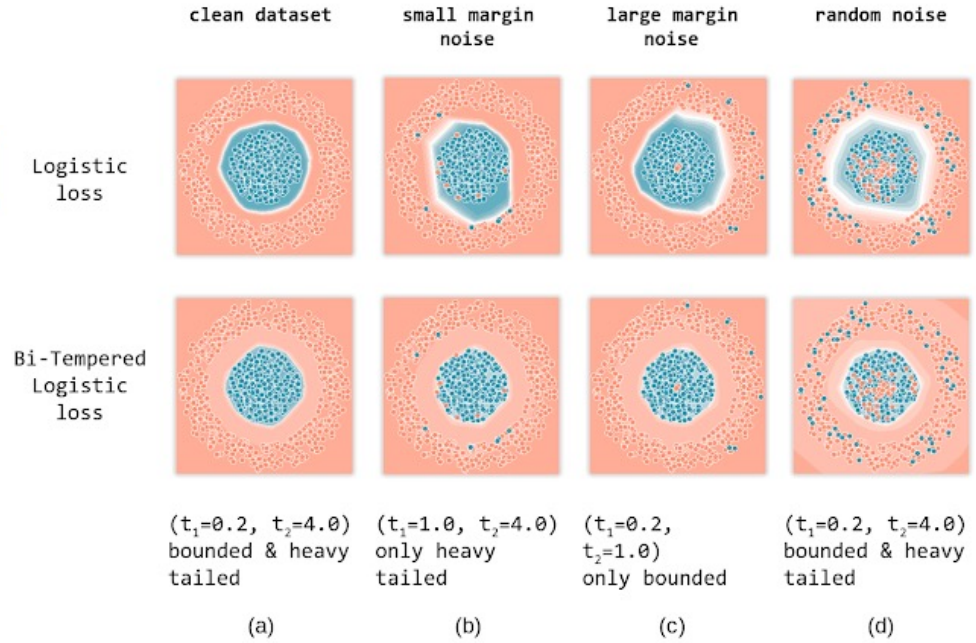
Handling outlier – Bi tempered loss

Tempered cross-entropy

$$\mathcal{L}(y, \hat{y}) = \sum_c \left[y_c (\log_{t_1} y_c - \log_{t_1} \hat{y}_c) - \frac{1}{2 - t_1} (y_c^{2-t_1} - \hat{y}_c^{2-t_1}) \right]$$

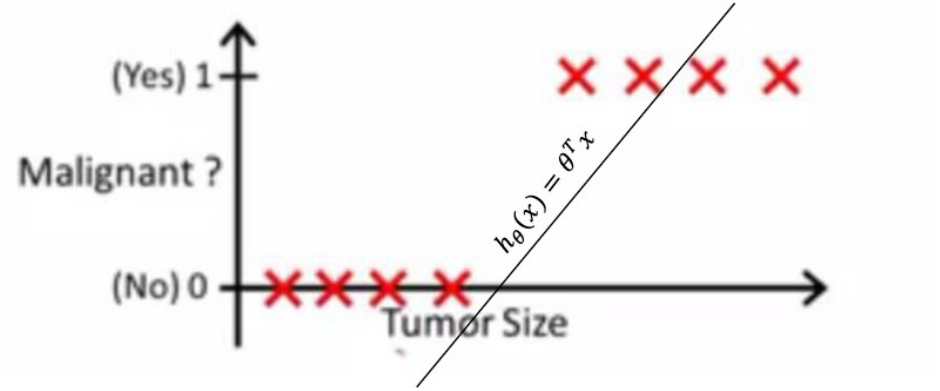
Tempered softmax

$$\hat{y}_c = \exp_{t_2}(a_c - \lambda_{t_2}(a))$$

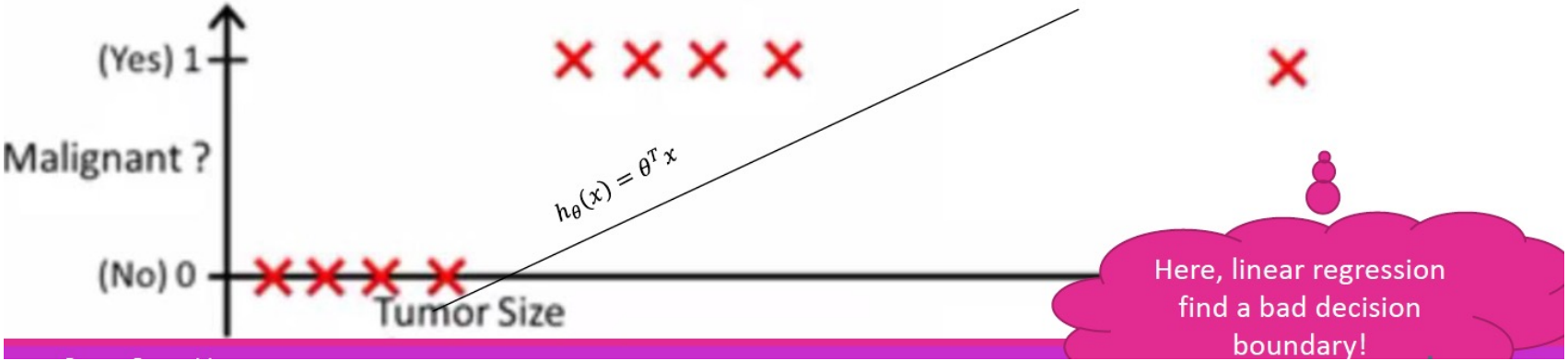


Logistic vs. bi-tempered logistic loss: (a) noise-free labels, (b) small-margin label noise, (c) large-margin label noise, and (d) random label noise. The temperature values (t_1, t_2) for the tempered loss are shown above each figure. We find that for each situation, the decision boundary recovered by training with the bi-tempered logistic loss function is better than before.

Linear Regression and Classification!!!



It is not suitable!
We know $y \in \{0,1\}$
while $-\infty \leq h_\theta(x) \leq \infty$



Here, linear regression find a bad decision boundary!

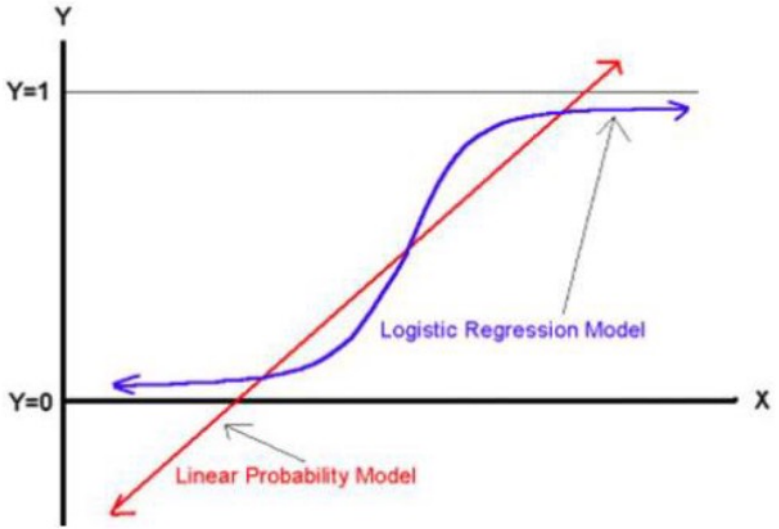
Logistic Regression

- In binary classification $y \in \{0,1\}$
- In Logistic regression, we want: $0 \leq h_\theta(x) \leq 1$

Logistic Regression: Hypothesis Representation

- We want: $0 \leq h_\theta(x) \leq 1$

Linear Regression:
$h_\theta(x) = \theta^T x$



Logistic Regression:
$h_\theta(x) = g(\theta^T x)$

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

Sigmoid function

Logistic Regression: Hypothesis Representation

- Interpretation of logistic model:

