Logistic Regression

Research Skills: Machine Learning

Grzegorz Chrupała
g.chrupala @ uvt.nl

Models and learning algorithms

- We saw how to decouple model from learning algorithm
- (Stochastic) Gradient Descent can train various models
- Today, a classification model
  - can be fit using (S)GD

Error function

- Learning a model – minimizing error function
- Different models – different error functions
- For example, SSE for linear regression

$$\mathrm{SSE} = \sum_{i=1}^{N} (y^i_{\mathrm{pred}} - y^i)^2$$

Loss function

- Loss function quantifies our mistake on a **single example**
- **Squared loss** corresponds to **SSE**.

$$\ell_{\text{squared}}(z) = (z - y)^2$$

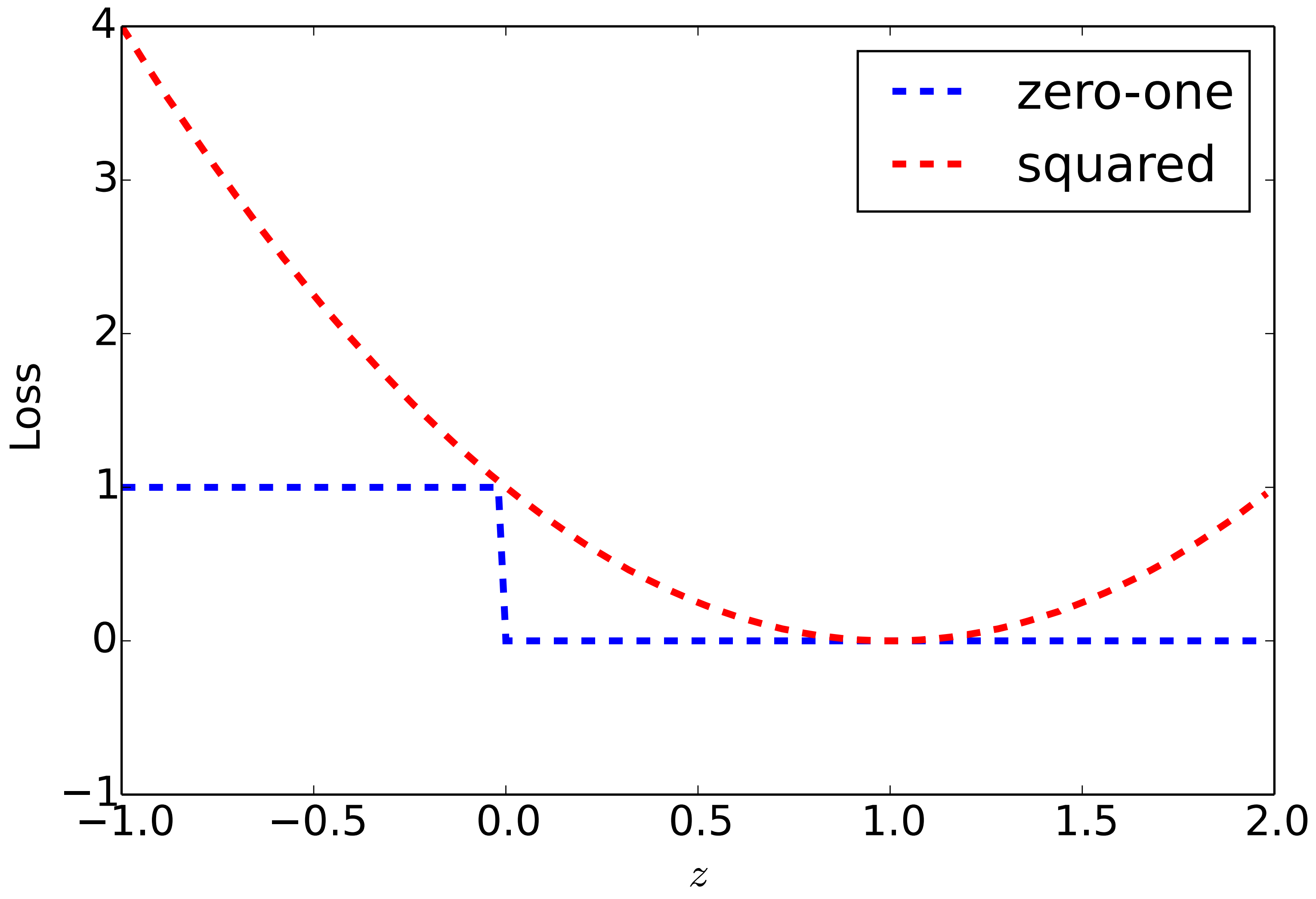
where $z = \mathbf{w} \cdot \mathbf{x} + b$ is the score of the model and y the target

Loss for classification

- Zero-one loss:

