Linear Models and ML Fundamentals II

ML4SE

Denis Litvinov

September 28, 2022

Table of Contents

Maximum Likelihood Estimation

MLE is a method of point estimation of parameters of a distribution, given observed data samples. Let $f_\theta(x)$ - probability density of x. Then function $f(X, \theta)$ with fixed θ is called Likelihood.

$$f(X, \theta) = \prod_{i=1}^{N} f(x_i, \theta)$$

Maximum Likelihood Estimation is

$$\hat{\theta} = \arg\max_{\theta} f(X, \theta)$$

Properties of MLE:

- Consistent

$$\forall \varepsilon > 0, \lim_{n \to \infty} \mathbb{P}(|\hat{\theta} - \theta| > \varepsilon) = 0$$

- Efficient. This means that no consistent estimator has lower asymptotic mean squared error than the MLE.

$$\forall \hat{\theta}_2 \neq \hat{\theta}_{MLE}, \; E_\theta[\hat{\theta}_{MLE} - \theta] \leq E_\theta[\hat{\theta}_2 - \theta]$$

L_2 regularization and early stopping 1

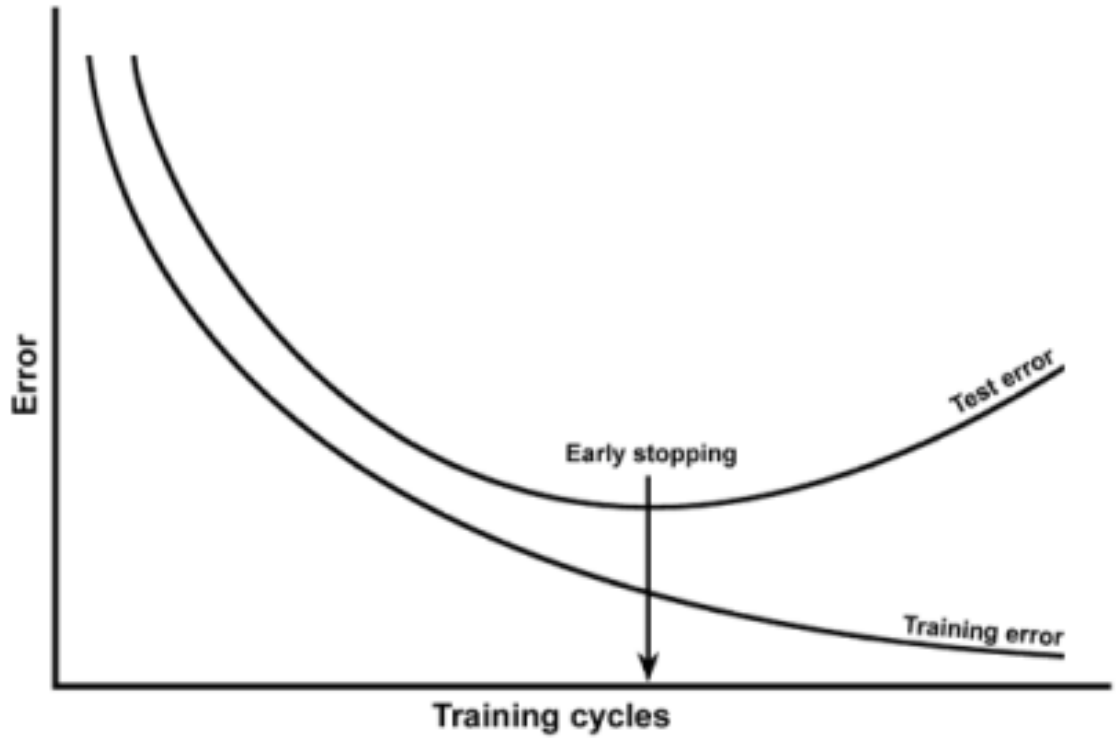
In Iterative solution we have mentioned some stopping criteria. We can imagine another one called *early stopping*:

1. split data into train and validation subsets
2. update model weights w on train dataset
3. keep track of the loss value on validation dataset
4. if on several consecutive iterations values of the loss function on validation dataset grows, than overfitting is observed $\rightarrow$ stop training

It can be shown, that number of consecutive iterations before early stopping τ can be expressed by coefficient of L_2 regularization λ

$$\tau \sim \frac{1}{\lambda}$$

L_2 regularization and early stopping 2



L_1 Regularization and sparsity 1

What if we use other norm for regularization?

$$R(w) = ||w||_1$$

For MSE with L_1 regularization

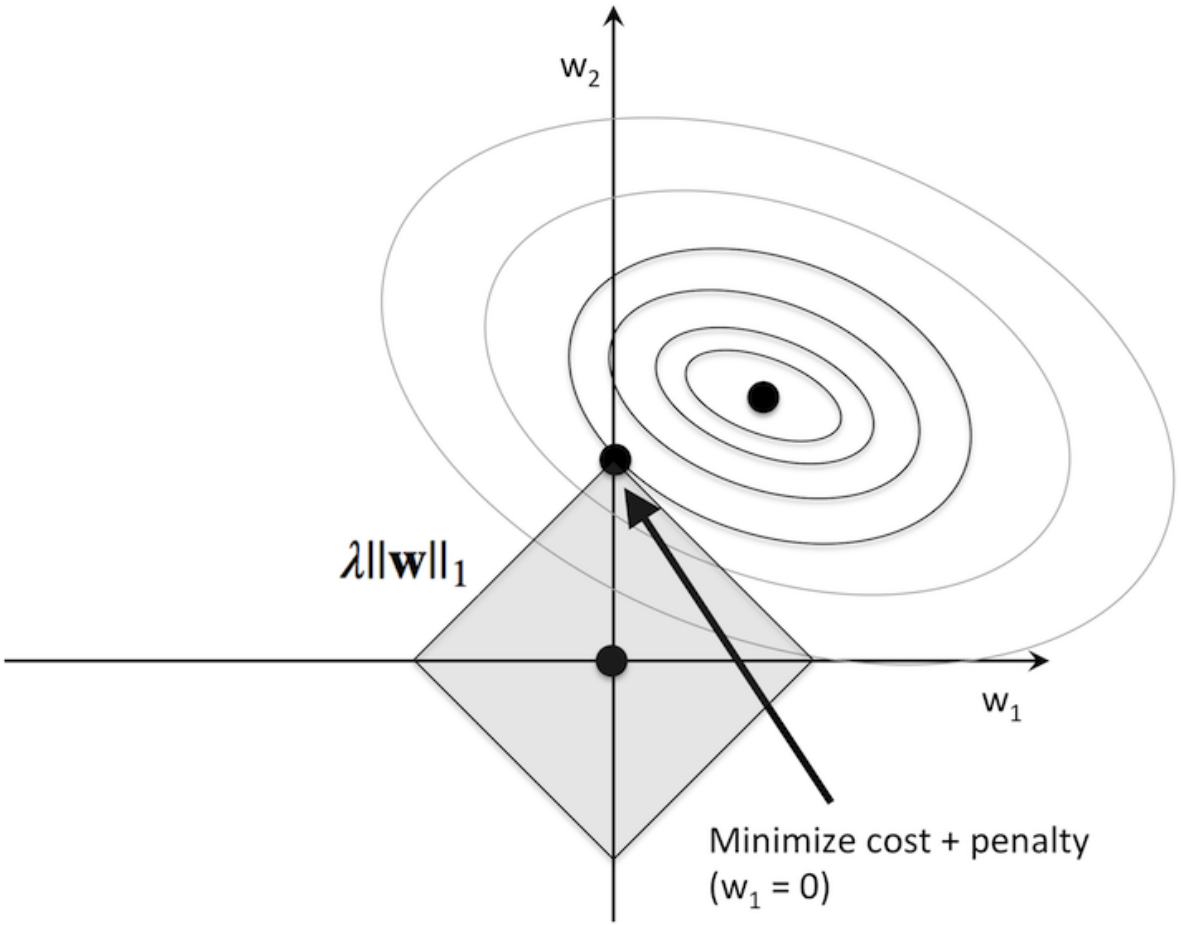
$$L_{MSE} = \frac{1}{N}||y - Xw||_2^2 + \frac{\lambda}{2}||w||_1$$

$$\nabla_w L_{MSE} = \frac{1}{N}X^T(Xw - y) + \lambda sign(w)$$

- L_1 norm is not differentiable at $w = 0$, but can be lower bounded by surrogate gradients (just say $\nabla_w R(0) \in [-1, 1]$)
- Gives sparse solutions: some of w components are 0
- Bayesian view on L_1 norm regularizer is a Laplacian prior on weights