$0 \le h_\theta(x) \le 1$

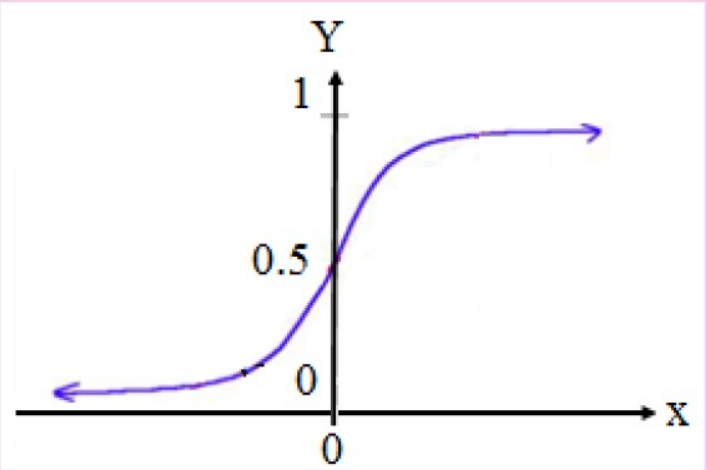
$h_\theta(x)$ = estimated probability that $y = 1$ on data x

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

$$P(y = 0|x;\theta) + P(y = 1|x;\theta) = 1$$

$$P(y = 0|x;\theta) = 1 - P(y = 1|x;\theta)$$

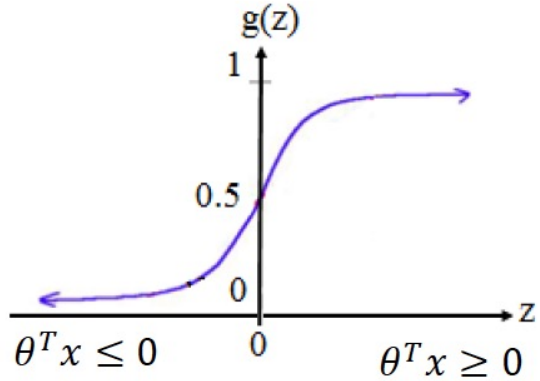
$P(y = 1|x;\theta)$



Logistic Regression: Decision Boundary

- Logistic regression

$$h_\theta(x) = g(\theta^T x) = \frac{1}{1+e^{-\theta^T x}}$$



- Suppose

If $h_\theta(x) \ge 0.5$, predict $y = 1$

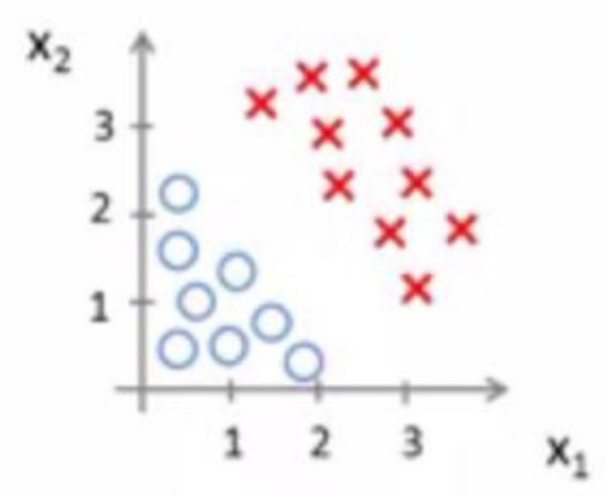
If $h_\theta(x) \le 0.5$, predict $y = 0$

$h_\theta(x) \le 0.5$ $\quad$ $h_\theta(x) \ge 0.5$

$\theta^T x$ is the decision boundary in Logistic Regression

Decision Boundary: Example I

- Linear boundary



$$h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2)$$

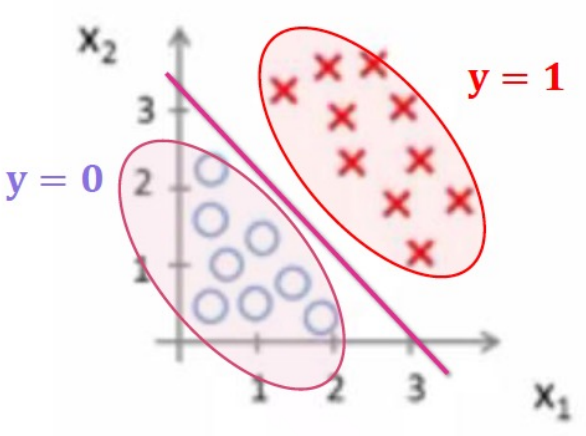
$$h_\theta(x) = g(-3 + x_1 + x_2)$$

$If \quad -3 + x_1 + x_2 \geq 0 \quad$ then predict $y = 1$

$If \quad -3 + x_1 + x_2 \leq 0 \quad$ then predict $y = 0$

Decision Boundary: Example I

- Linear boundary



$$h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2)$$

$$h_\theta(x) = g(-3 + x_1 + x_2)$$

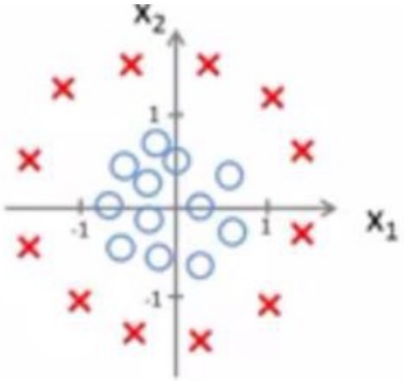
$If \quad -3 + x_1 + x_2 \geq 0 \quad$ then predict $y = 1$

$If \quad -3 + x_1 + x_2 \leq 0 \quad$ then predict $y = 0$

Decision Boundary is:
$-3 + x_1 + x_2 = 0$

Decision Boundary: Example II

- Non-Linear Decision boundary



$$h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1^2 + \theta_4 x_2^2)$$

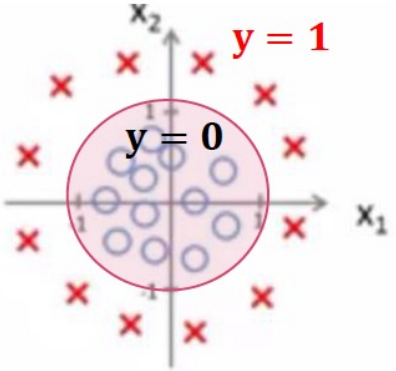
$$h_\theta(x) = g(-1 + x_1^2 + x_2^2)$$

$If \quad -1 + x_1^2 + x_2^2 \geq 0 \quad$ then predict $y = 1$

$If \quad -1 + x_1^2 + x_2^2 \leq 0 \quad$ then predict $y = 0$

Decision Boundary: Example II

- Non-Linear Decision boundary



$$h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1^2 + \theta_4 x_2^2)$$

$$h_\theta(x) = g(-1 + x_1^2 + x_2^2)$$

$If \quad -1 + x_1^2 + x_2^2 \geq 0$ then predict $y = 1$

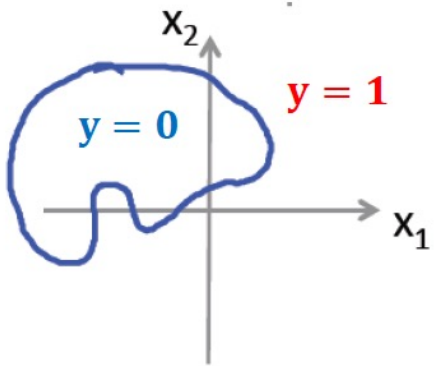
$If \quad -1 + x_1^2 + x_2^2 \leq 0$ then predict $y = 0$

Decision Boundary is: $x_1^2 + x_2^2 = 1$

Decision Boundary: Example III

- Other non-linear boundaries

$$h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1^2 + \theta_4 x_1^2 x_2 + \theta_5 x_1^2 x_2^2 + \theta_6 x_1^3 x_2 + \dots)$$



Logistic Regression: Cost function

- How to choose parameters θ?

