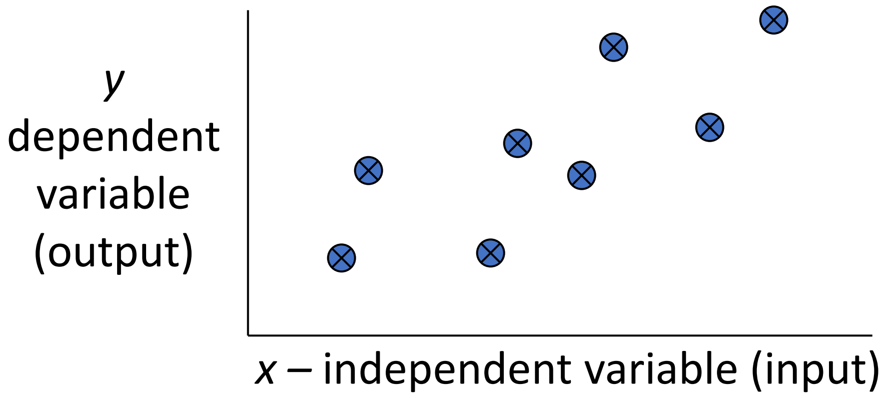
Machine Learning

Lecture: 2-3

Regression

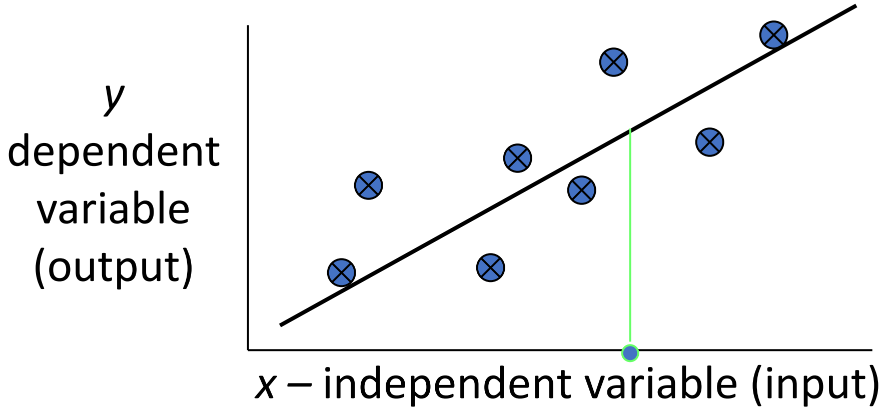
Regression

- For classification the output(s) is nominal
- In regression the output is continuous
  - Function Approximation
- Many models could be used – Simplest is linear regression
  - Fit data with the best hyper-plane which "goes through" the points



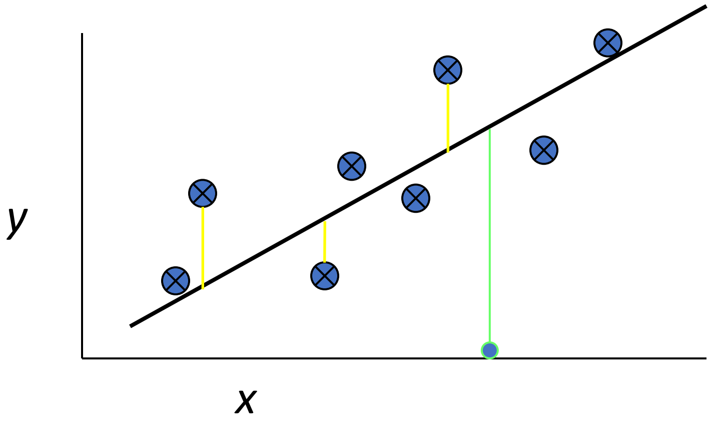
Regression

- For classification the output(s) is nominal
- In regression the output is continuous
  - Function Approximation
- Many models could be used – Simplest is linear regression
  - Fit data with the best hyper-plane which "goes through" the points



Regression

- For classification the output(s) is nominal
- In regression the output is continuous
  - Function Approximation
- Many models could be used – Simplest is linear regression
  - Fit data with the best hyper-plane which "goes through" the points
  - For each point the difference between the predicted point and the actual observation is the *residue*



Simple Linear Regression

- For now, assume just one (input) independent variable *x*, and one (output) dependent variable *y*
  - Multiple linear regression assumes an input vector **x**
  - Multivariate linear regression assumes an output vector **y**
- We "fit" the points with a line (i.e. hyperplane)
- Which line should we use?
  - Choose an objective function
  - For simple linear regression we use sum squared error (SSE)
    - Σ ($\textit{predicted}_i - \textit{actual}_i)^2 = \Sigma$ ($\textit{residue}_i)^2$
  - Thus, find the line which minimizes the sum of the squared residues (e.g. least squares)
  - This exactly mimics the case assuming data points were sampled from an actual target hyperplane with Gaussian noise added

How do we "learn" parameters

- For the 2-*d* problem (line) there are coefficients for the bias and the independent variable (*y*-intercept and slope)

$$Y = \beta_0 + \beta_1 X$$

- To find the values for the coefficients (weights) which minimize the objective function we can take the partial derivates of the objective function (SSE) with respect to the coefficients. Set these to 0, and solve.

$$\beta_1 = \frac{n \square\, xy - \square\, x \square\, y}{n \square\, x^2 - \left(\square\, x\right)^2} \qquad \beta_0 = \frac{\square\, y - \beta_1 \square\, x}{n}$$

Multiple Linear Regression

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_n X_n$$

- There is a closed form for finding multiple linear regression weights which requires matrix inversion, etc.
- There are also iterative techniques to find weights
- One is the delta rule. For regression we use an output node which is not thresholded (just does a linear sum) and iteratively apply the delta rule – *For regression net is the output*