$$P(x|\mu, b) = \frac{1}{2b}e^{-\frac{|x-\mu|}{b}}$$

L_1 Regularization and sparsity 2



L_1 Regularization and sparsity 3

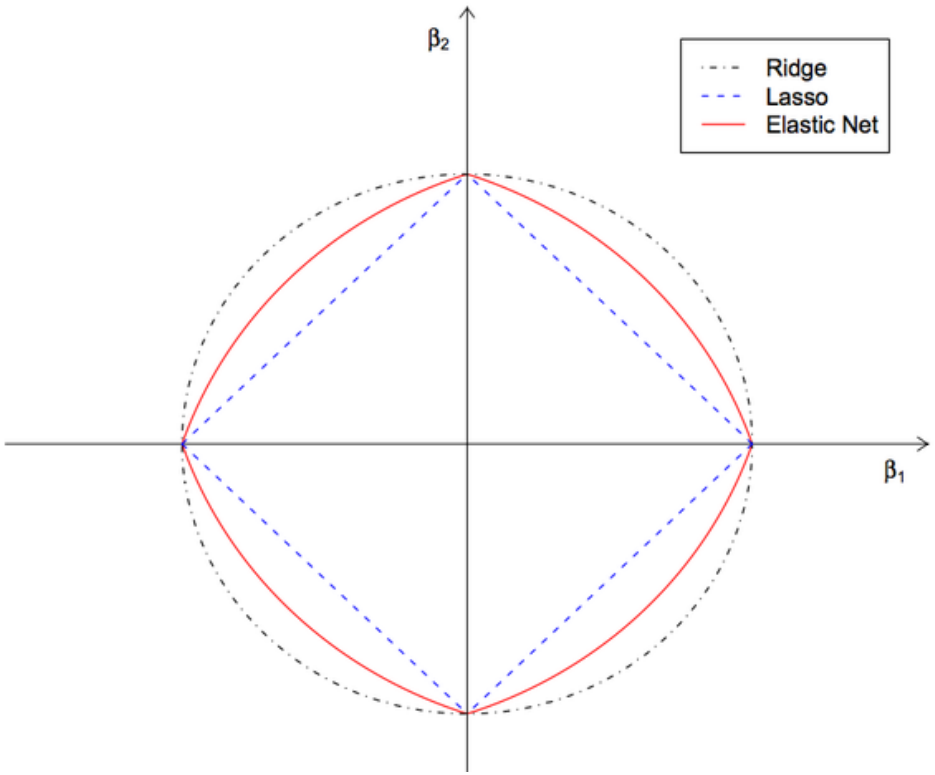
- L_1 can be used for feature selection
- **Remember, any L_p norm regularization shifts optimal solution w_***
- For linear models, if we want to make predictions with feature selection
  1. Train linear model with L_1 regularizer and select features with $|w_i| > 0$
  2. On selected subset of features, train linear model L_2 and use it for final prediction
- **Remember about situation with correlated features**

Elastic Net

$$L_{reg}(y, \hat{y}, w) = L(y, \hat{y}) + \lambda_1 ||w||_1 + \lambda_2 ||w||_2^2$$

Usually we would like to have convex combination in the form

$$L_{reg}(y, \hat{y}, w) = L(y, \hat{y}) + \lambda_1 ||w||_1 + (1 - \lambda_1) ||w||_2^2$$



Bias Variance Decomposition 1

Suppose our data is generated by:

$$y = f(x) + \epsilon$$

, where $\epsilon \in N(0, \sigma)$ is white noise.
We want to build such estimator, that:

$$\hat{y} = h(x)$$

is our prediction
Consider MSE regression

Bias Variance Decomposition 2

$$
\begin{aligned}
MSE &= E[(y - h(x))^2] \\
&= E[(y - f(x) + f(x) - h(x))^2] \\
&= E[(y - f(x))^2] + E[(f(x) - h(x))^2] - 2E[(y - f(x)(f(x) - h(x))] \\
&= E[\epsilon^2] + E[(f(x) - h(x))^2] - 2(E[yf(x)] - E[yh(x)] - E[f^2(x)] \\
&+ E[f(x)h(x)])
\end{aligned}
$$

Notes:

- since f is deterministic then $E[f^2(x)] = f^2(x)$
- since $E[y] = f(x)$ then $E[yf(x)] = f^2(x)$
- $E[yh(x)] = E[f(x)h(x)] + E[\epsilon h(x)] = E[f(x)h(x)] + 0$

Bias Variance Decomposition 3

$$
\begin{aligned}
MSE &= E[\epsilon^2] + E[(f(x) - h(x))^2] - 2(f^2(x) - E[f(x)h(x)] \\
&\quad + 0 - f^2(x) + E[f(x)h(x)]) \\
&= E[\epsilon^2] + E[(f(x) - h(x))^2] \\
&= E[\epsilon^2] + E[(f(x) - E[h(x)] + E[h(x)] - h(x))^2] \\
&= E[\epsilon^2] + E[(f(x) - E[h(x)])^2] + E[(E[h(x)] - h(x))^2] \\
&\quad + 2E[(E[h(x)] - h(x))(f(x) - E(h(x))] \\
&= E[\epsilon^2] + E[(f(x) - E[h(x)])^2] + E[(E[h(x)] - h(x))^2] \\
&\quad + 2(E[f(x)E[h(x)]] - E[E[h(x)]^2] - E[h(x)f(x)] + E[h(x)E[h(x)]])
\end{aligned}
$$

Notes:

- $E[fE[h(x)]] = f(x)E[h(x)]$
- $E[E[h(x)]^2] = E[h(x)]^2$
- $E[f(x)h(x)] = f(x)E[h(x)]$
- $E[h(x)E[h(x)]] = E[h(x)]^2$

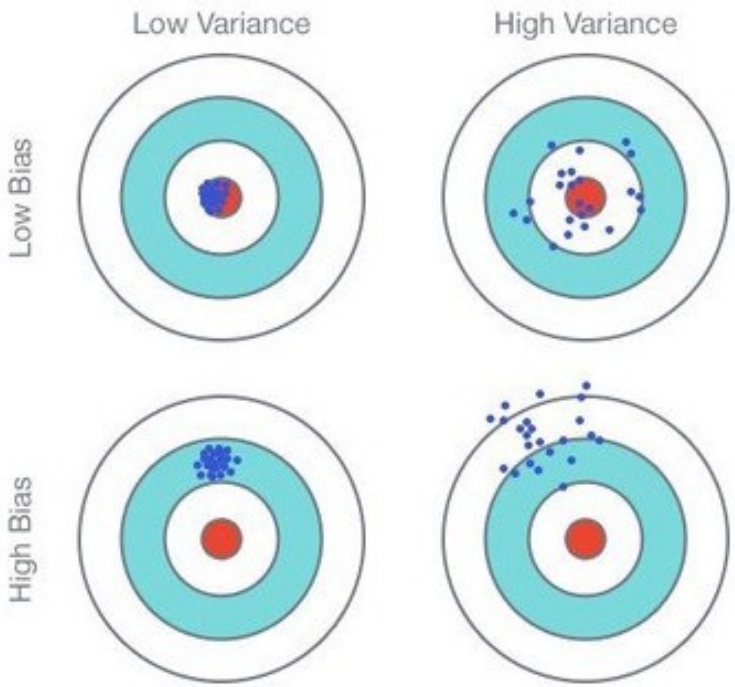
Bias Variance Decomposition 4

$$
\begin{aligned}
MSE &= E[\epsilon^2] + E[(f(x) - E[h(x)])^2] + E[(E[h(x)] - h(x))^2] \\
&\quad + 2(f(x)E[h(x)] - E[h(x)]^2 - f(x)E[h(x)] + E[h(x)]^2) \\
&= E[\epsilon^2] + E[(f(x) - E[h(x)])^2] + E[(E[h(x)] - h(x))^2] \\
&= Var[\epsilon] + E[(f(x) - E[h(x)])^2] + Var[h(x)] \\
&= Var[\epsilon] + bias^2 + Var[h(x)]
\end{aligned}
$$