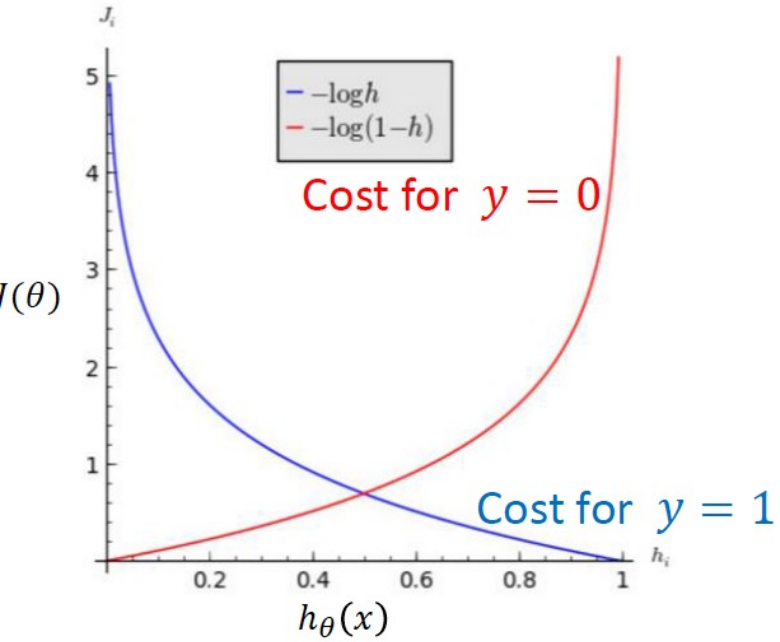
$$h_\theta(x) = \frac{1}{1 + e^{-\theta^T x}}$$

- For linear regression: we try to minimize

$$J(\theta) = \frac{1}{2m} \sum_{i=1}^{i=m} \left(h_\theta(x^{(i)}) - y^{(i)} \right)^2$$

- For logistic regression this cost function is not suitable. Why?

Logistic Regression: Cost function



What happen if

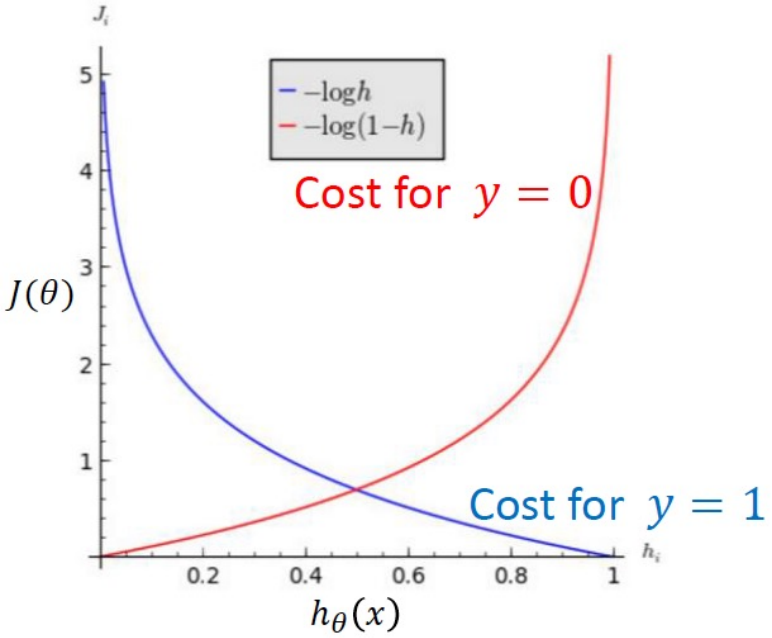
$h_\theta(x) = 0 \; and \; y = 0$?

$h_\theta(x) = 1 \; and \; y = 1$?

$h_\theta(x) = 0 \; and \; y = 1$?

$h_\theta(x) = 1 \; and \; y = 0$?

Logistic Regression: Cost function

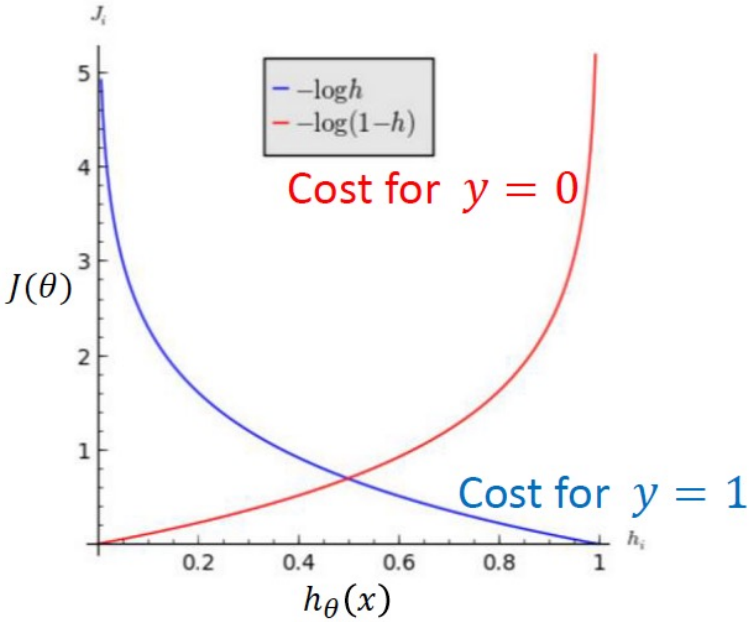


$$Cost(h_\theta(x), y) = \begin{cases} -\log(h_\theta(x)) & if\ y = 1 \\ -\log(1 - h_\theta(x)) & if\ y = 0 \end{cases}$$

Cost function for logistic regression

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m} Cost\left(h_\theta\left(x^{(i)}\right), y^{(i)}\right)$$

Logistic Regression: Cost function



$$Cost(h_\theta(x), y) = \begin{cases} -\log(h_\theta(x)) & if\ y = 1 \\ -\log(1 - h_\theta(x)) & if\ y = 0 \end{cases}$$

Cost function for logistic regression

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m} Cost\left(h_\theta\left(x^{(i)}\right), y^{(i)}\right)$$

It can be shown as:

$$J(\theta) = -\frac{1}{m}\left[\sum_{i=1}^{m} y^{(i)} \log\left(h_\theta\left(x^{(i)}\right)\right) + (1 - y^{(i)}) \log\left(1 - h_\theta(x)\right)^{(i)}\right]$$

Logistic Regression

- Hypothesis

$$h_\theta(x) = \frac{1}{1 + e^{-\theta^T x}}$$

- Cost function

$$J(\theta) = -\frac{1}{m}\left[\sum_{i=1}^{m} y^{(i)} \log\left(h_\theta\left(x^{(i)}\right)\right) + (1 - y^{(i)}) \log\left(1 - h_\theta(x)\right)^{(i)}\right]$$

$$(\textbf{\textit{This cost function is convex. Why (Excersise)}}\ ?)$$

- Goal: $minimize \;\; J(\theta)\ over\ \theta$

- How choose parameters θ ?

