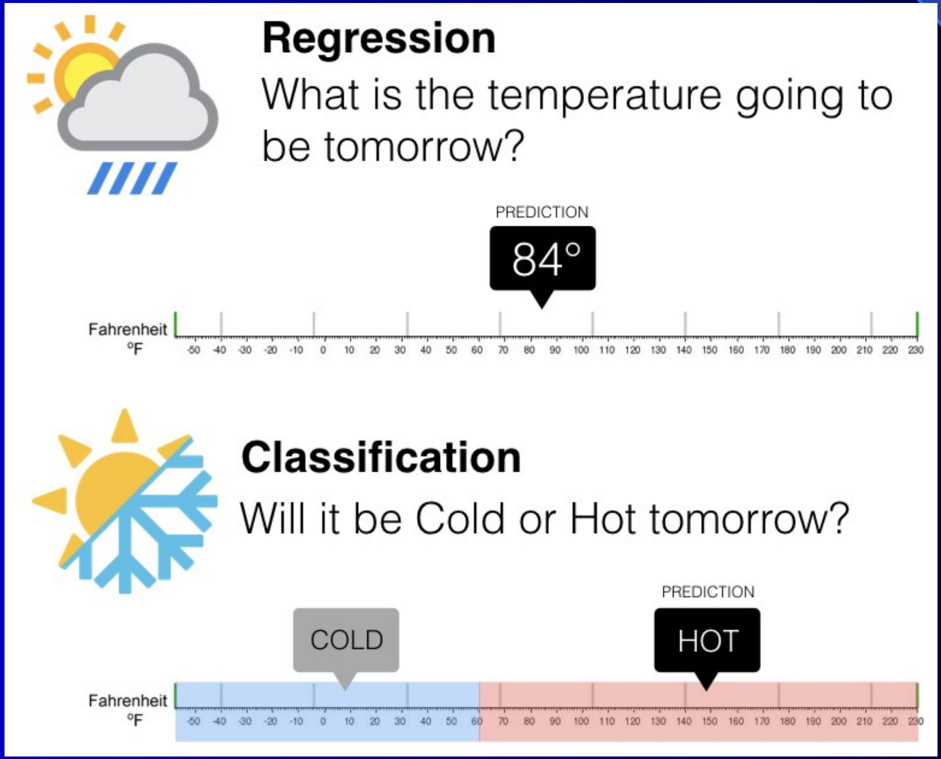
Classification & Regression

Classification & Regression

- Classification predicts a discrete class for a given input
  - And/or a probability of belonging in that class

- Regression predicts a continuous value based on the input



Regression

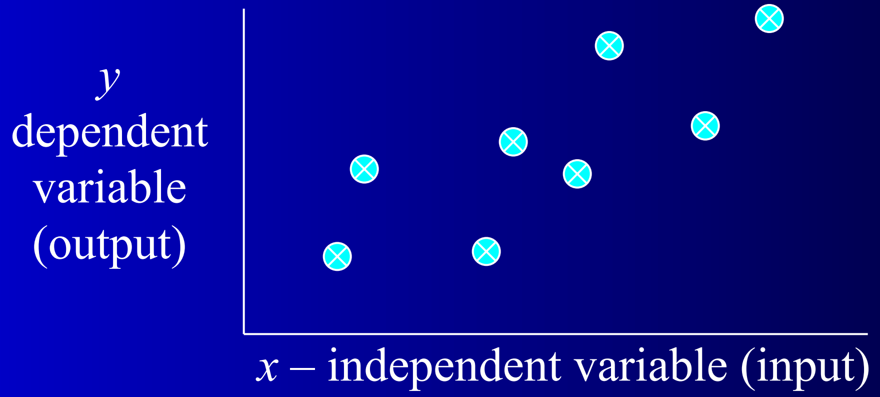
What is the temperature going to be tomorrow?

Classification

Will it be Cold or Hot tomorrow?