Bias Variance Decomposition 5

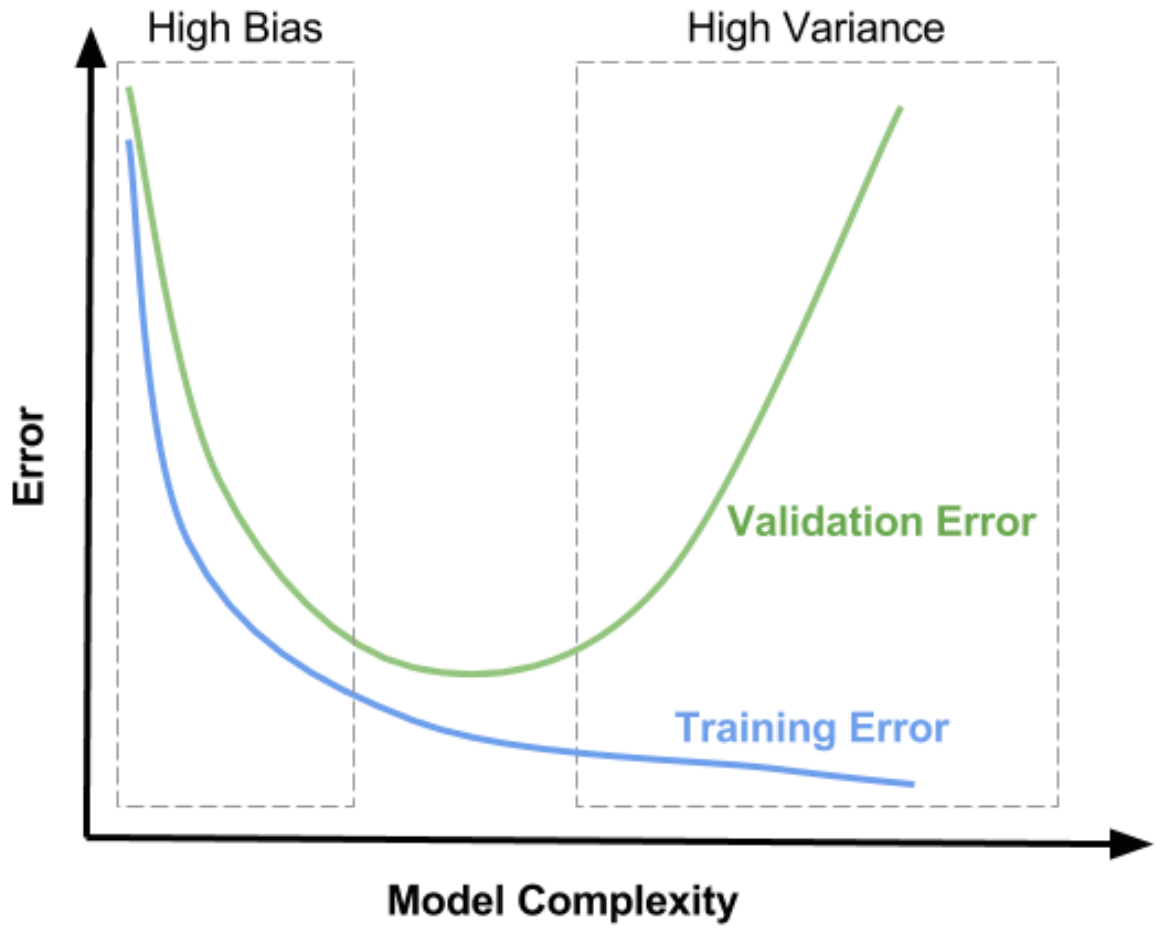
So prediction error can be decomposed into:

- variance of the noise
- bias of prediction
- variance of prediction



Validation curve 1

Validation curve is a dependence of model performance on the model complexity



Validation curve 2

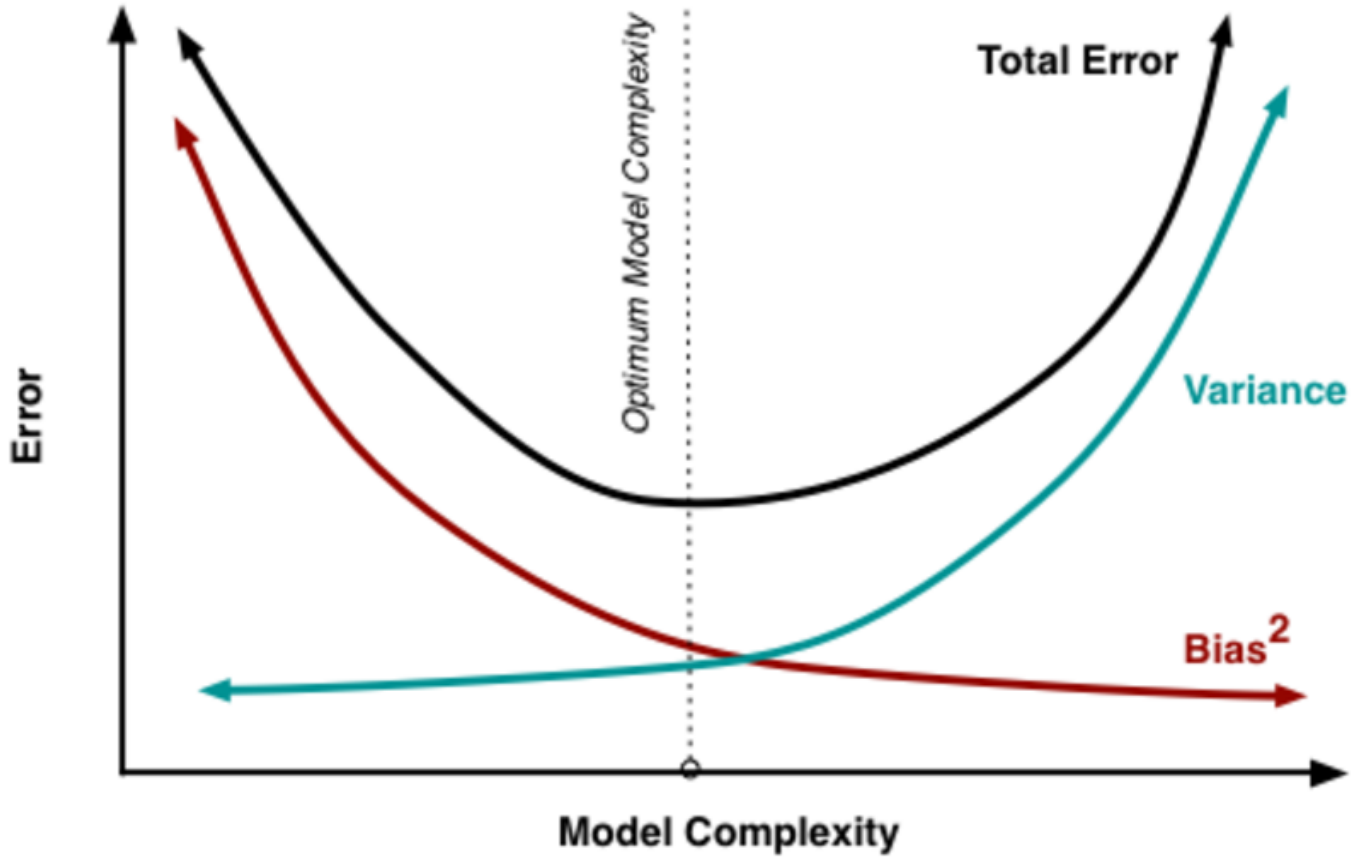


Figure: Generalization error

Learning curve 1

Learning curve is a dependence of model performance on the size of training dataset.