Logistic Regression

- $Minimize \;\; J(\theta) \;\; over \; \theta$

$$J(\theta) = -\frac{1}{m}\left[\sum_{i=1}^{m} y^{(i)} \log\left(h_\theta\left(x^{(i)}\right)\right) + (1 - y^{(i)}) \log\left(1 - h_\theta(x)\right)^{(i)}\right]$$

$$\frac{\partial}{\partial \theta_j} J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}\left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)x_j^{(i)}$$

$\frac{\partial}{\partial \theta_j} J(\theta) = 0$ → It does not have closed form solution → Gradient Descent

Gradient Descent for Logistic Regression

- Minimize $$J(\theta) = -\frac{1}{m}\left[\sum_{i=1}^{m} y^{(i)} \log\left(h_\theta\left(x^{(i)}\right)\right) + (1 - y^{(i)}) \log\left(1 - h_\theta(x)\right)^{(i)}\right]$$

Gradient Descent

Repeat {

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta)$$

(simultaneously update all θ_j)

}

Gradient Descent for Logistic regression

Repeat {

$$\theta_j := \theta_j - \alpha \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

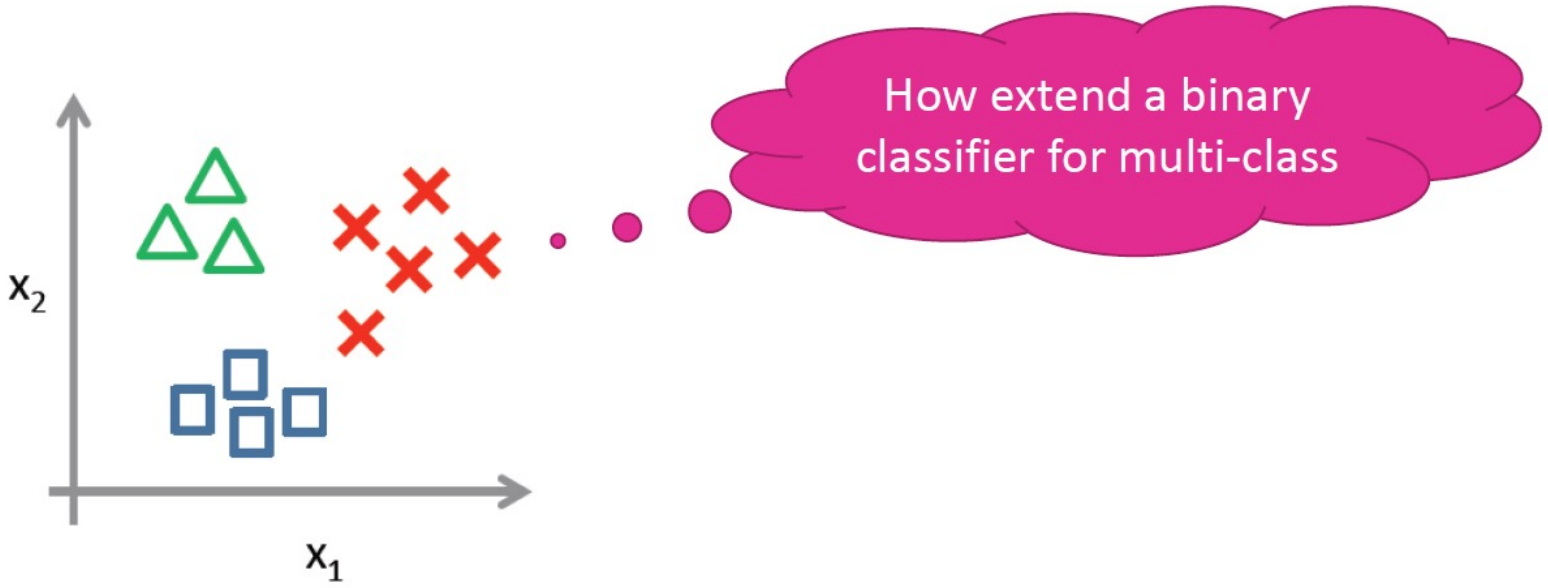
(simultaneously update all θ_j)

}

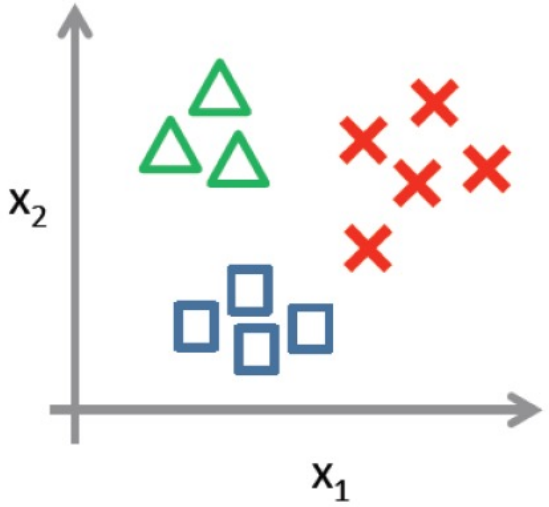
$$\frac{\partial}{\partial \theta_j} J(\theta) = -\frac{1}{m} \sum_{i=1}^{m} \left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right) x_j^{(i)}$$

Is it similar to linear regression ?!

Multi-class Classification



Multi-class Classification

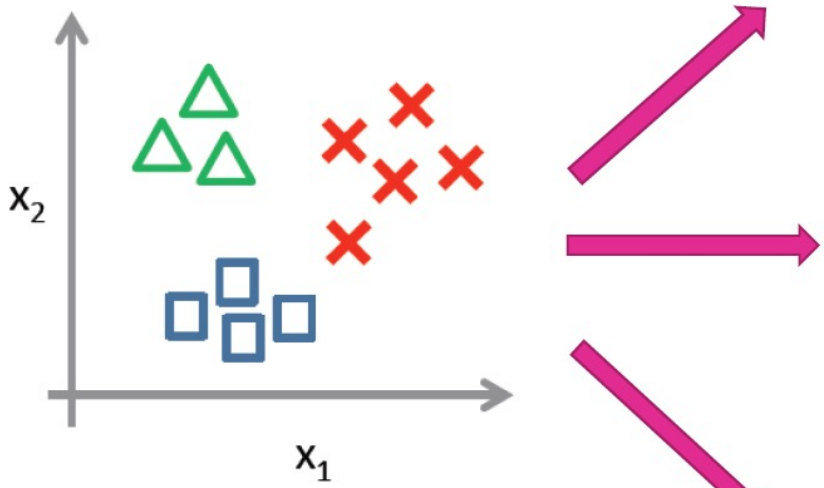


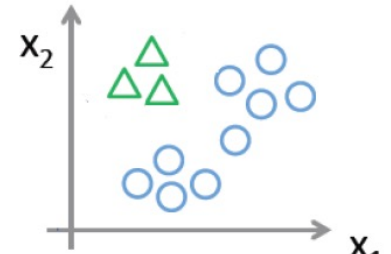
How extend a binary classifier for multi-class

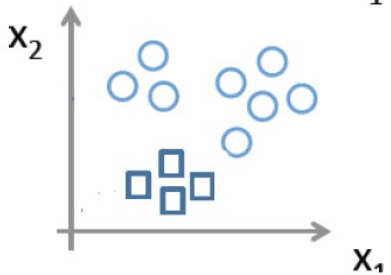
Generally any binary classifier can be extended to multi-class classifier via:

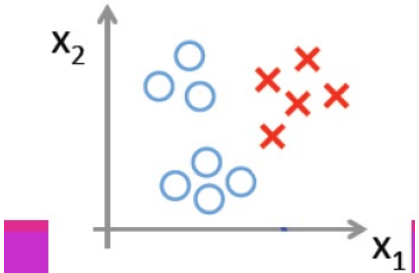
- One-versus-All (OVA)
- One-versus-One (OVO)

One-versus-All (OVA)









How many classifiers are needed?

One-versus-All (OVA)

Train a logistic regression classifier

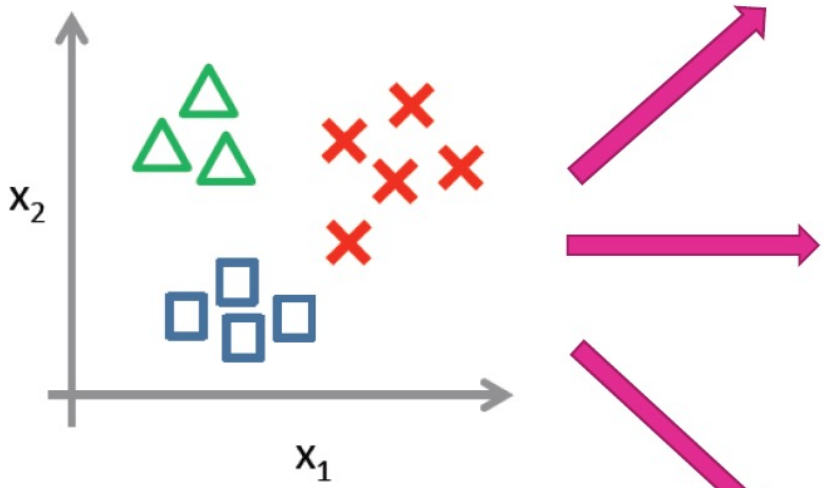
$$h_\theta^{(i)}(x) = P(y = 1|x; \theta) \qquad i = 1,2,3$$