  1 if we're made a mistake, 0 otherwise

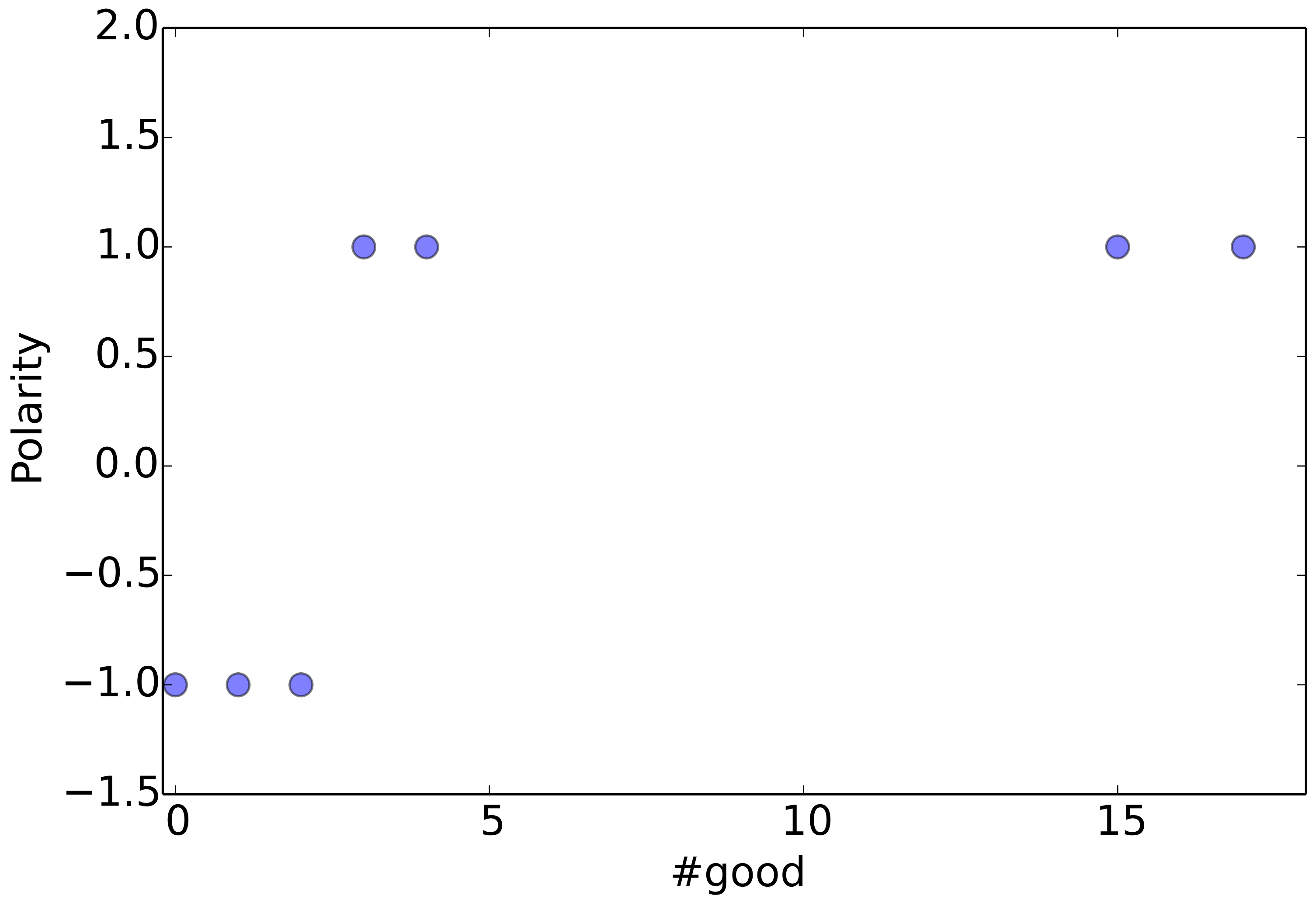
$$\ell_{0/1}(z) = \begin{cases} 1 \text{ if } t = 1 \text{ and } z < 0 \\ 1 \text{ if } t = -1 \text{ and } z >= 0 \\ 0 \text{ otherwise} \end{cases}$$


zero-one
squared
Loss
z
4
3
2
1
0
−1
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0