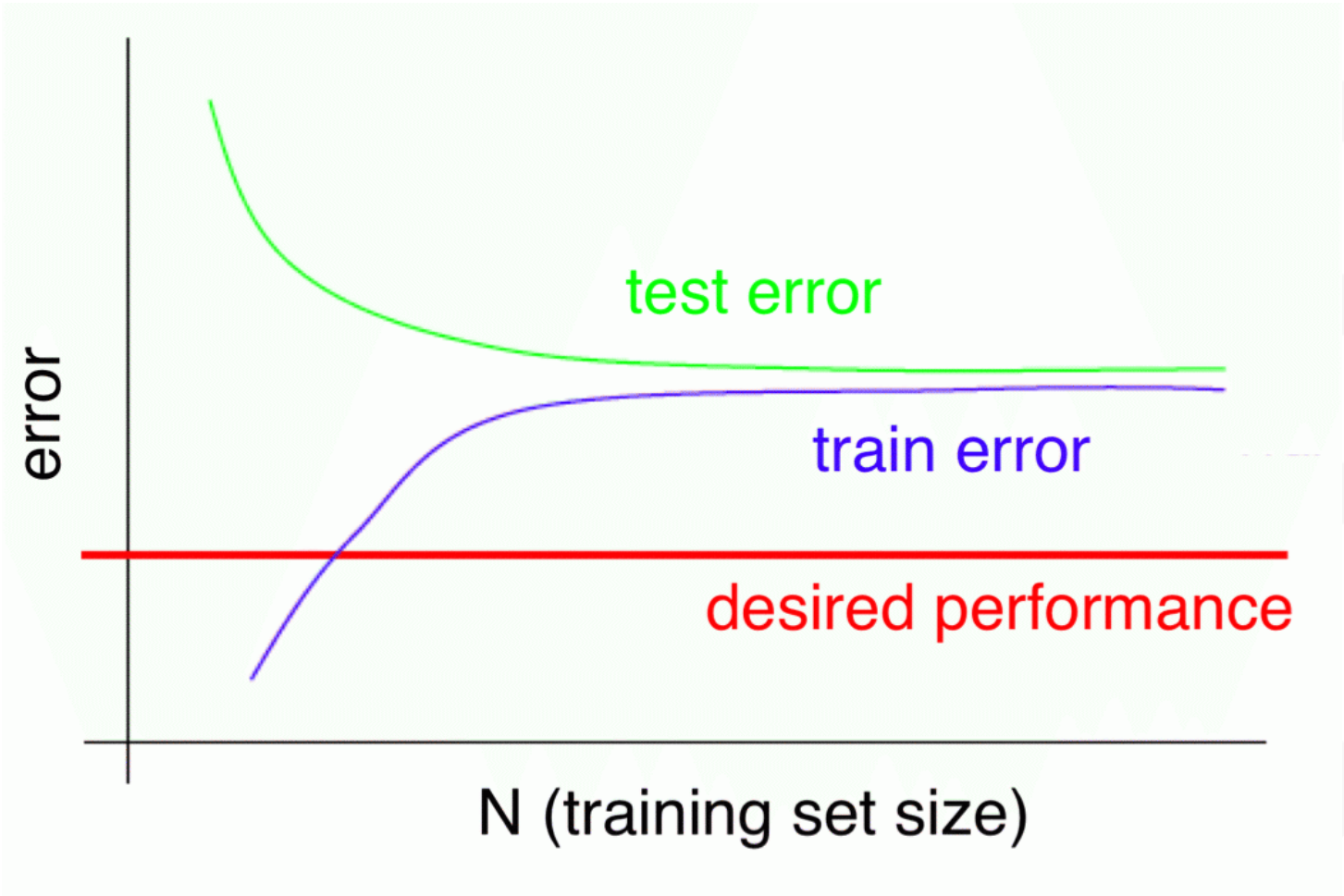


Figure: High bias

Learning curve 2

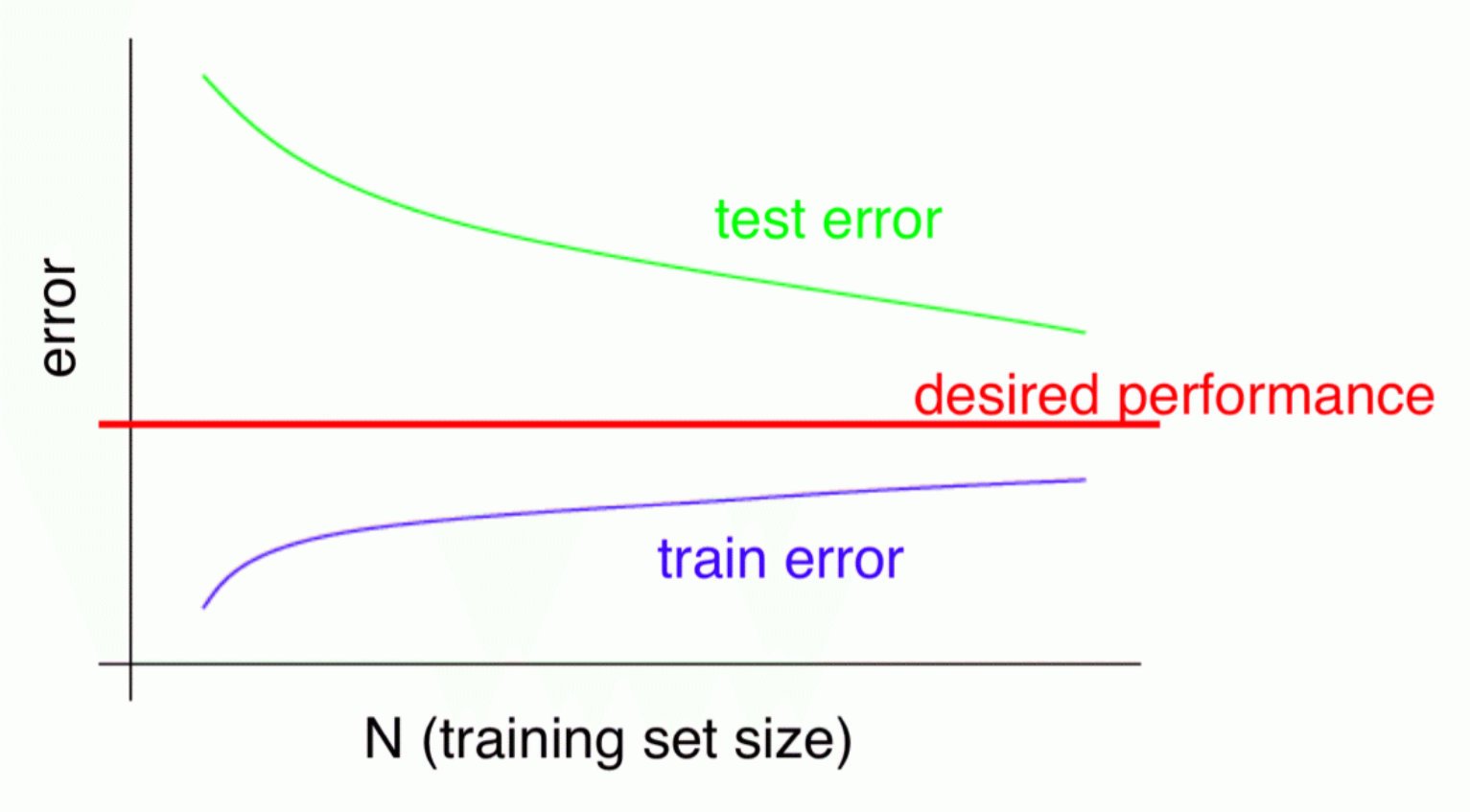


Figure: High variance

Problem Statement

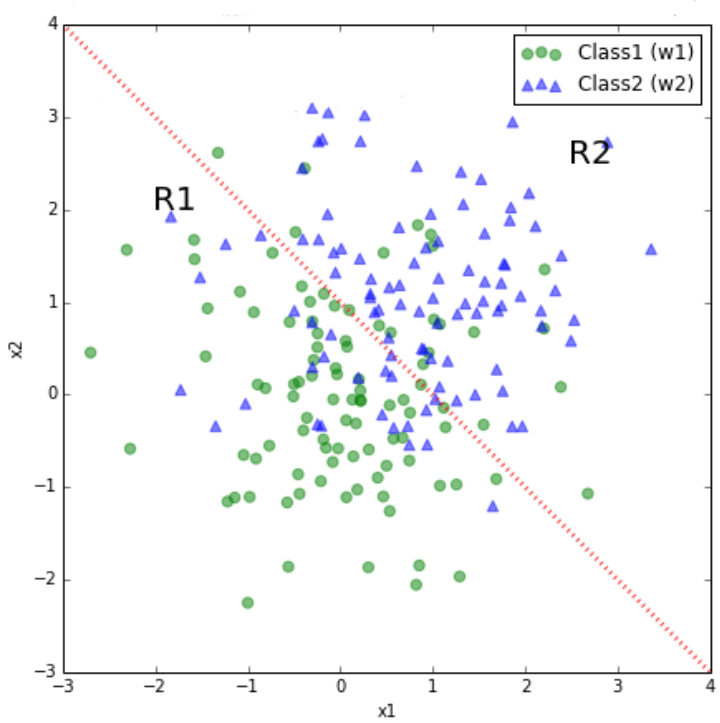
Given dataset $\{(x_i, y_i)\}_{i=1}^{N}$ of i.i.d. objects