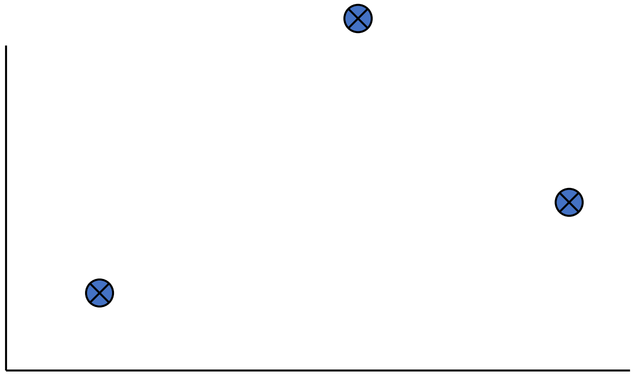
$$\Delta w_i = c(t - net)x_i$$

- Where c is the learning rate and x_i is the input for that weight
- Delta rule will update until minimizing the SSE, thus solving multiple linear regression
- There are other regression approaches that give different results by trying to better handle outliers and other statistical anomalies

SSE and Linear Regression

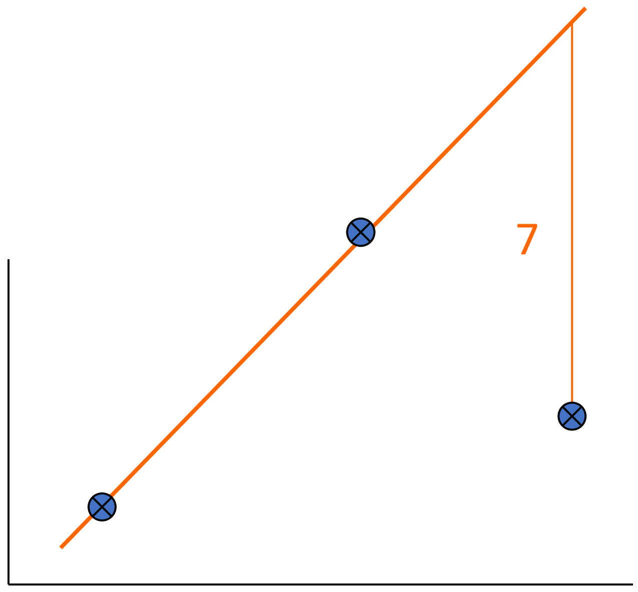
- SSE squares the difference of the predicted vs actual
- Don't want residues to cancel each other
- Could use absolute or other distances to solve problem

  $\Sigma\ |predicted_i - actual_i|$: L1 vs L2
- SSE leads to a parabolic error surface which is great for gradient descent
- Which line would least squares choose?
  - There is *always one* “best” fit with SSE (L2)
  - An L1 error can have multiple best fits



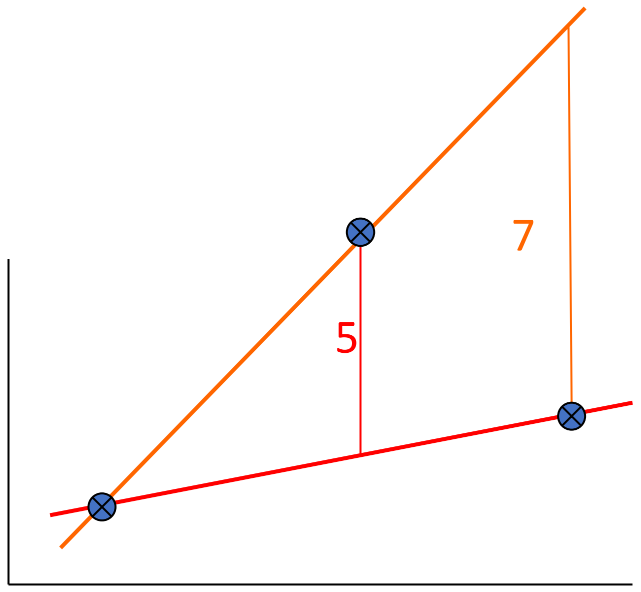
SSE and Linear Regression

- SSE leads to a parabolic error surface which is great for gradient descent
- Which line would least squares choose?
  - There is always one "best" fit



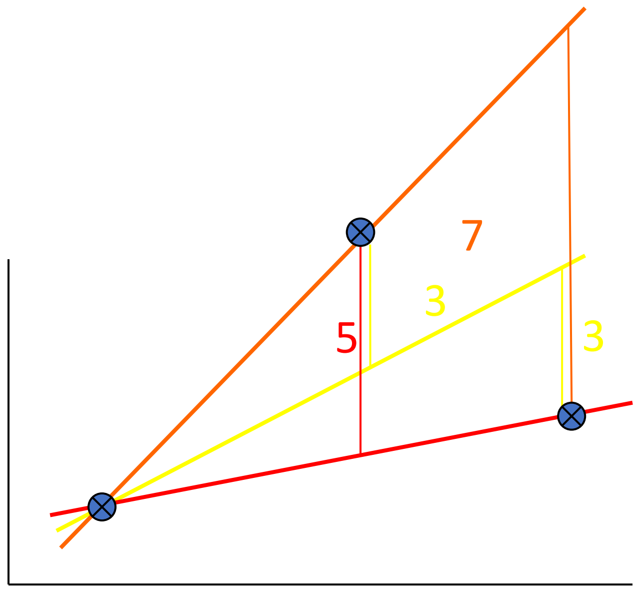
SSE and Linear Regression

- SSE leads to a parabolic error surface which is great for gradient descent
- Which line would least squares choose?
  - There is always one "best" fit