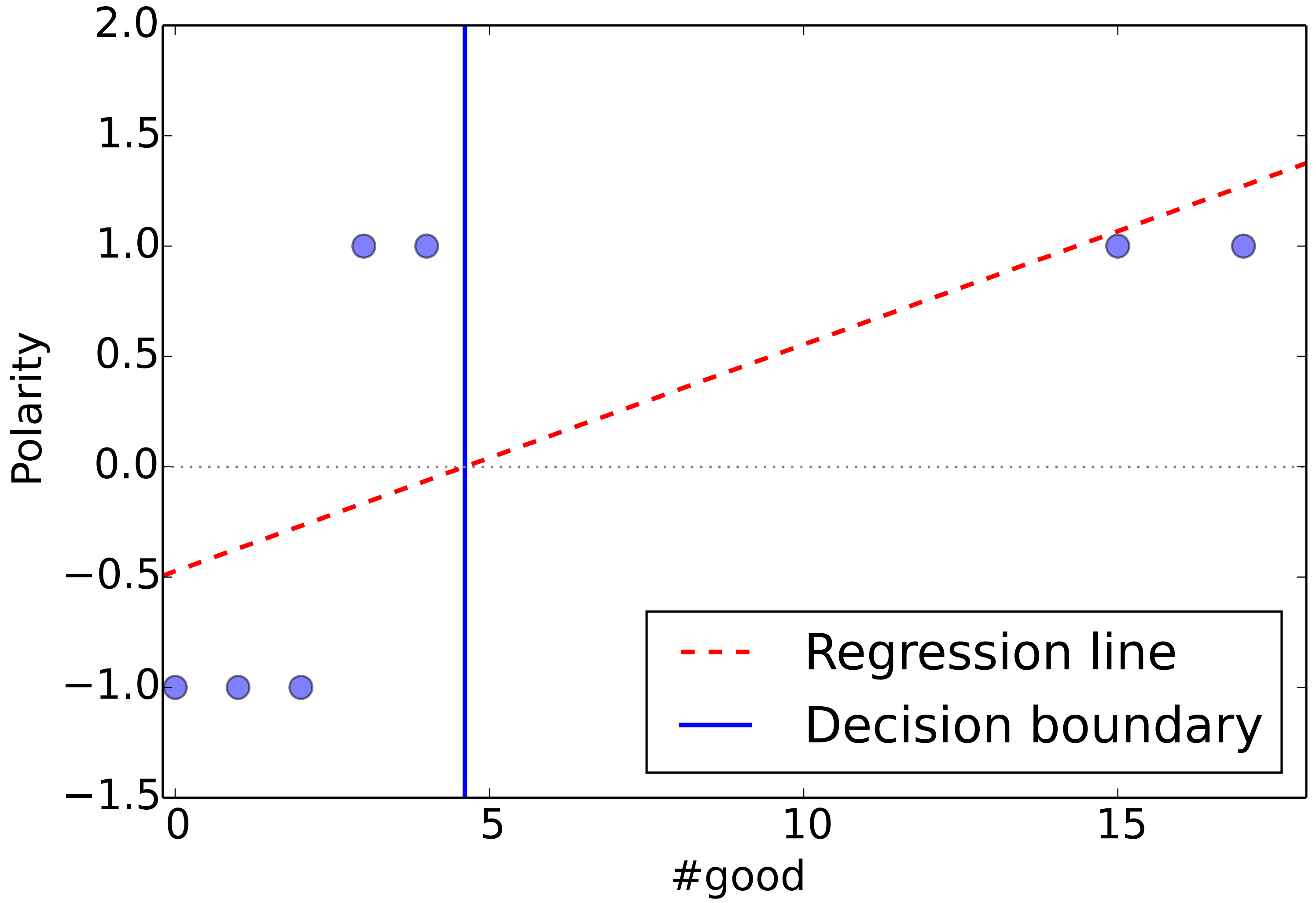
Loss for classification

- Zero-one loss – no gradient
- Could we just use squared loss for classification?
  - What happens if $z = 2.0$?
  - and $z = 10.0$?
  - Penalizes confident correct predictions

Example



Example



2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
Polarity
0
5
10
15
#good
Regression line
Decision boundary

Problem

- Bad decision boundary
- Model cares too much about predicting exactly 1 for examples with high *#good*
- Need better loss function

Let's find a better way.