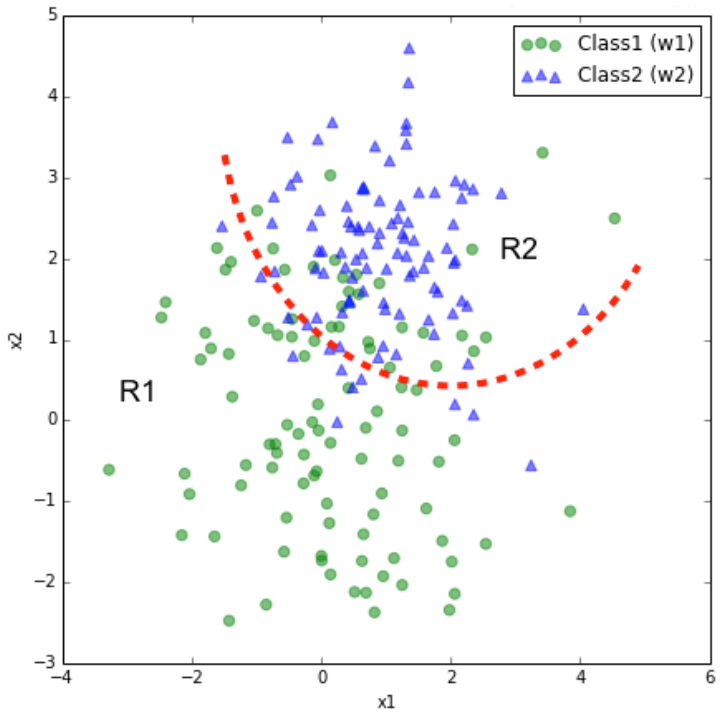
Or, equvalently given:

$X \in R^{NxD}$ - feature matrix, where D is dimension of feature space and N - number of objects.

$Y \in \{0, 1\}^N$ - target vector

Sometimes we will use notation $Y \in \{-1, 1\}^N$





Bayesian Classifier

Bayesian classifier is the best possible classifier given we know all joint distribution $P(x, y)$ of features and labels. (Which is an unrealistic assumption.)
Bayesian risk:

$$R = \sum_{x,y} I[h(x) \neq y] P(x, y) c_y$$

, where c_y is cost function for misclassification, e.g. $\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$ or $\begin{pmatrix} 0 & 0.7 \\ 0.3 & 0 \end{pmatrix}$

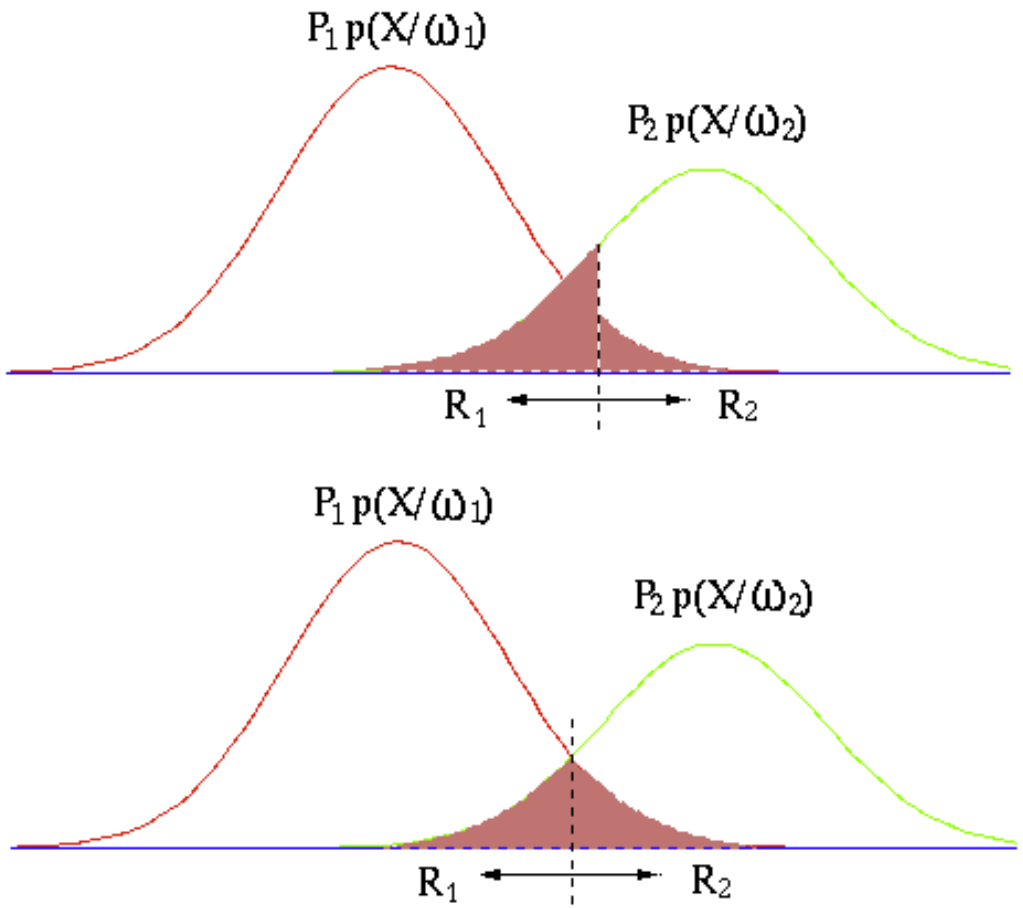
function $h(x)$ - decision function.
Which is minimized by

$$h(x) = \arg\max_y P(y|X)$$

$$h(x) = \arg\max_y P(X|y) P(y) c_y$$

Bayesian Classifier

$\Longrightarrow$ Cost function and prior class probabilities are interchangeable!



Naive Bayes Classifier

In Naive Bayes Classifier the features are assumed independent.

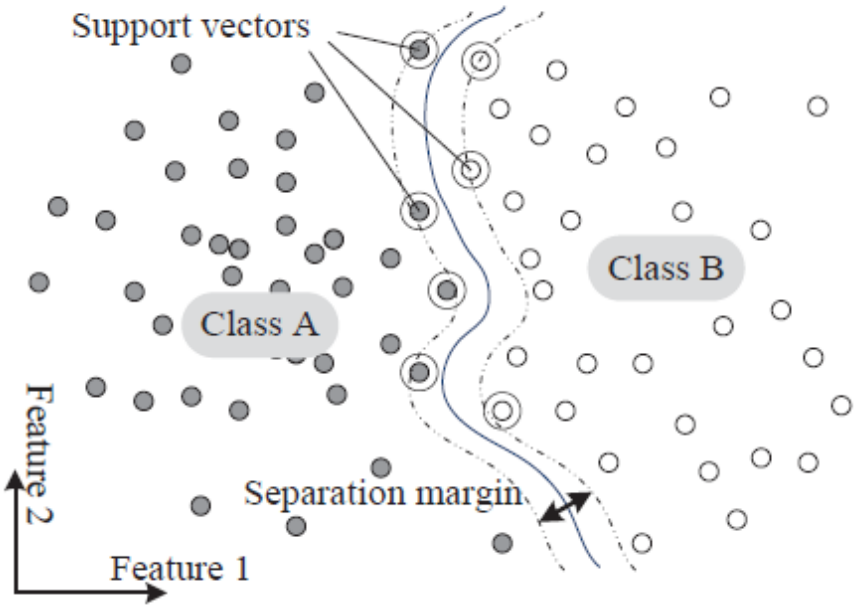
$$p(y|X) = \frac{p(y)p(x|y)}{p(x)}$$

$$p(y|X) = \frac{p(y)\prod_{i=1}^{N} p(x_i|y_i)}{p(x)}$$

$$\hat{y} = \arg\max_{k=1..K} p(y=k)\prod_{i=1}^{N} p(x_i|y=k)$$

Margin

For binary classification with $y \in \{-1, 1\}$ a variable $z = yh(x)$ is called **margin**.
Positive margin corresponds to successful classification, negative margin corresponds to error.
$|yh(x)|$ is a distance to decision boundary, which can be interpreted as confidence in classification of the object.