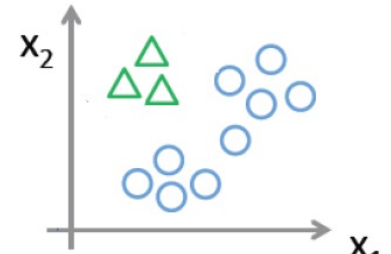
For a new input x :

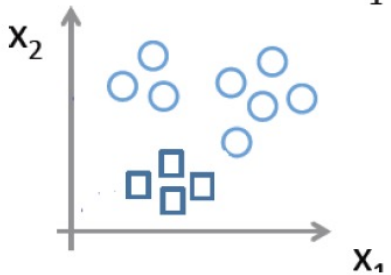
$$\text{Class(x)} = \underset{i}{argmax}\ h_\theta^{(i)}(x)$$

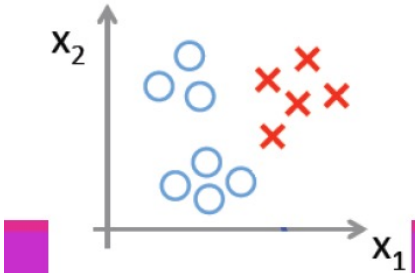
Leads to ambiguity if two classes with maximum probability have same probabilities

One-versus-All (OVA)



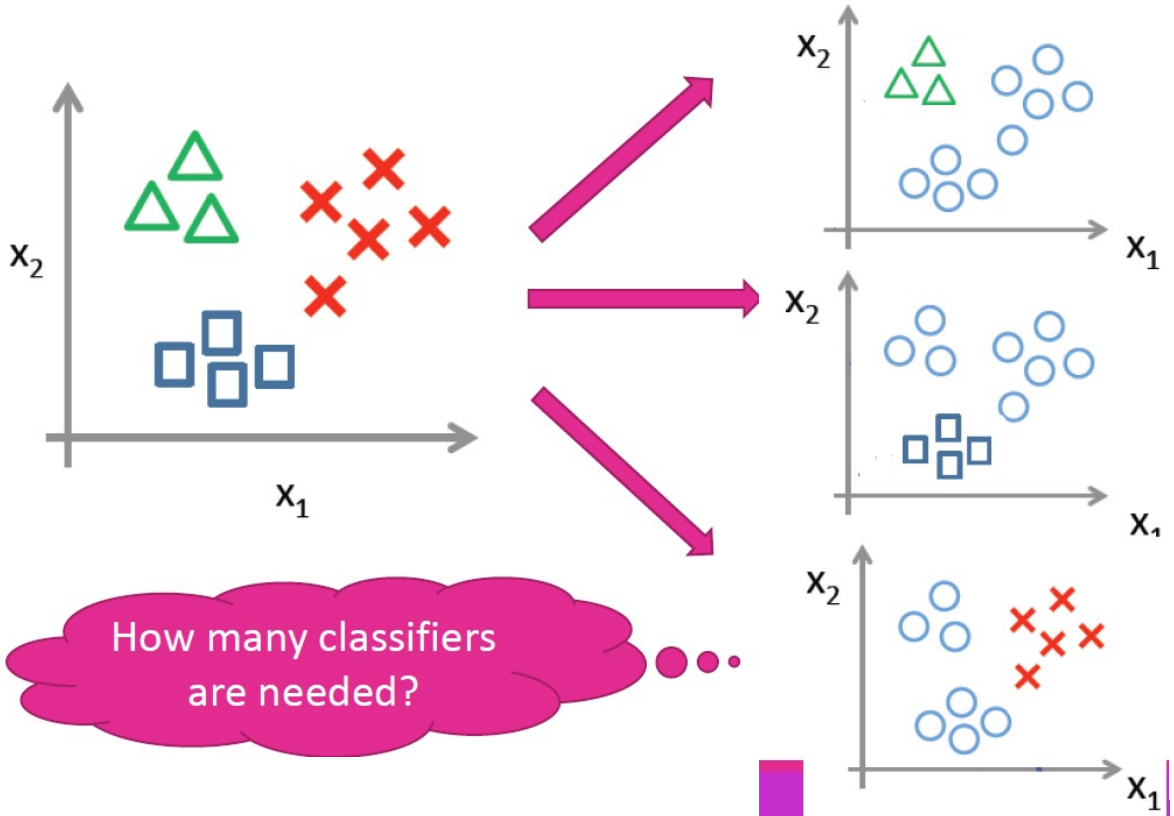






How many classifiers are needed?

One-versus-All (OVA)



For a new input x :

$$\text{Class(x)} = maximum\ vote$$

Leads to ambiguity
if the number of votes were equal

OR

$$\text{Class(x)} = \underset{i}{argmax} \sum_{j} h_{\theta}^{i,j}(x)$$

Softmax Regression

- Binary logistic regression

$$P(y=1|x;\theta)=\frac{1}{1+\exp(-\theta^T x)}$$

$$P(y=0|x;\theta)=1-P(y=1|x;\theta)=\frac{\exp(-\theta^T x)}{1+\exp(-\theta^T x)}$$

- Another alternative for multi-class is **Softmax Regression.**
  - For k class we work with softmax function instead of logistic sigmoid function

$$P(y=k|x;\theta_1,\theta_2,\dots,\theta_k)=\frac{\exp(\theta_k^T x)}{\sum_j \exp(\theta_k^T x)}$$

$$h_\theta(x^{(i)})=\begin{bmatrix}p(y^{(i)}=1|x^{(i)};\theta)\\p(y^{(i)}=2|x^{(i)};\theta)\\\vdots\\p(y^{(i)}=k|x^{(i)};\theta)\end{bmatrix}=\frac{1}{\sum_{j=1}^{k} e^{\theta_j^T x^{(i)}}}\begin{bmatrix}e^{\theta_1^T x^{(i)}}\\e^{\theta_2^T x^{(i)}}\\\vdots\\e^{\theta_k^T x^{(i)}}\end{bmatrix}$$

  - We should learn $\theta_1,\theta_2,\dots,\theta_k$ (see [2] for further details).

- [2] http://deeplearning.stanford.edu/tutorial/supervised/SoftmaxRegression/

Softmax Regression

- Binary logistic regression: Cost function

$$J(\theta) = -\frac{1}{m}\left[\sum_{i=1}^{m}(1-y^{(i)})\log(1-h_\theta(x^{(i)})) + y^{(i)}\log h_\theta(x^{(i)})\right]$$

$$= -\frac{1}{m}\left[\sum_{i=1}^{m}\sum_{j=0}^{1} 1\left\{y^{(i)}=j\right\}\log p(y^{(i)}=j|x^{(i)};\theta)\right]$$

No Closed-form solution!
It is solved using Gradient descent [2]

- Softmax Regression: Cost function

$$J(\theta) = -\frac{1}{m}\left[\sum_{i=1}^{m}\sum_{j=1}^{k} 1\left\{y^{(i)}=j\right\}\log\frac{e^{\theta_j^T x^{(i)}}}{\sum_{l=1}^{k} e^{\theta_l^T x^{(i)}}}\right]$$

$1\{y^{(i)}=j\}$ is non-zero when $y^{(i)}=j$

Linear Regression

Follow the following equation

$$p(y|\boldsymbol{x}, \boldsymbol{\theta}) = \mathcal{N}(y|w_0 + \boldsymbol{w}^\mathsf{T}\boldsymbol{x}, \sigma^2)$$

bias

Slope / weight

If input is 1-D, simple linear regression

$$f(\boldsymbol{x}; \boldsymbol{w}) = ax + b,$$

If input is N-D, multiple/multivariate linear regression

$$p(\boldsymbol{y}|\boldsymbol{x}, \mathbf{W}) = \prod_{j=1}^{J} \mathcal{N}(y_j|\boldsymbol{w}_j^\mathsf{T}\boldsymbol{x}, \sigma_j^2)$$

dimension

Least square regression

To fit a linear regression model to data, we will minimize the negative log likelihood on the training set. The objective function is given by

$$\mathrm{NLL}(\boldsymbol{w}, \sigma^2) = -\sum_{n=1}^{N} \log \left[\left(\frac{1}{2\pi\sigma^2} \right)^{\frac{1}{2}} \exp\left(-\frac{1}{2\sigma^2} (y_n - \boldsymbol{w}^\mathsf{T} \boldsymbol{x}_n)^2 \right) \right] \tag{11.4}$$

$$= \frac{1}{2\sigma^2} \sum_{n=1}^{N} (y_n - \hat{y}_n)^2 + \frac{N}{2} \log(2\pi\sigma^2) \tag{11.5}$$

weight

Error

Variance

Gaussian

The MLE is the point where $\nabla_{w,\sigma} NLL(w, \sigma^2) = 0$.
We can first optimize wrt w, and then solve for the optimal σ.