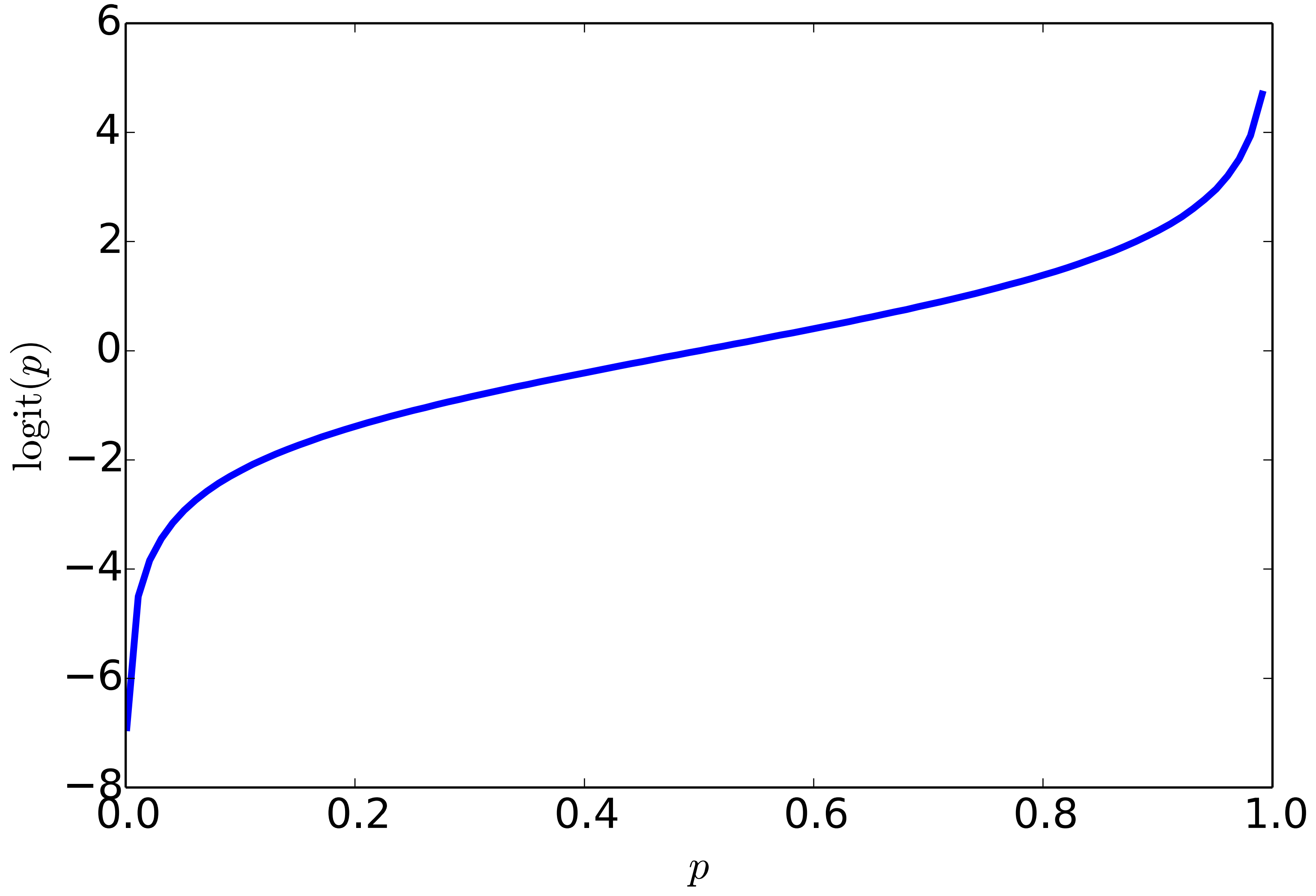
Regression for classifying

- In regression we predict numbers
- In classification we predict labels
- Logistic regression
  – regress on **probabilities** of labels

Logistic regression

- Let *p* = probability that label is positive
  - number between 0 and 1
- Logit function maps *p* to [-∞,∞]