Logistic Regression

Suppose $\hat{y} = h(x)$ and $y \in \{0, 1\}$.
Show, that $h(x_i)$ should be $p(y = 1|x_i)$.
Probability to generate such samples from the point view of $h(x)$ is a likelihood L:

$$L = \prod_{i=1}^{N} h(x_i)^{[y_i=1]}(1 - h(x_i))^{[y_i=0]} \to \max_h$$

$$\log L = \sum_{i=1}^{N} [y_i = 1] \log h(x_i) + [y_i = 0] \log(1 - h(x_i)) \to \max_h$$

$$-\log L = -\sum_{i=1}^{N} [y_i = 1] \log h(x_i) + [y_i = 0] \log(1 - h(x_i)) \to \min_h$$

$$E[-\log L|x] = -p(y = 1|x) \log h(x) - p(y = 0|x) \log(1 - h(x)) \to \min_h$$

Logistic Regression

$$\frac{\partial E[-\log L|x]}{\partial h} = -\frac{p(y=1|x)}{h(x)} + \frac{1-p(y=1|x)}{1-h(x)} = 0$$

$$\implies h(x) = p(y=1|x) = \sigma(w^T x)$$

What are requirements for $\sigma(x)$?

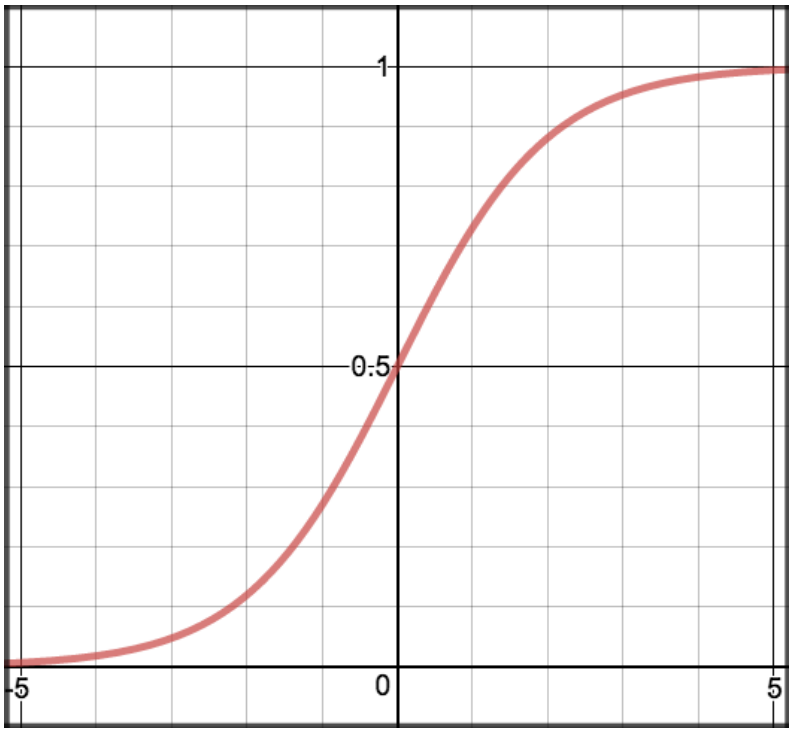
Logistic Regression

Requirements for $\sigma(x)$:

1. $\forall x \in R^D \ \sigma(x) \in [0, 1]$
2. $\sigma(0) = 0.5$
3. $\sigma(-x) = 1 - \sigma(x)$
4. σ - non-decreasing, continuous and differentiable

Sigmoid

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



Logistic Regression

You can find 2 different formulae for logistic loss: via cross-entropy as shown above

$$Loss(y_i, p_i) = -y_i \log p(y_i = 1|x_i) - (1 - y_i) \log(1 - p(y_i = 1|x_i))$$

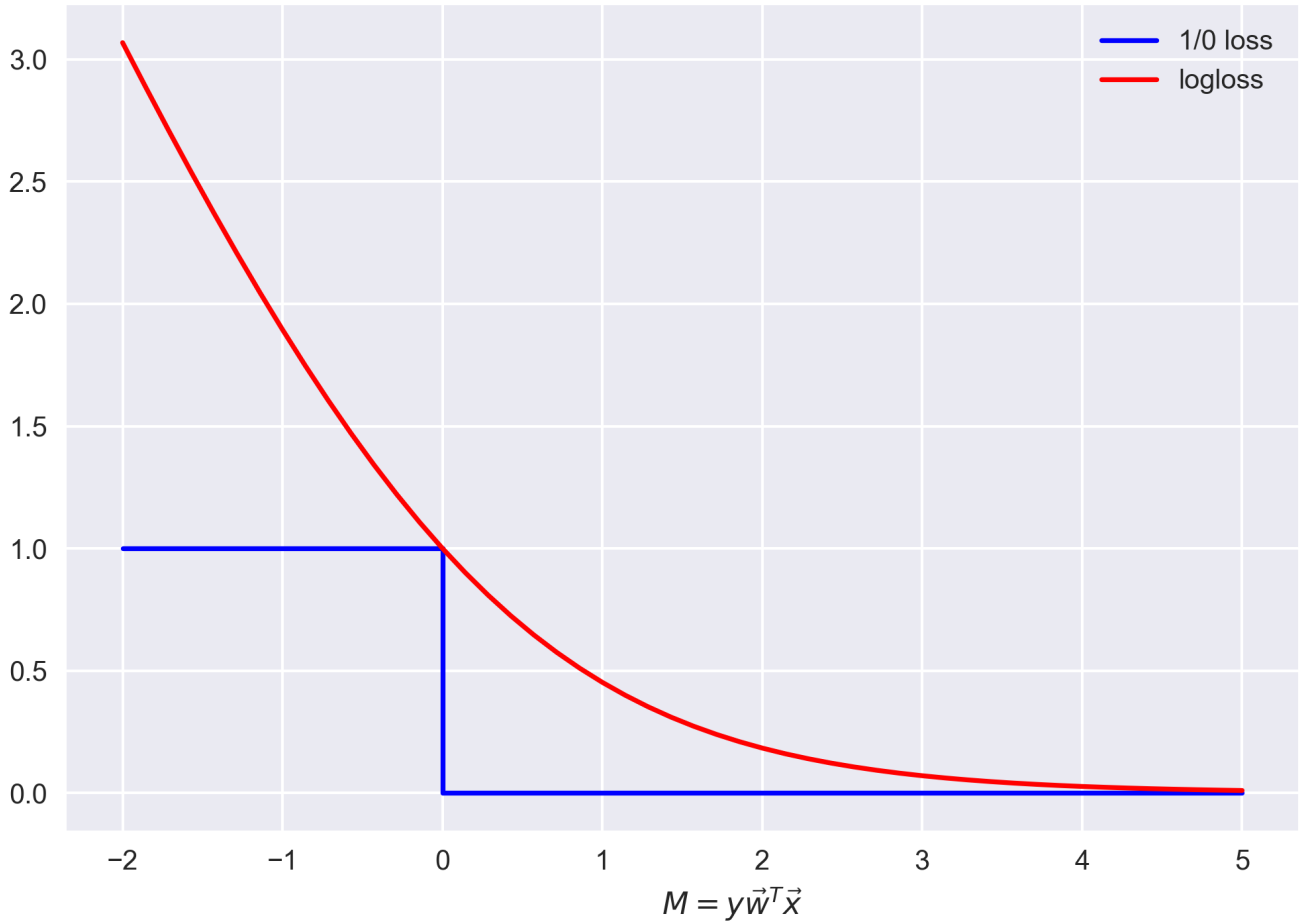
where probability of $y = 1$ class given sample x is