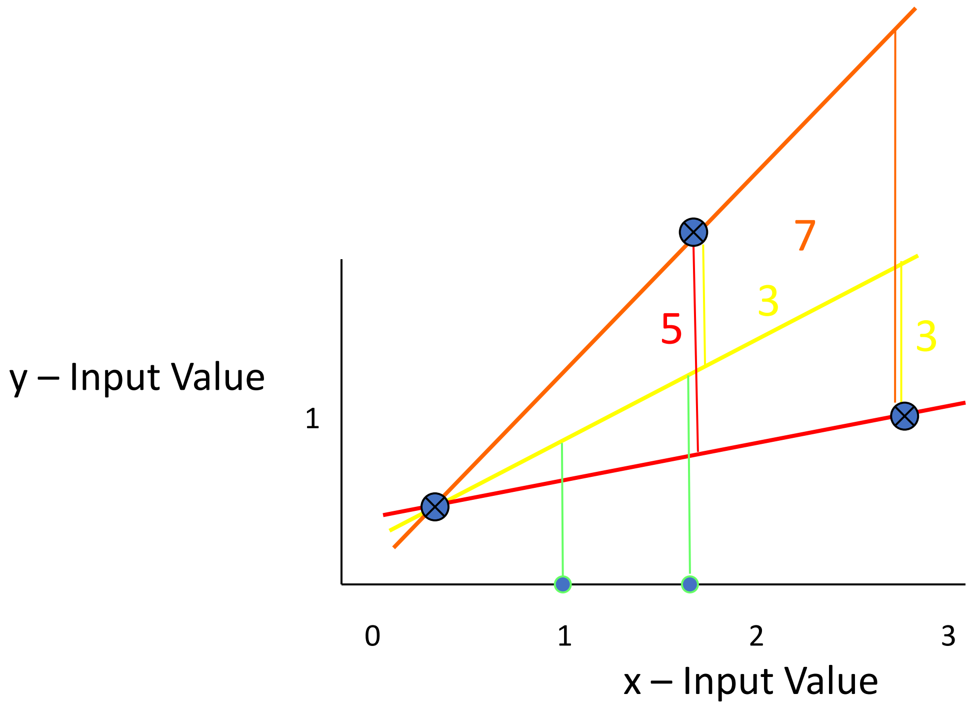
SSE and Linear Regression

- SSE leads to a parabolic error surface which is great for gradient descent
- Which line would least squares choose?
  - There is always one "best" fit
- Note that the squared error causes the model to be more highly influenced by outliers
  - But *is* the best fit assuming Gaussian noise error from true target



SSE and Linear Regression Generalization

- In generalization all *x* values map to a *y* value on the chosen regression line



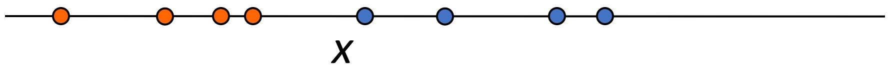
Intelligibility (Interpretable ML, Transparent)

- One advantage of linear regression models (and linear classification) is the potential to look at the weights to give insight into which input variables are most important in predicting the output
- The variables with the largest weight magnitudes have the highest correlation with the output
  - A large positive weight implies that the output will increase when this input is increased (positively correlated)
  - A large negative weight implies that the output will decrease when this input is increased (negatively correlated)
  - A small or 0 weight suggests that the input is uncorrelated with the output (at least at the 1st order)
- Linear regression/classification can be used to find best "indicators"
  - Be careful not to confuse correlation with causality
  - Linear models cannot detect higher order correlations! The power of more complex machine learning models!!

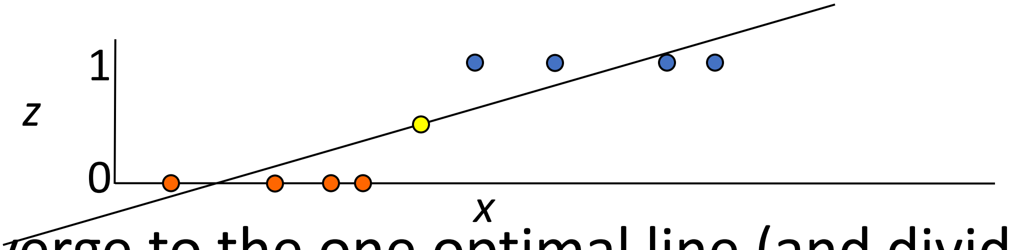
Delta rule natural for regression, not classification

$$\boxed{\Delta w_i = c(t - net)x_i}$$

- Consider the one-dimensional case
- The decision surface for the perceptron would be any point that divides the instances