Ordinal least square – 1D

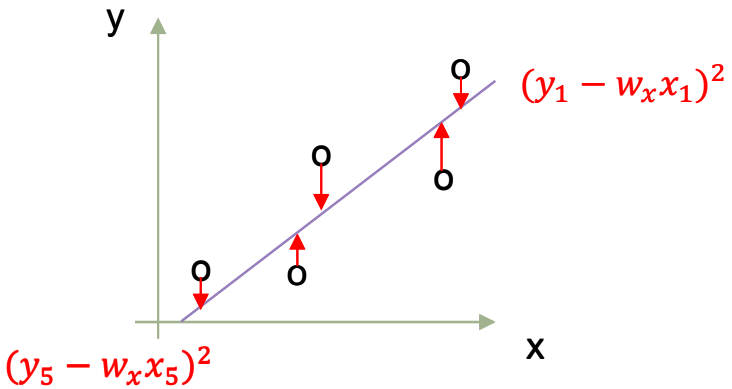
Residual sum of square is given

$$\mathrm{RSS}(\boldsymbol{w}) = \frac{1}{2}\sum_{n=1}^{N}(y_n - \boldsymbol{w}^{\mathsf{T}}\boldsymbol{x}_n)^2$$

$$(y_n - \boldsymbol{w}^{\mathsf{T}}\boldsymbol{x}_n)^2$$

$$[y_1 \quad \cdots \quad y_n] - [w_x]\,[x_1 \quad \cdots \quad x_n]$$

$$= [(y_1 - w_x x_1)^2 \quad \cdots \quad (y_n - w_x x_n)^2]$$



Ordinal least square – 2D

Residual sum of square is given

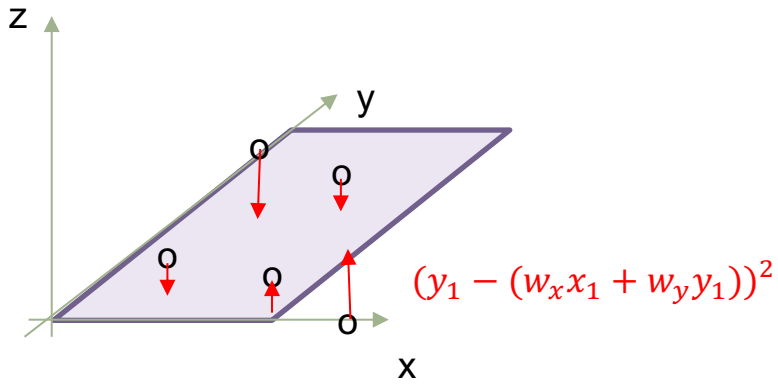
$$\mathrm{RSS}(\boldsymbol{w}) = \frac{1}{2}\sum_{n=1}^{N}(y_n - \boldsymbol{w}^{\mathsf{T}}\boldsymbol{x}_n)^2$$

$$(y_n - \boldsymbol{w}^{\mathsf{T}}\boldsymbol{x}_n)^2$$

$$[y_1 \quad \cdots \quad y_n] - \begin{bmatrix} w_x \\ w_y \end{bmatrix}^T [x_1 \quad \cdots \quad x_n]$$

$$= \left[(y_1 - (w_x x_1 + w_y y_1))^2 \quad \cdots \quad (y_n - (w_x x_n + w_y y_n))^2\right]$$

Add one dimension



How to get w?

Minimize RSS

$$\mathrm{RSS}(\boldsymbol{w}) = \frac{1}{2}\sum_{n=1}^{N}(y_n - \boldsymbol{w}^{\mathsf{T}}\boldsymbol{x}_n)^2$$

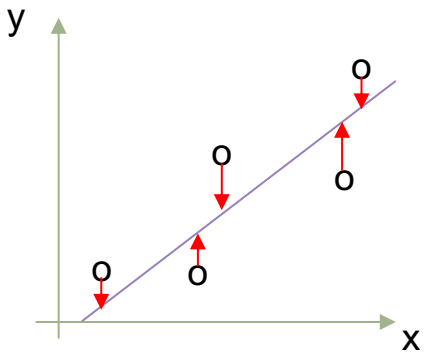
We know

$$\hat{\boldsymbol{y}} = w_1\boldsymbol{x}_{:,1} + \cdots + w_n\boldsymbol{x}_{:,D} = \mathbf{X}\boldsymbol{w}$$

To minimize the norm of the residual, $\boldsymbol{y} - \hat{\boldsymbol{y}}$, we want the residual vector to be orthogonal to every column of $\mathbf{X}$. Hence

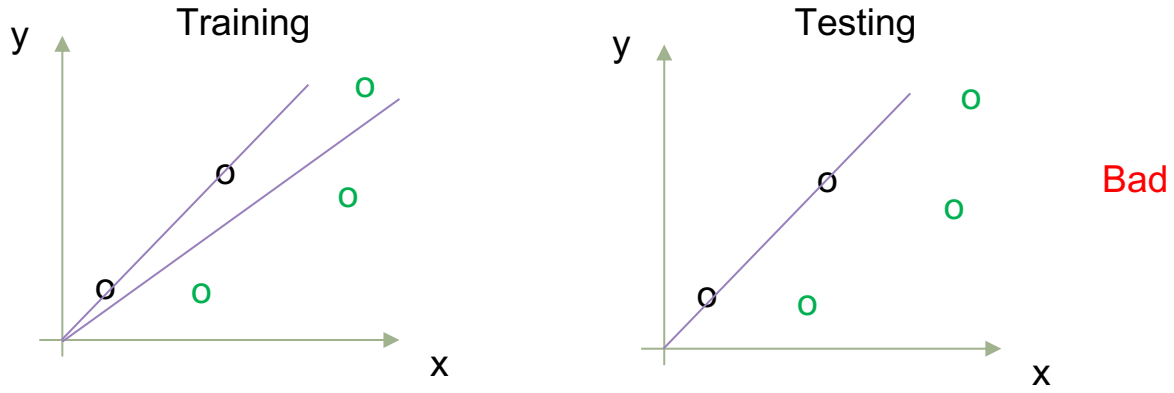
$$\boldsymbol{x}_{:,d}^{\mathsf{T}}(\boldsymbol{y} - \hat{\boldsymbol{y}}) = 0 \Rightarrow \mathbf{X}^{\mathsf{T}}(\boldsymbol{y} - \mathbf{X}\boldsymbol{w}) = \mathbf{0} \Rightarrow \boldsymbol{w} = (\mathbf{X}^{\mathsf{T}}\mathbf{X})^{-1}\mathbf{X}^{\mathsf{T}}\boldsymbol{y} \quad (11.14)$$