$$p(y = 1|x) = \sigma(w^T x) = \frac{1}{1 + e^{-w^T x}}$$

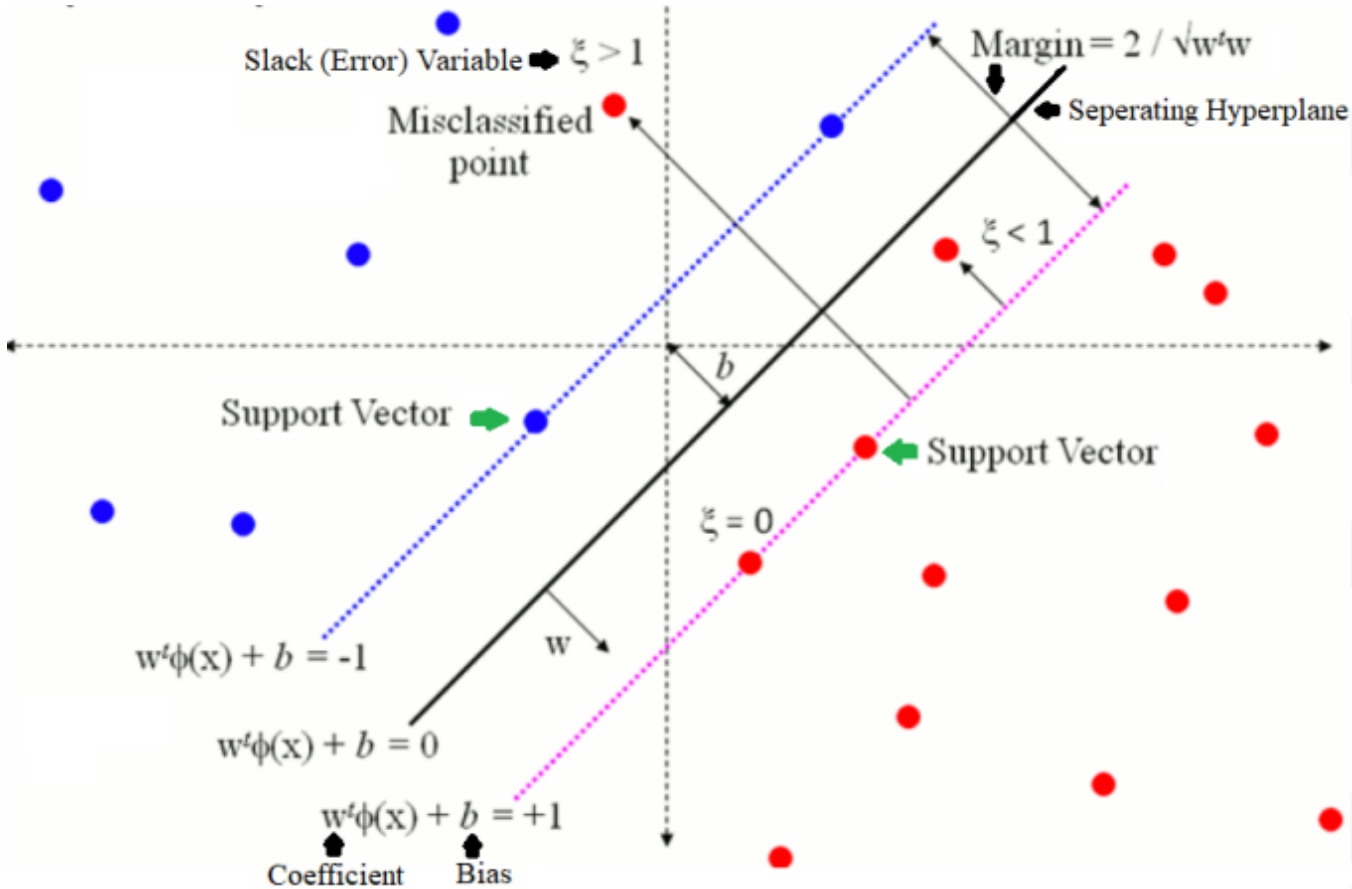
and with margins

$$Loss(y_i, x_i) = \log(1 + e^{-y_i w^T x_i})$$

Logistic Regression



Support Vector Machines



Support Vector Machines

Suppose we have some linear decision surface $h(x) = sign(w^T x)$
Then distance from point $x_0 \in R^D$ to decision surface is

$$\rho(x_0, h) = \frac{|w^T x|}{||w||_2}$$

Choose scale of w such that

$$\min_x |w^T x| = 1$$

Then distance from decision surface to the nearest object is

$$\min_x \frac{|w^T x|}{||w||_2} = \frac{1}{||w||_2} \min_x |w^T x| = \frac{1}{||w||_2}$$

For linear separable case we have optimization problem:

$$\begin{cases} \frac{1}{2}||w||_2^2 \to \min_w \\ y_i w^T x_i \geq 1 \end{cases}$$

Support Vector Machines

For linear inseparable case we introduce corrections for each object ξ_i:

$$\begin{cases} \frac{1}{2}||w||_2^2 + C\sum_{i=1}^{N} \xi_i \to \min_{w,\xi_i} \\ y_i w^T x_i \ \geq 1 - \xi_i \\ \xi_i \geq 0 \end{cases}$$

OR:

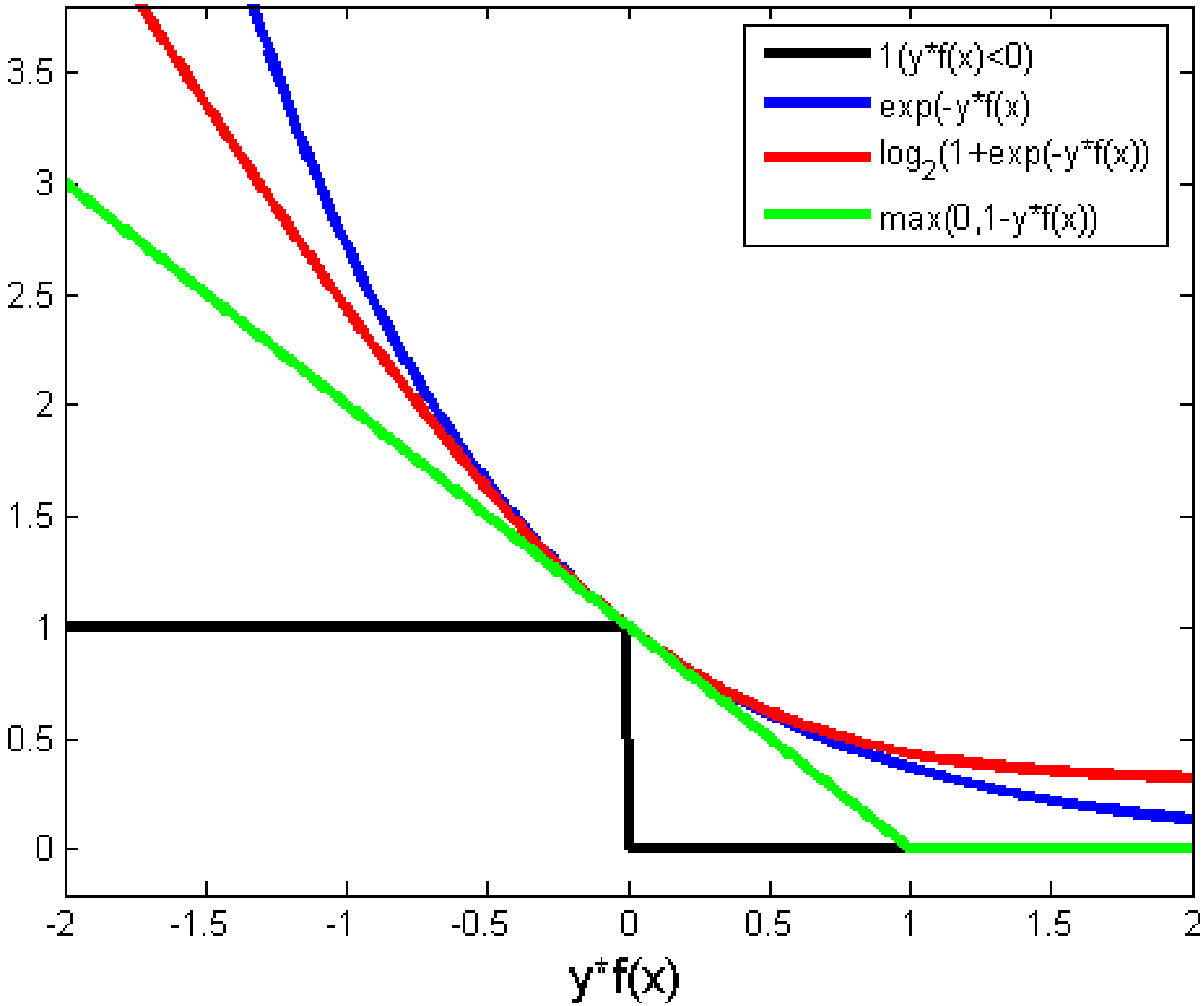
$$\xi_i = max(0, 1 - y_i w^T x_i)$$

Then,

$$Loss = \frac{1}{2C}||w||_2^2 + \sum_{i=1}^{N} max(0, 1 - y_i w^T x_i) \to \min_w$$