$$\operatorname{logit}(p) = \log\left(\frac{p}{1-p}\right)$$



Examples

logit(0.01) = -4.6
logit(0.50) = 0.0
logit(0.99) = 4.6

Logistic regression

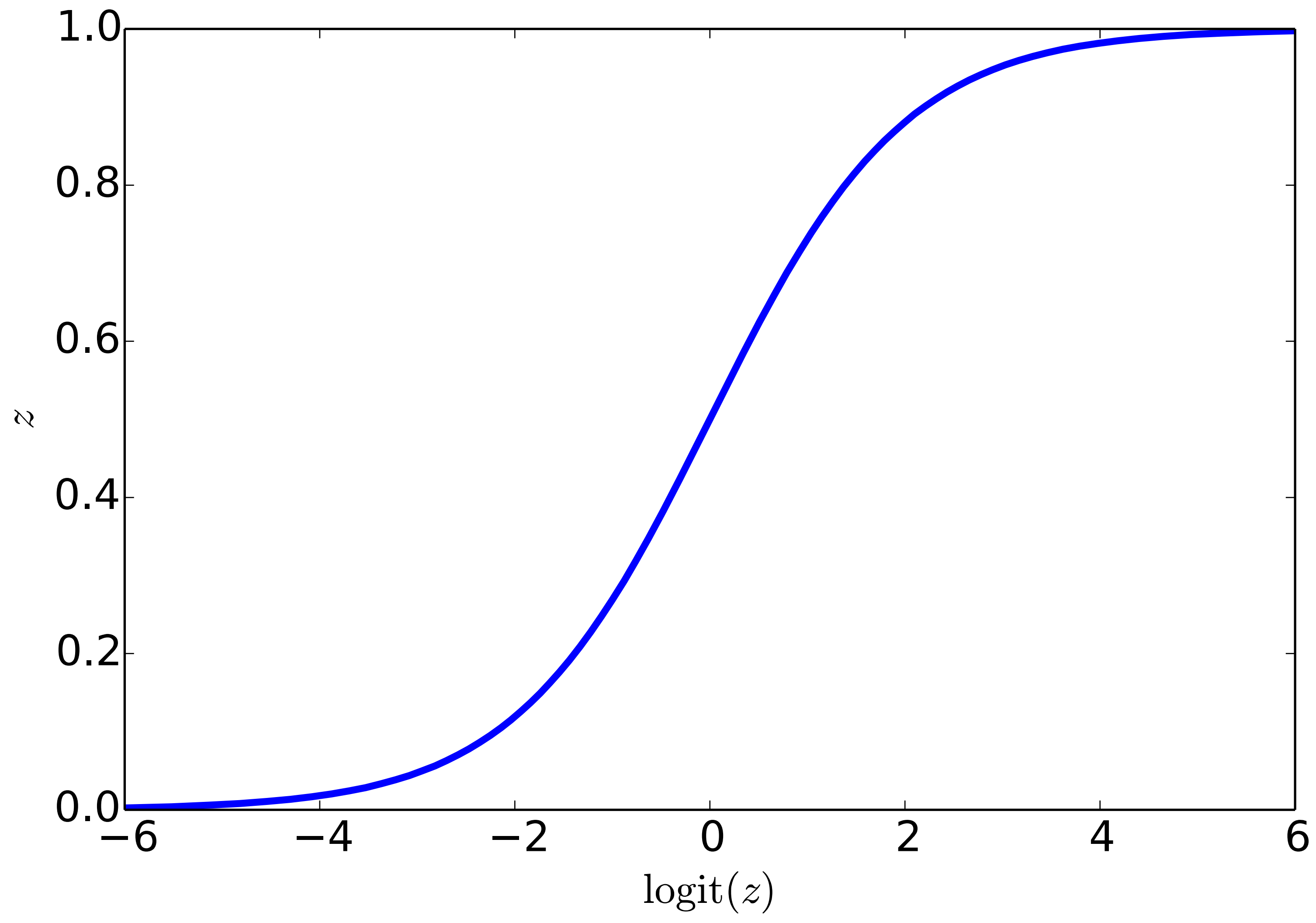
- A logistic regression model predicts logit(p) using a linear model

$$\operatorname{logit}(p)_{\text{pred}} = \mathbf{w} \cdot \mathbf{x} + b$$

Logistic regression

- We can map the logit back to probability using **inverse logit** (or **logistic**, or **sigmoid**) function

$$\operatorname{logit}^{-1}(z) = \frac{1}{1 + \exp(-z)}$$



Examples

$\mathrm{logit}^{-1}(0) = 0.50$
$\mathrm{logit}^{-1}(-4.6) = 0.01$
$\mathrm{logit}^{-1}(4.6) = 0.99$

Logistic regression

- Putting the pieces together

$$p_{\text{pred}} = \text{logit}^{-1}(\mathbf{w} \cdot \mathbf{x} + b)$$

Example: movie reviews

$$\begin{array}{lcccccc}
 & & \#good & \#dark & \#mediocre & \#the & \\
\mathbf{x}^1 & = (& 1, & 0, & 0, & 5 &) \\
\mathbf{x}^2 & = (& 2, & 3, & 2, & 7 &) \\
\mathbf{w} & = (& 2.5, & 0.5, & -4.0, & 0.0 &)
\end{array}$$

- $b = 0.5$
- $score^1 = 3.0$, $p^1 = logit^{-1}(3.0) = 0.95$
- $score^2 = -1.0$, $p^2 = logit^{-1}(-1.0) = 0.27$

Log loss function aka cross-entropy

- Loss function quantifying mistakes for LR

$$\ell_{\log}(z) = \begin{cases} -\log(p_{\text{pred}}) & \text{if } y = 1 \\ -\log(1 - p_{\text{pred}}) & \text{if } y = 0 \end{cases}$$

$$\text{where } p_{\text{pred}} = \text{logit}^{-1}(z)$$

- Minimize log loss – find model which gives **maximum probability** to training targets