The ‘w’ can be obtained



Ridge regression

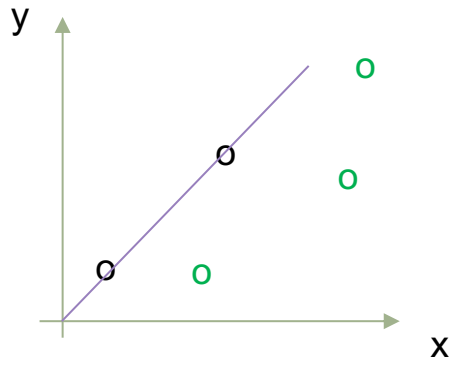
Least square estimation can results in overfitting



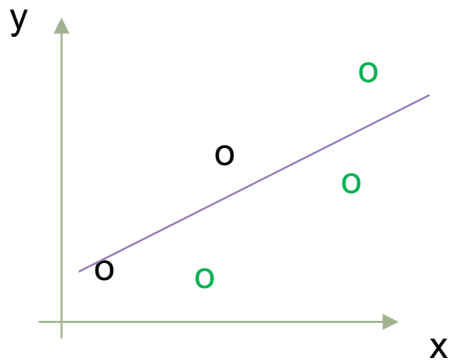
Ridge regression add a L2 regularizer to avoid overfitting (avoid very high gradient).
RSS + $\lambda * w^2$

Ridge regression

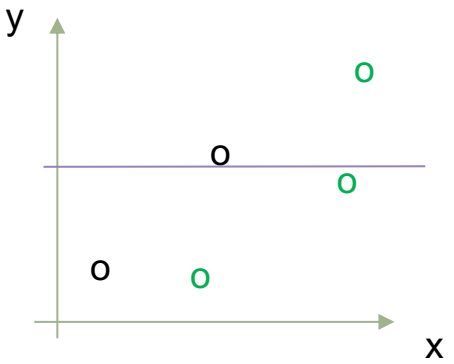
Zero λ = Least Square



Big λ



Very Big λ = 100000



$$\hat{w}_{\text{map}} = \text{argmin}\, \frac{1}{2\sigma^2}(y - \mathbf{X}w)^\mathsf{T}(y - \mathbf{X}w) + \frac{1}{2\tau^2}w^\mathsf{T}w$$
$$= \text{argmin}\, \text{RSS}(w) + \lambda||w||_2^2$$

where $\lambda \triangleq \frac{\sigma^2}{\tau^2}$ is proportional to the strength of the prior, and

$$||w||_2 \triangleq \sqrt{\sum_{d=1}^{D} |w_d|^2} = \sqrt{w^\mathsf{T}w}$$

penalizing weights that become too large in magnitude

How to choose lambda

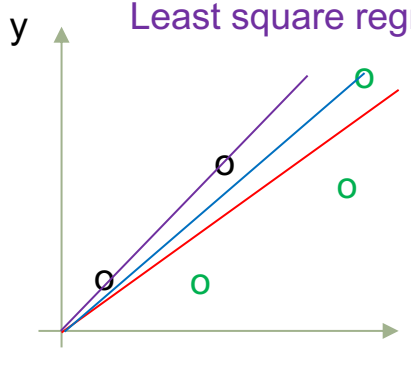
Ridge Regression add in a penalty function, L2 regularizer.

$$\begin{aligned}\hat{w}_{\text{map}} &= \text{argmin}\, \frac{1}{2\sigma^2}(y - \mathbf{X}w)^{\mathsf{T}}(y - \mathbf{X}w) + \frac{1}{2\tau^2}w^{\mathsf{T}}w \\ &= \text{argmin}\, \text{RSS}(w) + \lambda||w||_2^2\end{aligned}$$

Methods:

1. Try with strong lambda and then softer it. Check results. → regularization path
2. Cross validation.

Lasso Regression



Least square regression, *RSS* → *less bias, high variance*

Ridge regression, $RSS + \lambda * w^2$ → *high bias, low variance*

Lasso regression, $RSS + \lambda * |w|$

Absolute of w

Ridge allows parameters to be small but cannot reach zero.

Lasso allow parameters to be exactly zero.

This is useful because it can be used to perform feature selection, where the weight of certain features can be zero.

→ Can make equation simpler

Q-norm

General Equation

$$\|\boldsymbol{w}\|_q = \left(\sum_{d=1}^{D} |w_d|^q\right)^{1/q}$$

L0 loss

$$||\boldsymbol{w}||_0 = \sum_{d=1}^{D} \mathbb{I}(|w_d| > 0)$$

L1 loss

$$||\boldsymbol{w}||_1 \triangleq \sum_{d=1}^{D} |w_d|$$

L2 loss

$$||\boldsymbol{w}||_2 \triangleq \sqrt{\sum_{d=1}^{D} |w_d|^2} = \sqrt{\boldsymbol{w}^\mathsf{T}\boldsymbol{w}}$$

Elastic Net – Lasso + Ridge

$$\mathcal{L}(\boldsymbol{w}, \lambda_1, \lambda_2) = ||\boldsymbol{y} - \mathbf{X}\boldsymbol{w}||^2 + \lambda_2||\boldsymbol{w}||_2^2 + \lambda_1||\boldsymbol{w}||_1$$

Example - Cancer Data

id	lcavol	lweight	age	lbph	svi	lcp	gleason
1	-0.579818	2.76946	50	-1.38629	0	-1.38629	6
2	-0.994252	3.31963	58	-1.38629	0	-1.38629	6
3	-0.510826	2.69124	74	-1.38629	0	-1.38629	7
4	-1.20397	3.28279	58	-1.38629	0	-1.38629	6
5	0.751416	3.43237	62	-1.38629	0	-1.38629	6
6	-1.04982	3.22883	50	-1.38629	0	-1.38629	6
8	0.693147	3.53951	58	1.53687	0	-1.38629	6
11	0.254642	3.60414	65	-1.38629	0	-1.38629	6
12	-1.34707	3.59868	63	1.26695	0	-1.38629	6

Term	Least Squares	Ridge	Lasso	Elastic Net
Intercept	2.452	2.452	2.3520	2.423
lcavol	0.705	0.552	0.5710	0.611
lweight	0.292	0.278	0.2290	0.212
age	-0.142	-0.091	0.0000	0.000
lbph	0.211	0.193	0.1050	0.054
svi	0.307	0.269	0.1710	0.121
lcp	-0.276	-0.102	0.0000	0.000
gleason	-0.012	0.025	0.0000	0.000
pgg45	0.262	0.177	0.0653	0.021
Test MSE	**0.547**	**0.511**	**0.4820**	**0.450**

Least Square – Worst
Ridge – weight is smaller but won't reach zero, better than LS
Lasso – some features' weight are zero; features eliminated
Elastic Net - Best

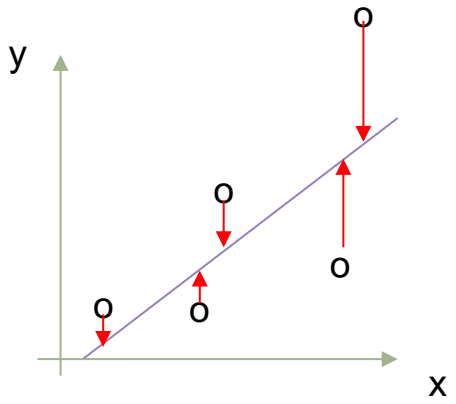
Generalized Linear Model

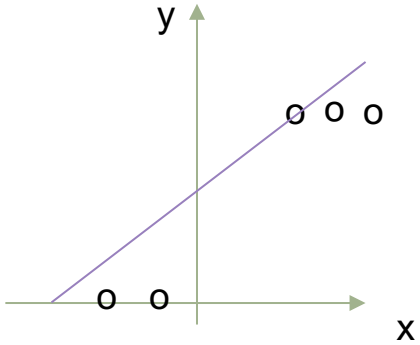
If we have the following, ordinal least square is not suitable.