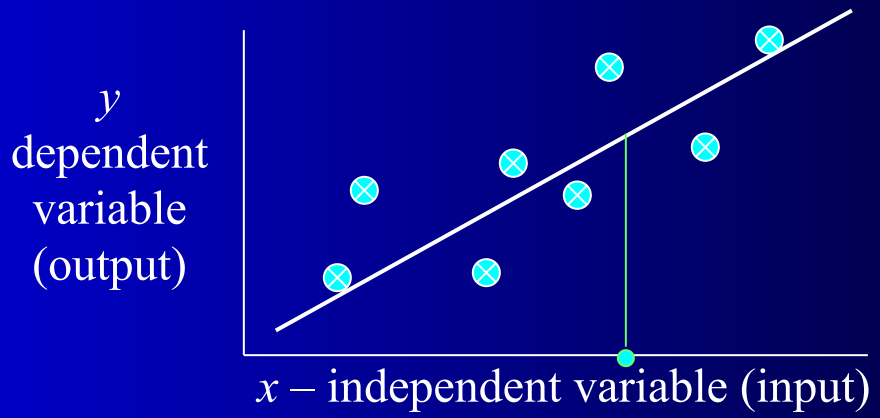
Classification vs. Regression

- 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



Classification vs. 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

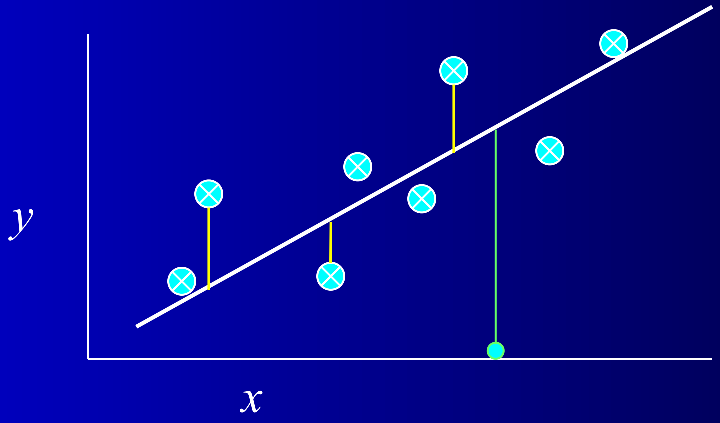


Classification vs. Regression

- 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 *error*



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)
    - $\Sigma\,(\mathit{predicted}_i - \mathit{actual}_i)^2 = \Sigma\,(\mathit{residue}_i)^2$
  - 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)

- 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.

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*

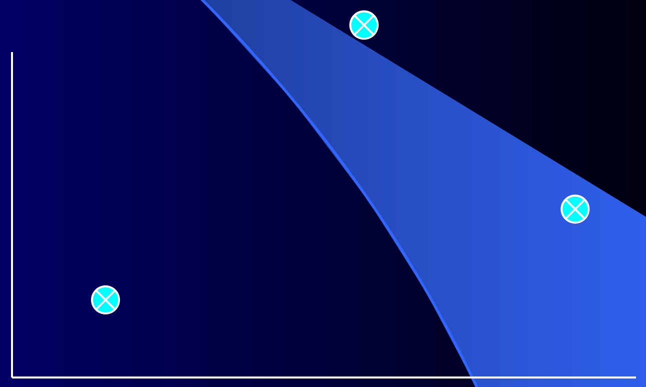
- Where *c* is the learning rate and x_i is the input for that weight

- 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

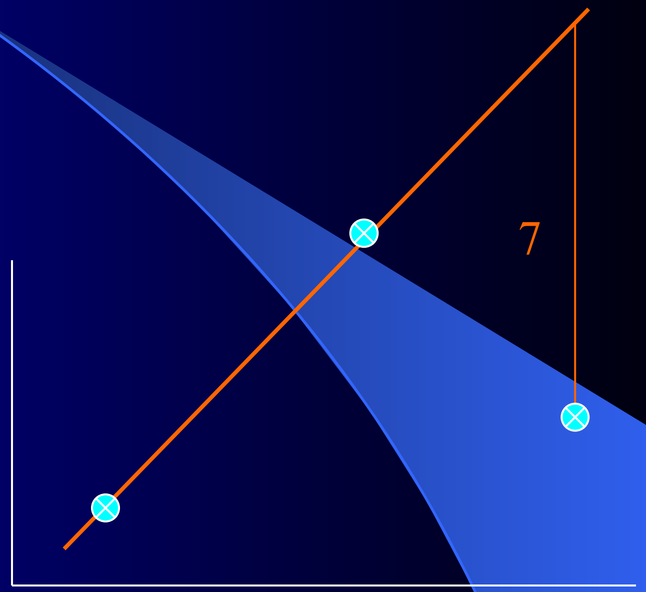
- SSE squares the difference of the predicted vs actual
- Don't want residues to cancel each other
- SSE leads to a parabolic error surface which is great for gradient descent
- Which line would least squares choose?
  - There is *always one* “best” fit with SSE (L2)
  - An L1 error can have multiple best fits