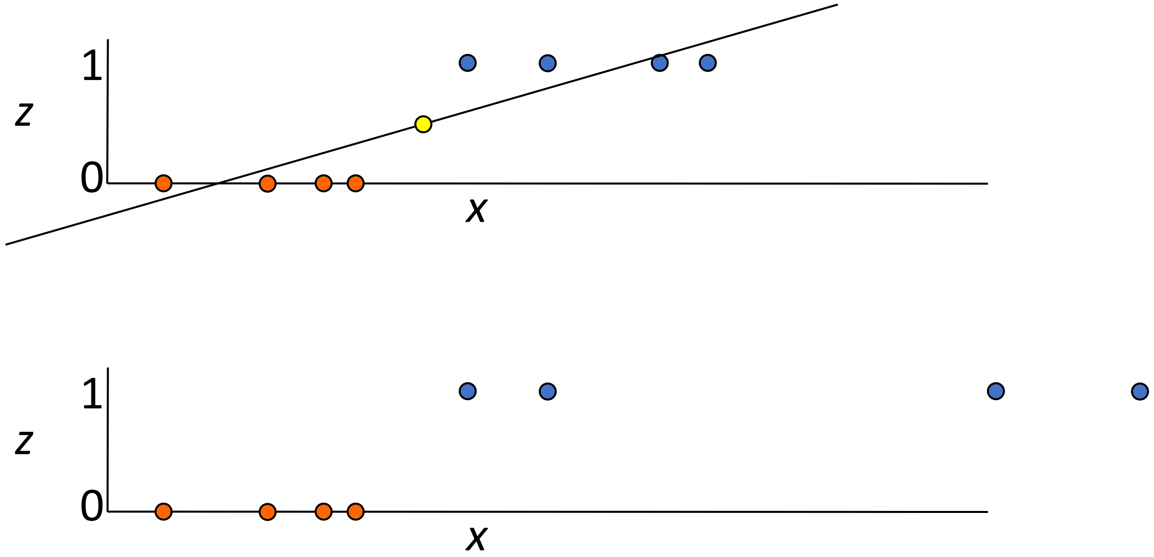


- Delta rule will try to fit a line through the target values which minimizes SSE and the decision point is where the line crosses .5 for 0/1 targets. Looking down on data for perceptron view. Now flip it on its side for delta rule view.



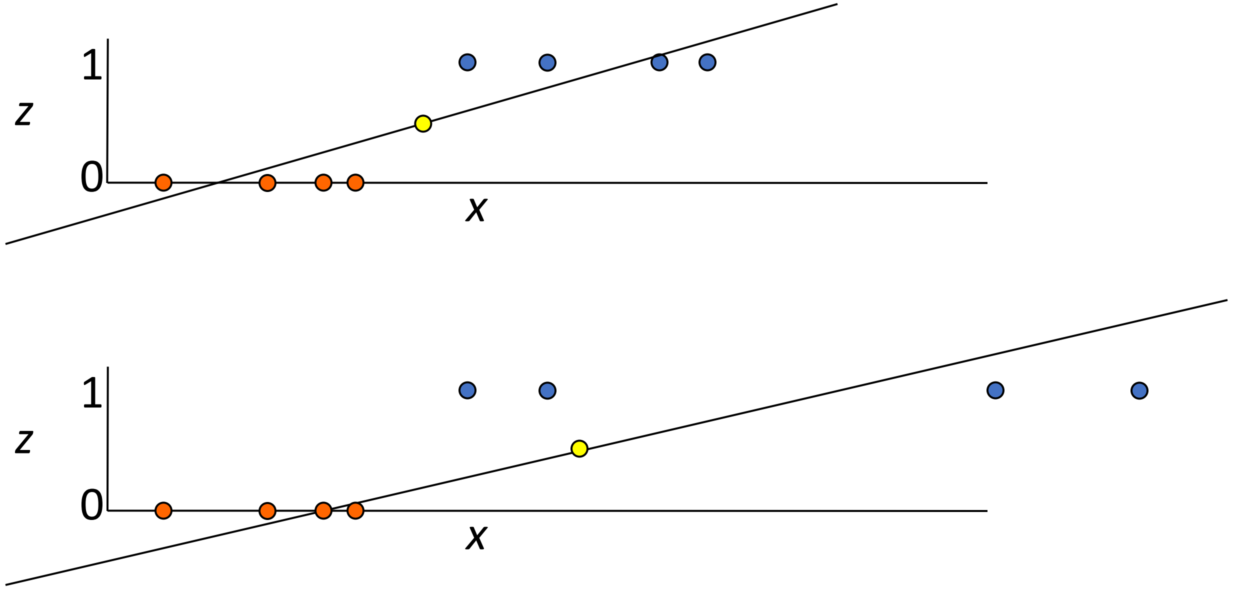
- Will converge to the one optimal line (and dividing point) for this objective

Delta Rule for Classification?



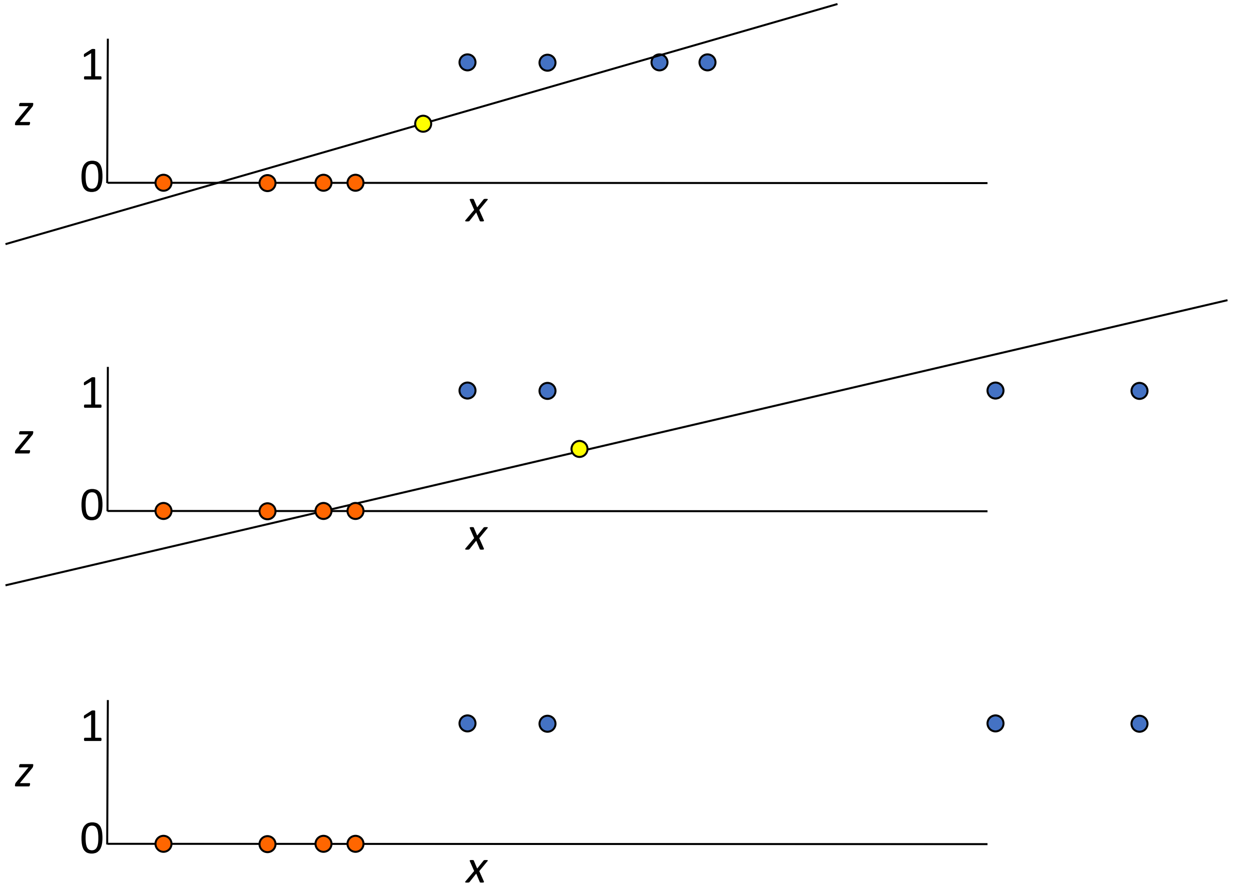
- What would happen in this adjusted case for perceptron and delta rule and where would the decision point (i.e. .5 crossing) be?