Log loss function

$$\ell_{\log}(z) = \begin{cases} -\log(p_{\text{pred}}) & \text{if } y = 1 \\ -\log(1 - p_{\text{pred}}) & \text{if } y = 0 \end{cases}$$

$$\text{where } p_{\text{pred}} = \operatorname{logit}^{-1}(z)$$

alternative notation

$$\ell_{\log}(z) = -y\log(p_{\text{pred}}) - (1-y)\log(1-p_{\text{pred}})$$

Summary

Logistic vs Linear Regression

Logistic vs Linear: prediction

- Both use the score of the linear model

$$z = \mathbf{w} \cdot \mathbf{x} + b$$

- Linear regression uses it directly

$$y_{\text{pred}} = z$$

- Logistic regression via inverse logit

$$p_{\text{pred}} = \text{logit}^{-1}(z)$$

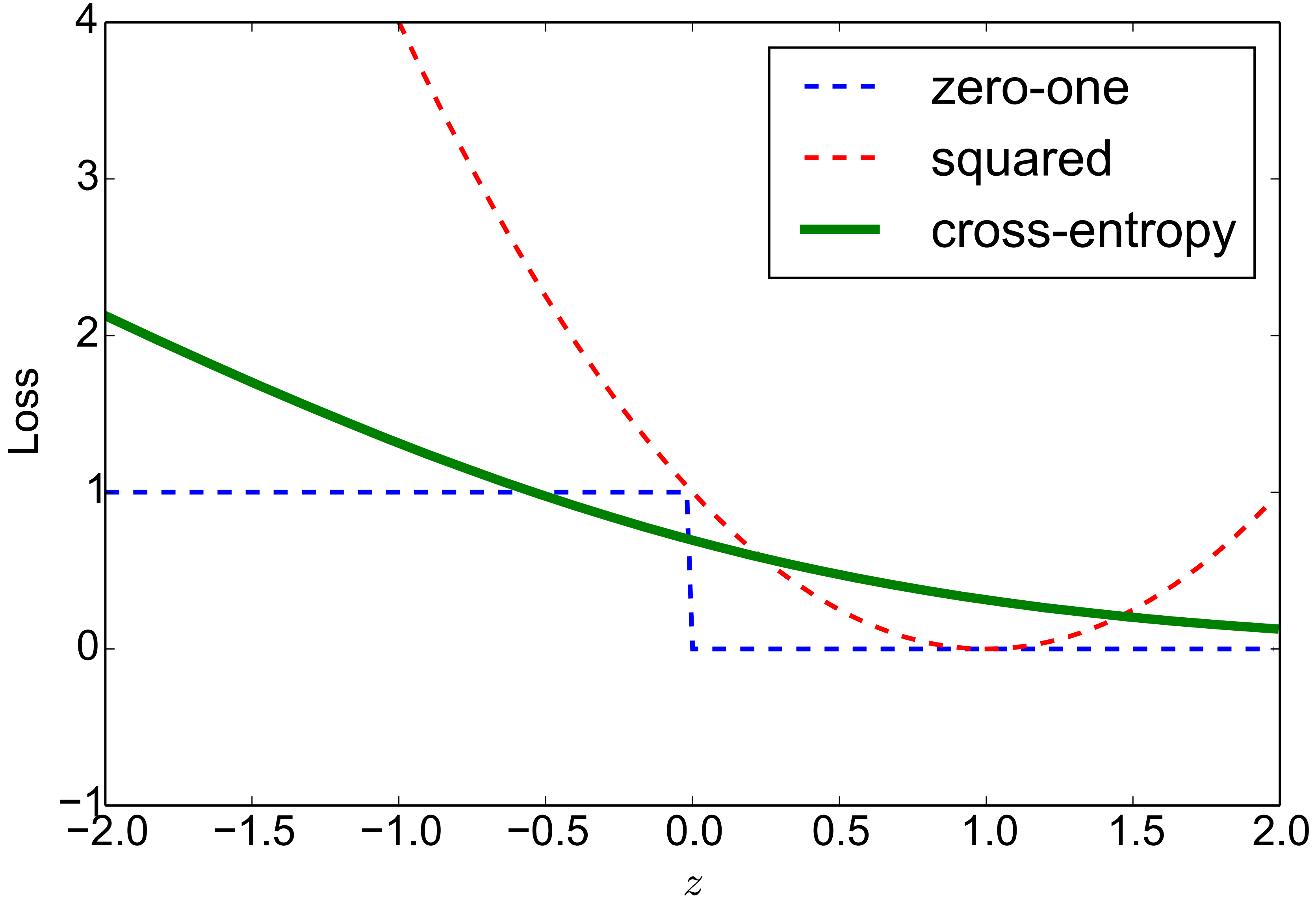
Logistic vs Linear: loss

- For linear regression use squared loss

$$\ell_{\text{squared}}(z) = (z - y)^2$$

- For logistic regression use log loss

$$\ell_{\log}(z) = -y \log(p_{\text{pred}}) - (1 - y) \log(1 - p_{\text{pred}})$$



zero-one
squared
cross-entropy
Loss
z
4
3
2
1
0
−1
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0