SSE and Linear Regression

- SSE leads to a parabolic error surface which is great for gradient descent

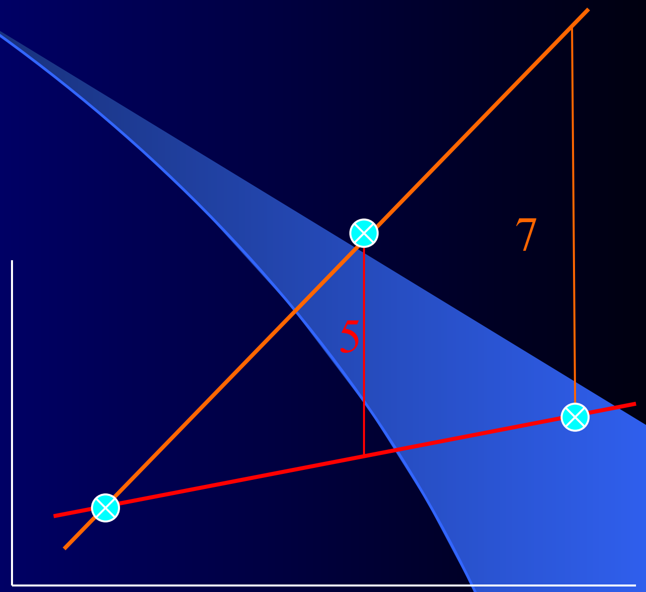
- Which line would least squares choose?
  - There is always one “best” fit



SSE and Linear Regression

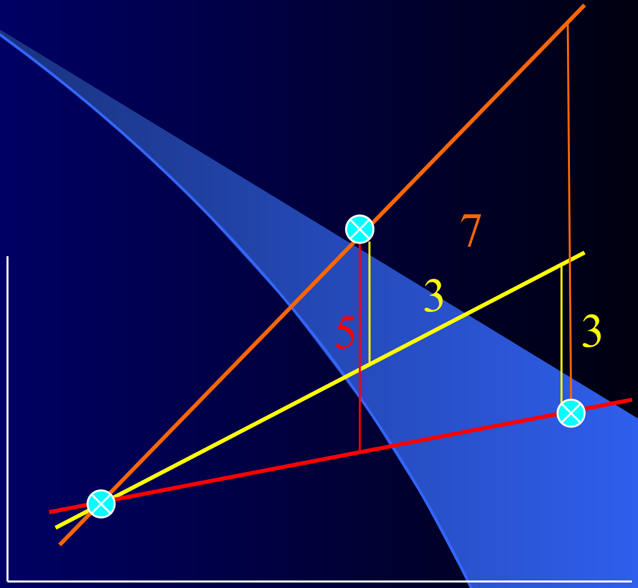
- SSE leads to a parabolic error surface which is great for gradient descent