Delta Rule for Classification?



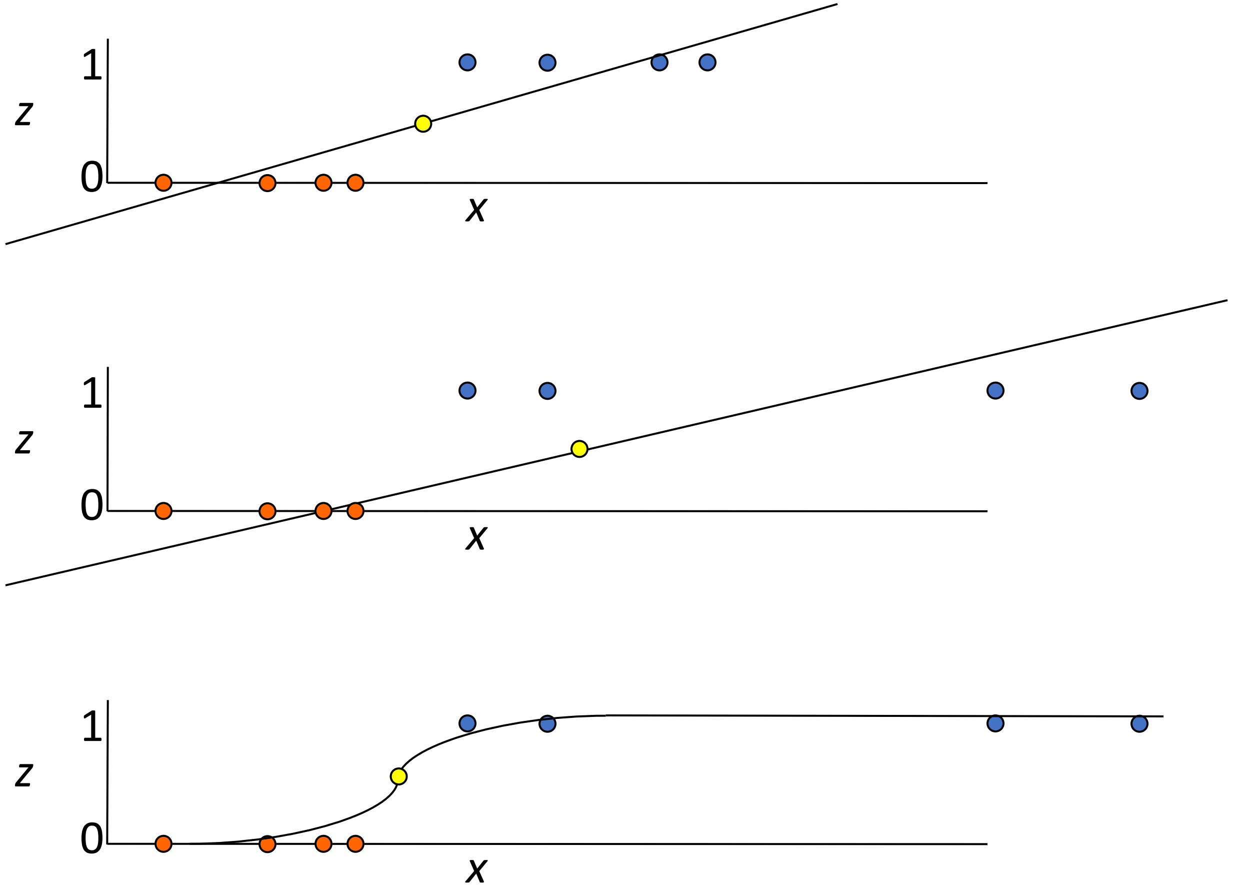
- Leads to misclassifications even though the data is linearly separable
- For Delta rule the objective function is to minimize the regression line SSE, not maximize classification

Delta Rule for Classification?



- What would happen if we were doing a regression fit with a sigmoid/logistic curve rather than a line?