Logistic vs Linear: SGD

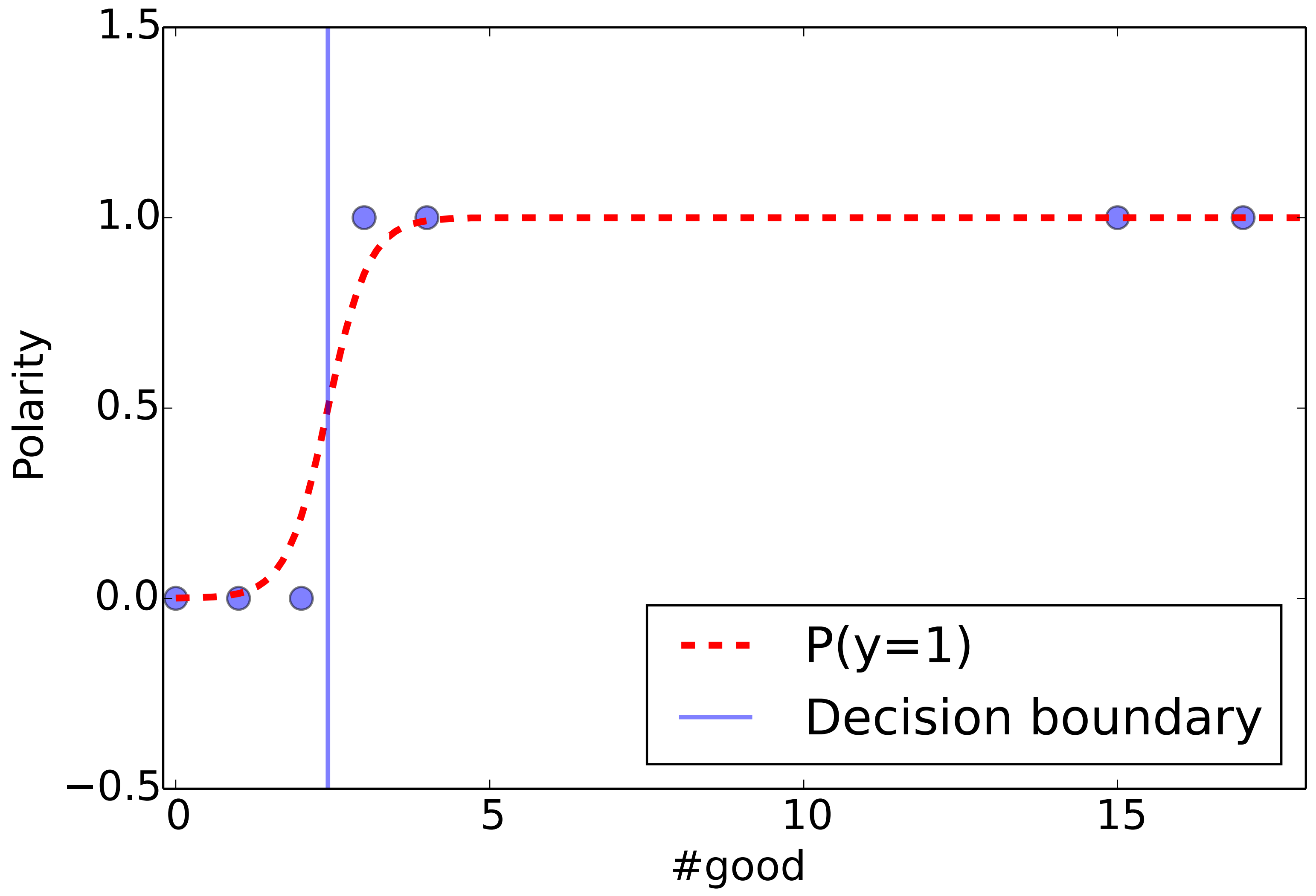
- Both models can be learned via **SGD**
- Linear regression

$$\mathrm{w}_{\text{new}} = \mathbf{w}_{\text{old}} - \eta \times 2(y_{\text{pred}} - y)\mathbf{x}$$

- Logistic regression

$$\mathrm{w}_{\text{new}} = \mathbf{w}_{\text{old}} + \eta \times (y - p_{\text{pred}})\mathbf{x}$$