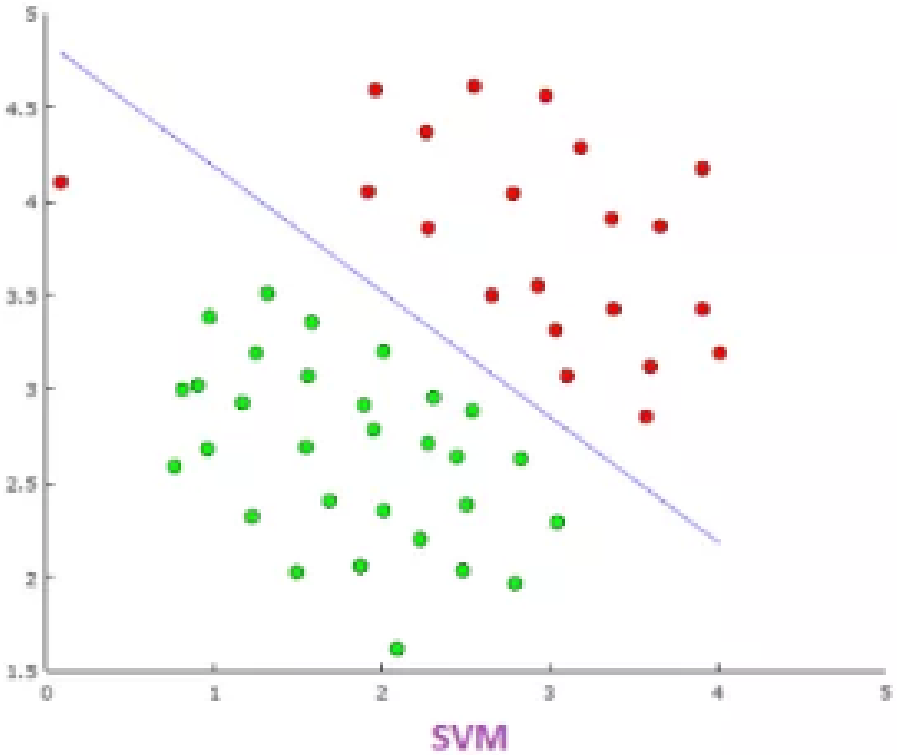
Unlike logistic regression, weight norm penalty already build in the model.

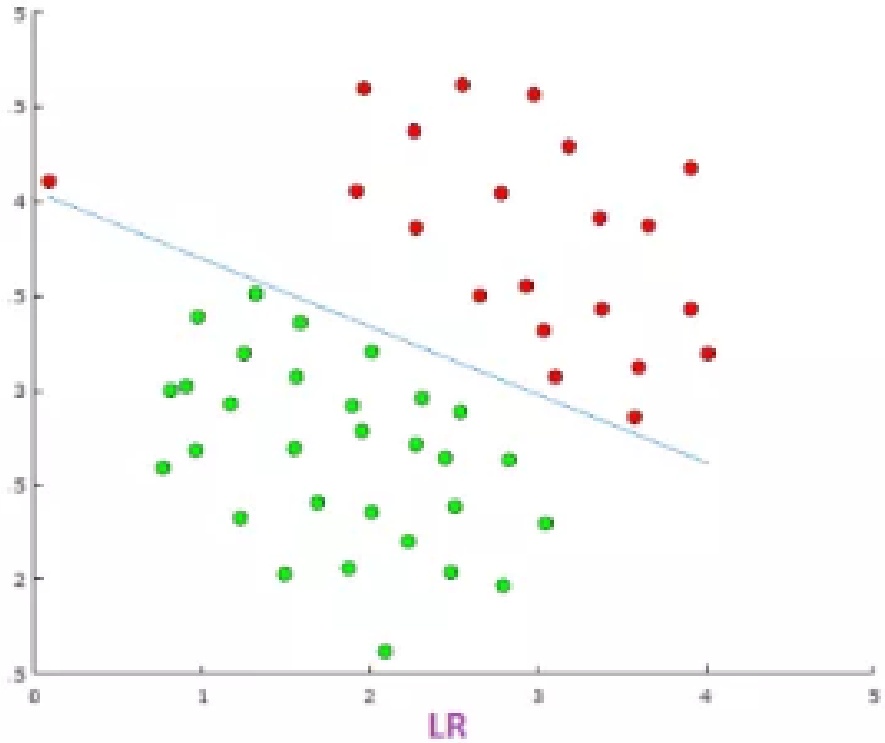
Losses


1(y*f(x)<0)
exp(-y*f(x)
log2(1+exp(-y*f(x))
max(0,1-y*f(x))
3.5
3
2.5
2
1.5
1
0.5
0
-2
-1.5
-1
-0.5
0
0.5
1
1.5
2
y*f(x)

Robustness

Model is called robust if its performance do not change significantly for new samples drawn from the same distribution $P(x, y)$.
In other way, model robustness depends on how the model handle the outliers.





F1 score

Accuracy $acc(y, \hat{y}) = \frac{1}{N} \sum_{i=1}^{N} [\hat{y}_i = y_i]$

		Predicted class P	**Predicted class** N
Actual Class	P	True Positives (TP)	False Negatives (FN)
	N	False Positives (FP)	True Negatives (TN)

Precision

$$Pr = \frac{TP}{TP + FP}$$

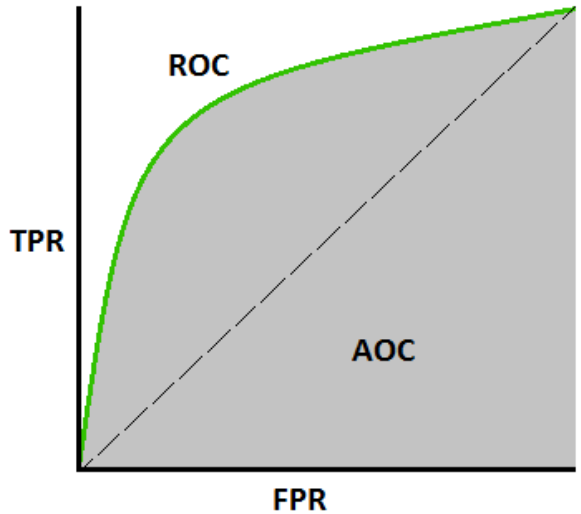
Recall

$$Re = \frac{TP}{TP + FN}$$

$$F1 = \frac{2 * Pr * Re}{Pr + Re}$$

AUC

$FPR = \frac{FP}{FP+TN}$ false positive rate
$TPR = \frac{TP}{TP+FN}$ true positive rate
AUC = area under the curve ROC
$ROC(t) = (TPR(t), FPR(t))$ is parametrized by threshold t on the probability $p(y = 1|x)$



How does class imbalance affect quality metrics given above?

Empirical Loss Minimization

In general, we want to optimize Expected Risk:

$$R = E[Loss(x, y)] = \int_{-\infty}^{\infty} Loss(x, y) dP(x, y) = Pr_{(x_i, y_i) \sim D}[Loss(x_i, y_i)]$$

But since we don't now the joint distribution $P(x, y)$, we can only deal with Empirical Risk (Loss functional):

$$\hat{R} = \frac{1}{N} \sum_{i=1}^{N} Loss(x_i, y_i)$$

Generalization error

How well does $\hat{R}$ approximates R?
Using Hoeffding's inequality it can be shown, that given N random examples, and $\forall \delta > 0$, with probability $Pr >= 1 - \delta$, the following upper bound holds on the generalization error of h:

$$R \leq \hat{R} + \sqrt{\frac{\ln(1/\delta)}{2N}}$$

For finite hypothesis space H under the same conditions:

$$R \leq \hat{R} + \sqrt{\frac{\ln|H| + \ln(1/\delta)}{2N}}$$

where $\ln|H|$ serves as a measure of model complexity.

Rademacher Complexity

How to approximately estimate model complexity?
Given a dataset $\{x_i\}_{i=1}^m$, $x \sim D$
Let $\sigma_i \sim random(\{-1, 1\})$ be random labels for each x_i.
Consider binary classification task. We expect a model to have high complexity if it can fit well any random label assignment on the dataset.
Empirical Rademacher Complexity for a given dataset can be estimated by expected average margin

$$\hat{R}_m(H) = E_\sigma[\max_{h \in H} \frac{1}{m} \sum_{i=1}^{m} \sigma_i h(x_i)]$$

Rademacher Complexity for model H can be expressed as

$$R_m(H) = E_D(\hat{R}_m(H))$$

We also say, that model with high complexity tends to memorize training samples.

Data Augmentation

Generalization error depends on number of samples N in the dataset.
How to increase N?
Apply some transformation to original data which is invariant to some desirable properties.