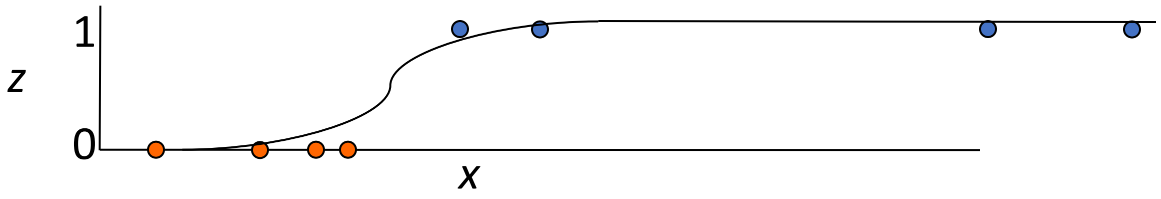
Delta Rule for Classification?



- Sigmoid fits many binary decision cases quite well with a probability. This is what logistic regression does.

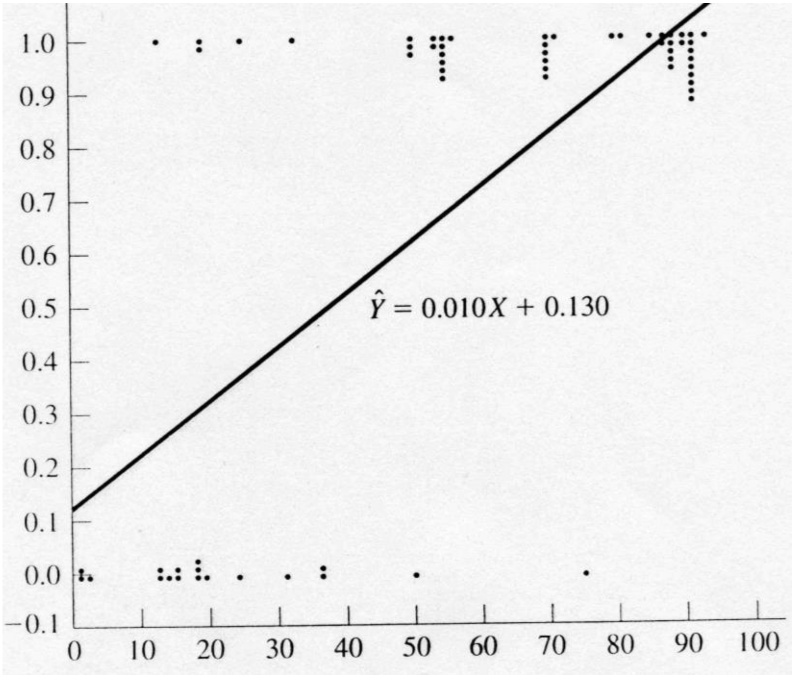
Logistic Regression

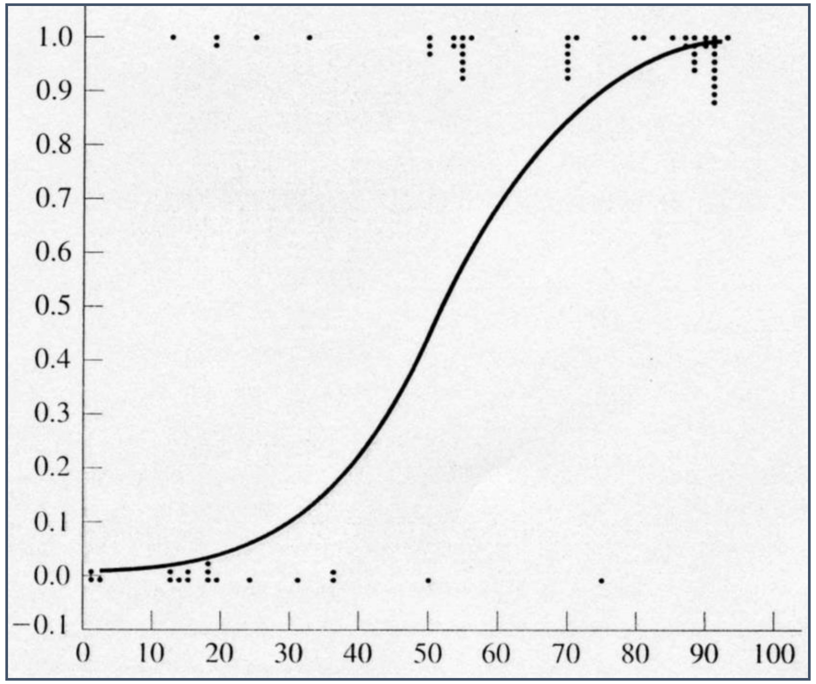
- One commonly used algorithm is Logistic Regression
- Assumes that the dependent (output) variable is binary which is often the case in medical and other studies. (Does person have disease or not, survive or not, accepted or not, etc.)
- Like Quadric, Logistic Regression does a particular non-linear transform on the data after which it just does linear regression on the transformed data
- Logistic regression fits the data with a sigmoidal/logistic curve rather than a line and outputs an approximation of the probability of the output given the input



Logistic Regression Example

- Age (X axis, input variable) – Data is fictional
- Heart Failure (Y axis, 1 or 0, output variable)
- If use value of regression line as a probability approximation
  - Extrapolates outside 0-1 and not as good empirically
- Sigmoidal curve to the right gives empirically good probability approximation and is bounded between 0 and 1





Logistic Regression Approach

Learning

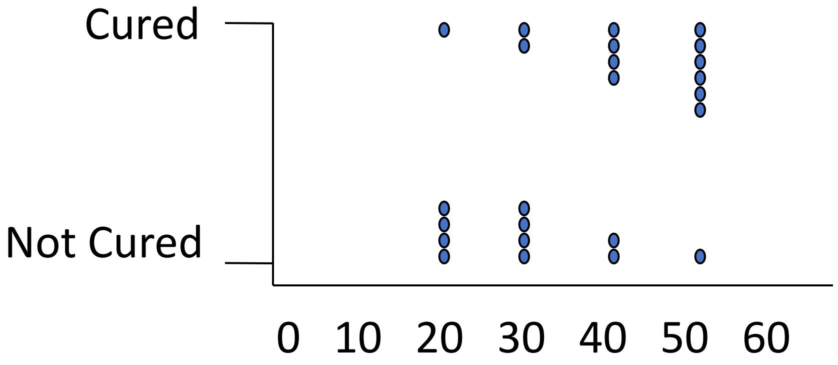
1. Transform initial input probabilities into log odds (logit)
2. Do a standard linear regression using the logit values
   - This effectively fits a logistic curve to the data, while still just doing a linear regression with the transformed input (ala quadric machine, etc.)