- Which line would least squares choose?
  - There is always one "best" fit



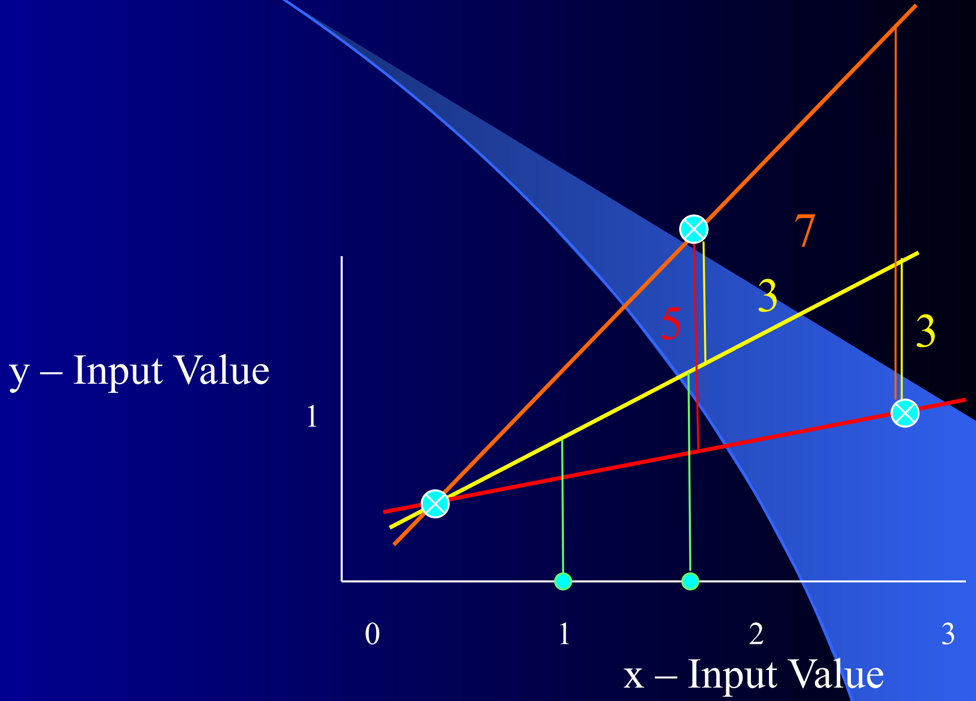
SSE and Linear Regression

- 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



Linear Regression - Example

- Assume we start with all weights as 1 (no bias)
- What are the new weights after one iteration through the training set using the delta rule with a learning rate $c = 1$
- How does it generalize for the novel input (1.5, 1)?

x_1	x_2	*Target*	*Net*	w_1	w_2
				1	1
0.7	-0.6	2			
1	0.25	-0.8			

Linear Regression - Challenge Question

- Assume we start with all weights as 1 (don't use bias weight though you usually always will – else forces the line through the origin)

- Remember for regression we use an output node which is not thresholded (just does a linear sum) and iteratively apply the delta rule – *thus the net is the output*

- What are the new weights after one iteration through the following training set using the delta rule with a learning rate *c* = 1

- How does it generalize for the novel input (-.3, 0)?

x_1	x_2	*Target y*
.5	-.2	1
1	0	-.4

- After one epoch the weight vector is:
  - A. 1 .5
  - B. 1.35 .94
  - C. 1.35 .86
  - D. -.4 .86
  - E. None of the above

Linear Regression - Challenge Question

- Assume we start with all weights as 1
- What are the new weights after one iteration through the training set using the delta rule with a learning rate *c* = 1
- How does it generalize for the novel input (-.3, 0)?
  - -.3*-.4 + 0*.86 = .12

Train on 1st instance

$$\Delta w_1 = 1 \times (1 - .3) \times 0.5 = .35$$

$$\Delta w_2 = 1 \times (1 - .3) \times (-.2) = -.14$$

$$w_1 = 1.35 \quad w_2 = .86$$

Train on 2nd instance

$$\Delta w_1 = 1 \times \left(-.4 - 1.35\right) \times 1 = -.1.75$$

$$\Delta w_2 = 1 \times \left(-.4 - 1.35\right) \times 0 = 0$$

$$w_1 = -.4 \quad w_2 = .86$$

x_1	x_2	*Target*	*Net*	w_1	w_2
				1	1
.5	-.2	1	.3	1.35	.86
1	0	-.4	1.35	-.4	.86

Linear Regression Homework

- Assume we start with all weights as 0 (**Include the bias!**)
- What are the new weights after one iteration through the following training set using the delta rule with a learning rate $c = .2$
- How does it generalize for the novel input (1, .5)?

x_1	x_2	*Target*
.3	.8	.7
-.3	1.6	-.1
.9	0	1.3

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!!