Example



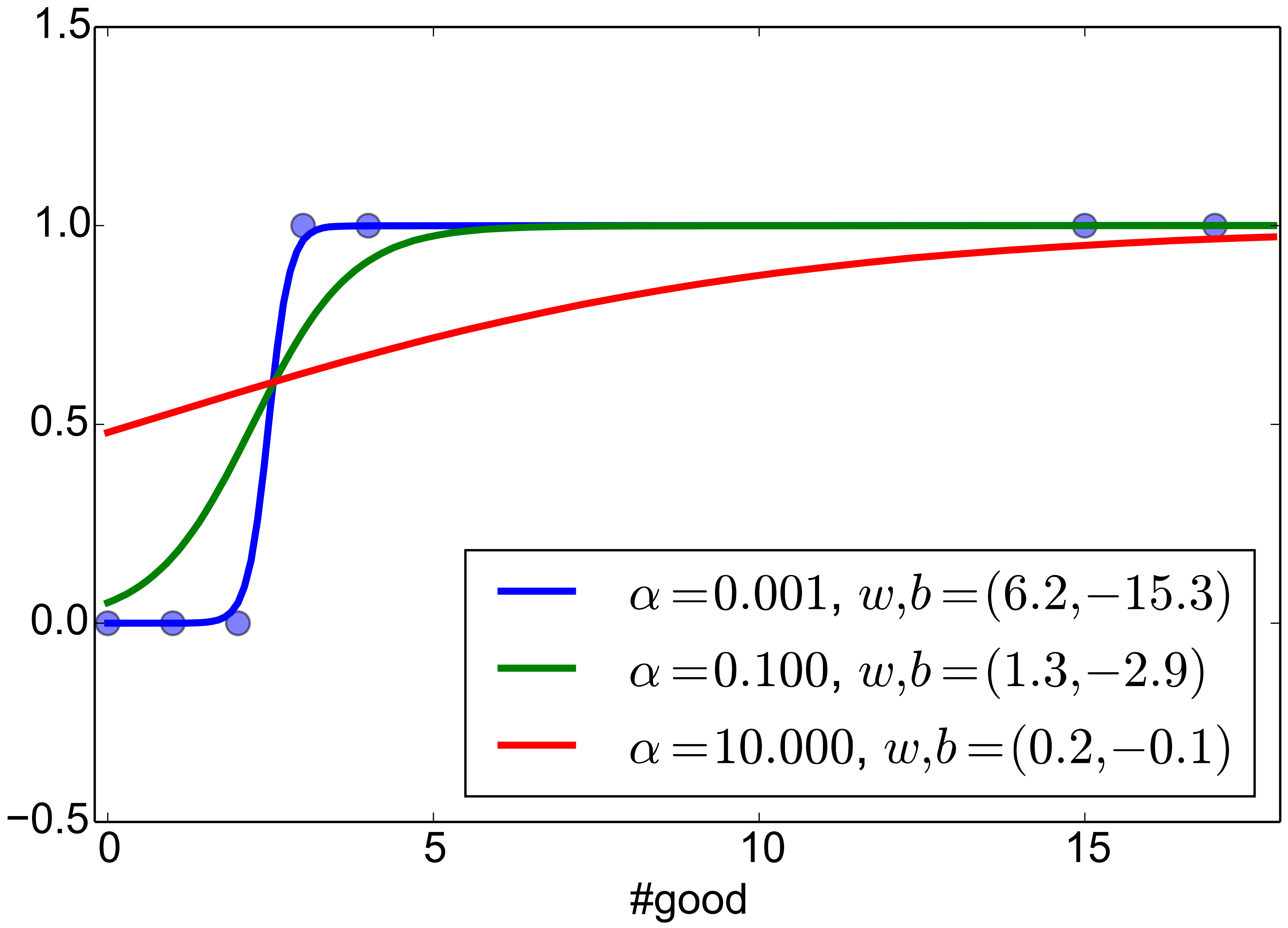
Regularization for control of overfitting

- Control how hard the model tries to fit to training data
- Models with small weights less flexible – less freedom to fit data
- Penalize large weights to control overfitting

L2 regularization penalty

$$\mathrm{Error}(\mathbf{w}, b) = \sum_{i=1}^{N} \ell(z^i) + \alpha \sum_{j=1}^{M} w_j^2$$

- We add a term to the error function which penalizes large weight
- Parameter ? controls strength of penalty



1.5
1.0
0.5
0.0
−0.5
Polarity
0
5
10
15
#good
$\alpha = 0.001,\ w,b = (6.2, -15.3)$
$\alpha = 0.100,\ w,b = (1.3, -2.9)$
$\alpha = 10.000,\ w,b = (0.2, -0.1)$

Logistic regression in scikit-learn

- `sklearn.linear_model.SGDClassifier`
  - Supports several loss functions including log loss
  - Good choice for large datasets
  - Regularization via parameter **alpha**
- `sklearn.linear_model.LogisticRegression`
  - Specialized LR algorithm
  - Good for small and medium data sizes
  - Regularization via param **C=1/alpha**

Summary

- Different loss functions give rise to different linear models
- Logistic regression for probabilistic classification
- Control overfitting via L2 regularization