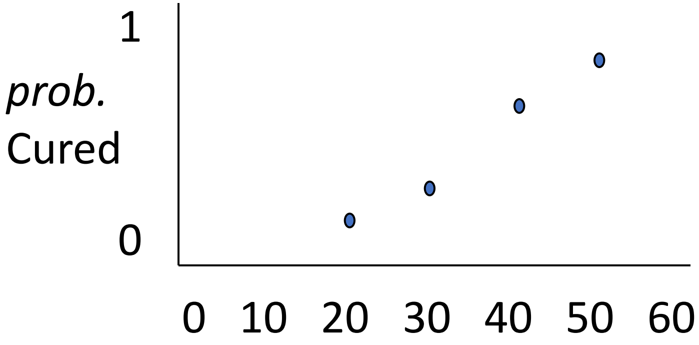
Generalization

1. Find the value for the new input on the logit line
2. Transform that logit value back into a probability

Non-Linear Pre-Process to Logit (Log Odds)

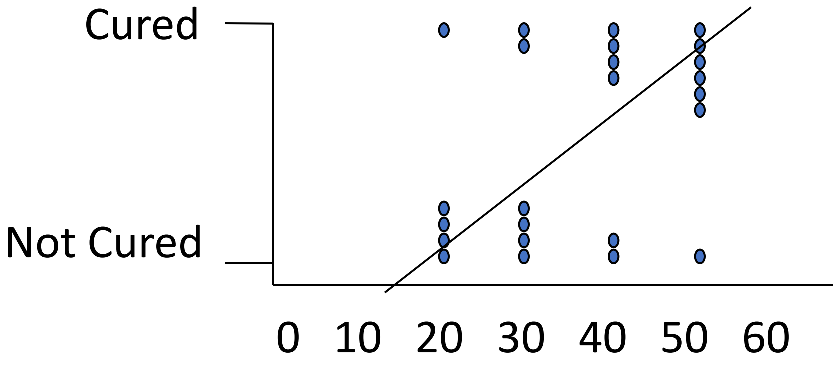
Medication Dosage	# Cured	Total Patients	Probability: # Cured/Total Patients
20	1	5	.20
30	2	6	.33
40	4	6	.67
50	6	7	.86

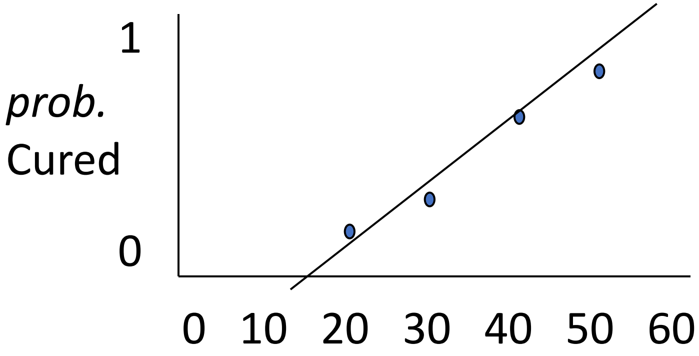




Non-Linear Pre-Process to Logit (Log Odds)

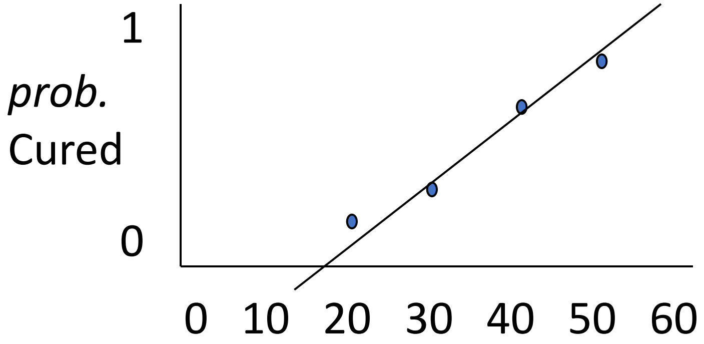
Medication Dosage	# Cured	Total Patients	Probability: # Cured/Total Patients
20	1	5	.20
30	2	6	.33
40	4	6	.67
50	6	7	.86