- Exponential graph

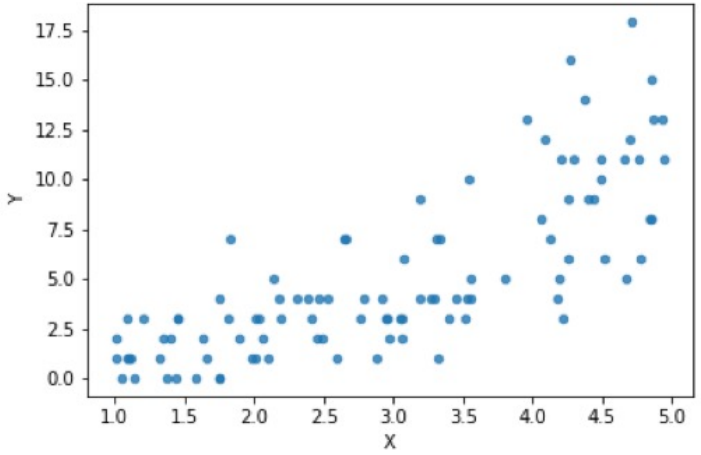
- Variance of errors in y is not constant, and varies with X.

- Response variable is not continuous, but categorical.





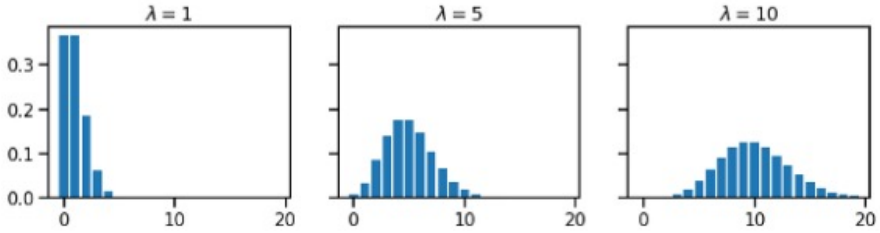
Generalized Linear Model



We can't use linear regression.

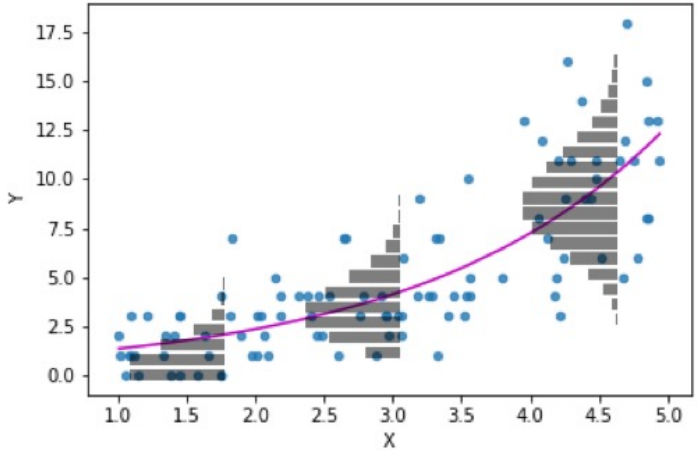
Variance increase with x

A suitable regression is Poisson Regression, one type of GLM model.



Poisson distribution with mean=1, 5, 10

GLM – Poisson Regression



GLM normally made up of three components:

1. Linear predictor – b0+b1x
2. Link function – log link function
3. Probability distribution – Poisson distribution

```
# Poisson regression code
import statsmodels.api as sm
exog, endog = sm.add_constant(x), y
mod = sm.GLM(endog, exog,
             family=sm.families.Poisson(link=sm.families.links.log))
res = mod.fit()
```

GLM – Linear/Logistic Regression

Linear Regression

1. Linear predictor – b0+b1x
2. Link function – identify link function
3. Probability distribution – Normal distribution

$$\mu_i = b_0 + b_1 x_i$$
$$y_i \sim \mathcal{N}(\mu_i, \varepsilon)$$

Linear regression

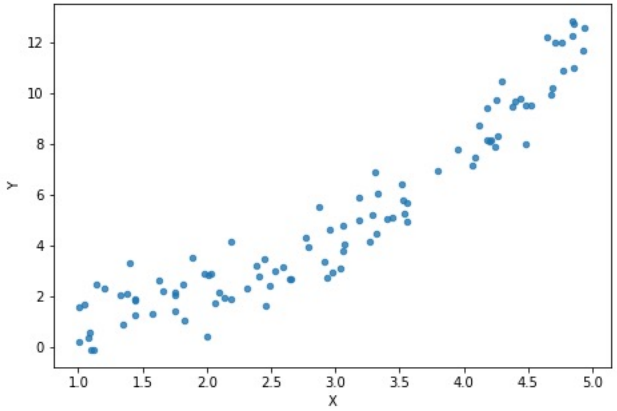
Logistic Regression

1. Linear predictor – b0+b1x
2. Link function – logic link function
3. Probability distribution – Binomial / Bernoulli distribution

$$z_i = b_0 + b_1 x_i$$
$$q_i = \frac{1}{1 + \exp(-z_i)}$$
$$y_i \sim \mathrm{Bern}(q_i)$$

logistic regression

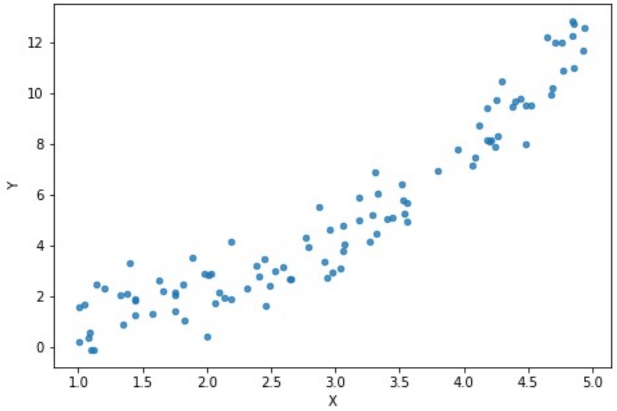
Custom GLM



Relationship between x and y is not linear.

Link function = Log link function

Custom GLM



Variance seems constants

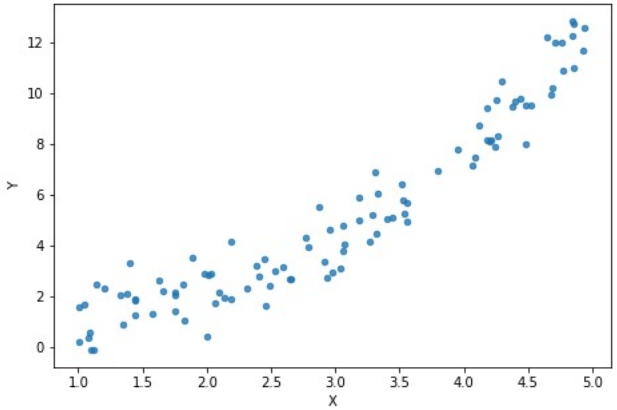
Which probability distribution for variance to choose?

1. Normal
2. Poisson

Which probability distribution for variance to choose?

https://towardsdatascience.com/generalized-linear-models-9cbf848bb8ab

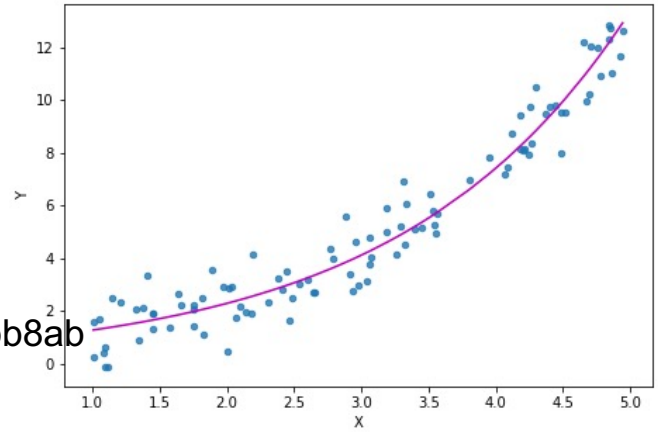
Custom GLM



Variance seems constants

Which probability distribution for variance to choose?

1. Normal
2. Poisson



https://towardsdatascience.com/generalized-linear-models-9cbf848bb8ab

Decision Theory | Information Theory | Optimization