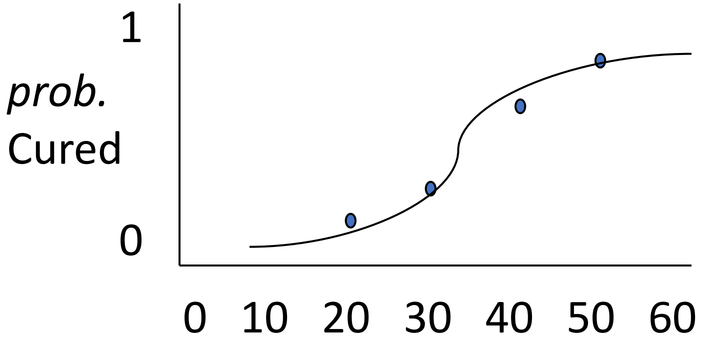




Logistic Regression Approach

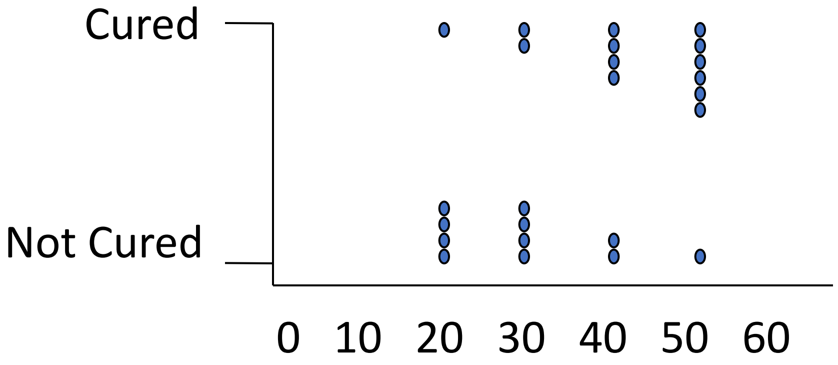
- Could use linear regression with the probability points, but that would not extrapolate well
- Logistic version is better but how do we get it?
- Similar to Quadric we do a non-linear pre-process of the input and then do linear regression on the transformed values – do a linear regression on the log odds - Logit

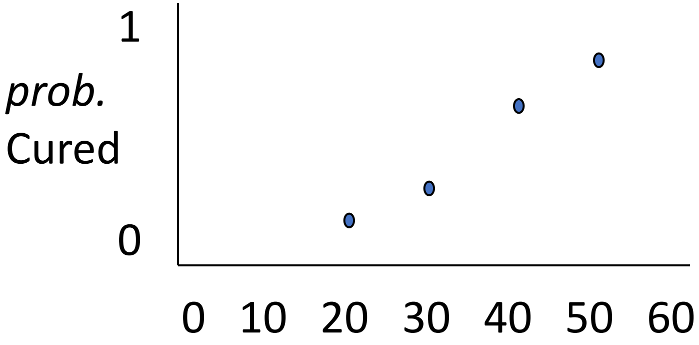




Non-Linear Pre-Process to Logit

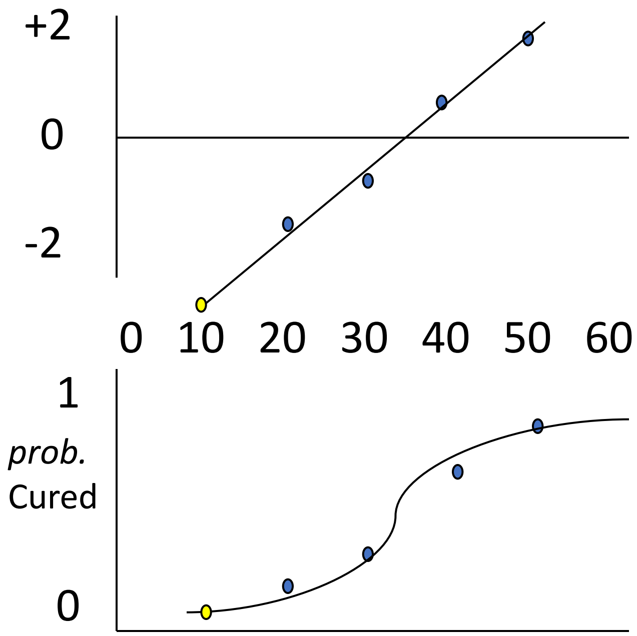
Medication Dosage	# Cured	Total Patients	Probability: # Cured/Total Patients	Odds: *p*/(1-*p*) = # cured/ # not cured	Logit Log Odds: ln(Odds)
20	1	5	.20	.25	-1.39
30	2	6	.33	.50	-0.69
40	4	6	.67	2.0	0.69
50	6	7	.86	6.0	1.79





Regression of Log Odds

Medication Dosage	# Cured	Total Patients	Probability: # Cured/Total Patients	Odds: *p*/(1-*p*) = # cured/ # not cured	Log Odds: ln(Odds)
20	1	5	.20	.25	-1.39
30	2	6	.33	.50	-0.69
40	4	6	.67	2.0	0.69
50	6	7	.86	6.0	1.79



- $y = .11x - 3.8$ - Logit regression equation
- Now we have a regression line for log odds (logit)
- To generalize, we use the log odds value for the new data point
- Then we transform that log odds point to a probability: $p = e^{\text{logit}(x)}/(1+e^{\text{logit}(x)})$
- For example assume we want p for dosage = 10

$$\text{Logit}(10) = .11(10) - 3.8 = -2.7$$

$p(10) = e^{-2.7}/(1+e^{-2.7}) = .06$ [note that we just work backwards from logit to p]

- These p values make up the sigmoidal regression curve (which we never have to actually plot)

Non-Linear Regression

- Note that linear regression is to regression what the perceptron is to classification
  - Simple, useful models which will often underfit
- The more powerful classification models which we will be discussing going forward in class can usually also be used for non-linear regression
  - MLP with Backpropagation, Decision Trees, Nearest Neighbor, etc.
- They can learn functions with arbitrarily complex high dimensional shapes

Summary

- Linear Regression and Logistic Regression are nice tools for many simple situations
    - But both force us to fit the data with one shape (line or sigmoid) which will often underfit
- Intelligible results
- When problem includes more arbitrary non-linearity then we need more powerful models which we will introduce
    - Yet non-linear data transformations (e.g. Quadric perceptron) can help in these cases while still using a linear model for learning
- These models are commonly used in data mining applications and also as a "first attempt" at understanding data trends, indicators, etc.

Home work

When to use linear regression algorithm ?

When to use non linear regression?

Differentiate linear and non linear regression.